NATIONAL ACADEMY OF SCIENCES OF BELARUS
UNITED INSTITUTE OF INFORMATICS PROBLEMS

Yu.N. Sotskov,
N.Yu. Sotskova,
T.-C. Lai,
F. Werner

SCHEDULING UNDER UNCERTAINTY
THEORY AND ALGORITHMS

Minsk
"Belorusskaya nauka"
2010

UDC 519.8

Scheduling problems with uncertain input data are considered. It is assumed that the job processing time may take any value from a given interval. The main attention is paid to multi-stage processing systems with a fixed job route through the given machines. For different objective functions, algorithms are developed which are based on a stability analysis of an optimal schedule with respect to variations of the input data. The main results included in this monograph have been obtained by the authors in the period from 1990 to 2010.

This book will provide the reader with a comprehensive understanding of the mathematical models and algorithms appropriate for sequencing and scheduling under uncertainty and will be useful for applied mathematicians, students and Ph.D. students dealing with scheduling theory, optimization and calendar planning.

Tables 68. Figures 20. Bibliography 374.

R e f e r e e s:
doctor of sciences, professor V.A. Emelichev
doctor of sciences, professor M.K. Kravtsov

ISBN 978-985-08-1173-8

Contents

Preface 5

Introduction 9

Chapter 1. Stability Radius of an Optimal Schedule 17
1.1. Mixed Graphs for Modelling a General Shop 19
1.2. Regular Criterion 27
1.3. Maximum Flow Time 32
1.4. Mean Flow Time 43
1.5. Calculation of the Stability Radius 50
1.6. Experimental Design and Results 56
1.7. Comments and References 73

Chapter 2. General Shop with Interval Processing Times . . . 81
2.1. Minimal Dominant Set 82
2.2. Relative Stability Radius 94
2.3. Algorithms for Problem $\mathcal{G}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ 103
2.4. Dominance Relations 117
2.5. Characterization of a G-solution 124
2.6. Algorithms for Problem $\mathcal{J}/a_i \leq p_i \leq b_i/\Phi$ 150
2.7. Computational Results 163
2.8. Comments and References 173

Chapter 3. Two-Machine Flow Shop and Job Shop 179
3.1. Flow Shop with Interval Processing Times 180
3.2. Schedule Execution for a Flow Shop 201
3.3. Job Shop with Interval Processing Times 226
3.4. Flow Shop with Limited Machine Availability 234
3.5. Job Shop with Limited Machine Availability 257
3.6. Comments and References 274

Conclusion 289

Index 292

Bibliography 295

List of Tables 320

List of Figures 322

Curriculum Vitae 325

Preface

No time like the present

The idea of our approach for solving a scheduling problem with *uncertain* parameters arose at the United Institute of Informatics Problems of the National Academy of Sciences of Belarus, when the first author investigated the *stability* of an optimal schedule. This approach was further developed at the Institute of Mathematical Optimization of the Otto-von-Guericke University of Magdeburg, where the second author wrote her Ph.D. thesis "Optimal scheduling with *uncertainty* in the numerical data on the basis of *stability* analysis" under the supervision of the fourth author. Of course, the ideas and even the results based on these ideas are only necessary (but not sufficient) conditions for writing a monograph. What one needs else is *time*. Such a *time* was ensured by the National Scientific Council of Taiwan (projects NSC 94-2811-H-002-001, NSC 97-2410-H-002-107, NSC 98-2811-H-002-005). Due to these projects, the book was written in Taipei, when the first author visited the Department of Industrial and Business Management of the National Taiwan University. These projects were headed by the third author in the *time* from 1997 till 2009.

This book is about optimal *scheduling* with uncertain numerical data. A schedule is considered as a mathematical object and so it does not matter where the schedule is used. Optimal scheduling is considered as sequencing the given activities over *time* in order to meet some given objective. Activities (jobs, operations) represent processes which use resources (machines) to produce goods or provide services. In contrast to many other books that addressed sequencing and scheduling [25, 36, 37, 51, 81, 84, 122, 129, 183, 248, 253, 269, 270, 278, 289, 299, 335, 352, 355, 357], this book addresses the *stability* of an optimal schedule with respect to *uncertainty* of the input data. This book shows how one can use the stability of an optimal (or near optimal) sequence of the given activities to overcome the uncertainty of the numerical input data.

It is obvious that there is no unique method that will fit all the different

types of uncertainties arising in the real world, and the method described in this book aims to complement but not to replace other methods to deal with uncertainty in scheduling. In particular, a *stochastic* method (see the second part of monograph [269] and articles [33, 95, 108, 120, 121, 171, 176, 178, 179, 226, 268, 274, 275, 302, 303] among others) is useful when one has enough prior information to characterize the probability distributions of the random activity durations and there is a large number of repetitions of similar processes. However, a stochastic method may have a limited significance for a small number of realizations of the process.

In contrast to a stochastic model and a fuzzy model (see book [299] and articles [143, 196, 239, 254, 262, 280, 343] among others), we assume that in spite of the *uncertainty* of the input data, the desired *schedule* has to fix the unique place in the desired *sequence* for each given activity. Contrary to most methods for dealing with uncertainty in scheduling, the main aim of our method is to construct really an optimal schedule for the actual numerical input data. Of course, this is only possible when the level of uncertainty is not very high. Otherwise, a decision-maker has to use other approaches to scheduling under uncertainty, e.g., those presented and reviewed in [24, 26, 61, 68, 89, 107, 123, 159, 162, 184, 193, 221, 234, 243, 255, 260, 264, 269, 299, 362, 364, 370].

In order to outline the approach considered in this book let us discuss a connection between *time* which is necessary for solving a scheduling problem and *time* which is necessary for the realization of a schedule. What is the main role of *time* in practical scheduling? Since *time* has only one direction for variation (from present to future) a decision-maker may have definitely reliable information only about the activities which are already realized, and there may be not enough information about the realization of the future activities involved in a real schedule. In practice, it is often useful to take into account that a unit of *time* may have a different "price" while solving a scheduling problem. With this in mind, we distinguish several *time* phases in decision-making depending on the static or dynamic scheduling environment. The first, *off-line, proactive,* phase of scheduling is used before realization of the given activities (static scheduling environment) while the subsequent, *on-line, reactive* phases of scheduling are used when a part of the schedule has already been realized (dynamic scheduling environment).

At the off-line phase of scheduling, there may be more *time* for decision-making and as a result, the "price" of a *time* unit may be relatively low. However, before realization of the activities, there is usually not enough reliable information to construct actually an optimal schedule for the future

realization of the given activities. Thus, at the off-line phase of scheduling, it is necessary to create a solution (multiple schedules) to the scheduling problem provided that the main numerical input data are *uncertain*. At the off-line phase of scheduling, we propose to realize a *stability* analysis of the constructed feasible schedules to deal with different contingencies that may arise during the schedule execution.

At the on-line phases (in the dynamic scheduling environment when a part of the schedule has already been realized), there is more reliable information for the decision-making (the situation under consideration becomes more *certain*, e.g., due to known durations of the activities already executed). However, there is often not enough *time* for using this additional information in order to construct an optimal schedule. The main idea of the approach described in this book is to prepare for the on-line phases of scheduling when the "price" of a *time* unit will be higher than that at the off-line phase of scheduling.

We have to emphasize that the randomness of the activity durations considered in this book is due to external forces in contrast to scheduling problems with *controllable durations* [70, 71, 72, 76, 88, 165, 166, 173, 181, 296, 297, 345], where the objective is to determine optimally the durations (which are under control of a decision-maker) and the schedule at the same time. Another related yet different model is used for *hoist scheduling* (see articles [65, 67, 209, 233, 236, 342, 347] among others). Hoist scheduling problems arise in chemical, electroplating and medical industries, where the objective is to minimize the cycle time of a repetitive process. Due to the nature of the chemical process, the activity durations have to be strictly controlled by a decision-maker within the given lower and upper bounds.

The processing systems considered in this book are those arising in *practice* for scheduling in continuous manufacturing industries (e.g., process industries) and in discrete manufacturing industries (e.g., cars, semiconductors). Continuous manufacturing industries often have two types of machines: Machines for the main operations (e.g., paper mills, steel mills, aluminum mills, chemical plants, refineries) and machines for finishing operations such as cutting of the material, bending, folding, painting and printing. Medium and short term scheduling of the latter operations often leads to *single machine* scheduling problems. Scheduling the main production operations leads to *flow shop* and *job shop* scheduling problems. The most general processing system considered in this book is a so-called *general shop*.

Within this book, we widely use mixed graphs for modelling the scheduling environment with different types of uncertainties. All chapters of the

book are written as an independent one of others and as self-sufficient for understanding. A summary, remarks and bibliographic notes are given at the end of each chapter in the section titled "Comments and References", where publications are cited in which the results presented in the chapter have been originally proven. We attempted to minimize the number of references used within other sections of the book.

It is clear for us that we would not have been able to write this book without many people that helped us. During the last more than ten years, where we worked on the results presented in this book, we have received support of many people from the United Institute of Informatics Problems of the National Academy of Sciences of Belarus, the Otto-von-Guericke University of Magdeburg, and the National Taiwan University. In particular, we acknowledge Georgii V. Andreev and Natalja G. Egorova from the United Institute of Informatics Problems for their qualified help in programming and testing algorithms. We would like to thank Natalja M. Matsveichuk from the same institute for her new results on the two-machine flow shop and job shop problems with uncertain numerical data included in Chapter 3 of this book. Special thanks go to Nina Sotskova and Ivan Mihov for their enthusiasm, faith and love which were of great help for us during the period of writing this book. We would like to gratefully acknowledge the collaborators of the Department of Industrial and Business Management of the National Taiwan University for their good attitude and for creating an optimal atmosphere for our research and writing this book. As it was already mentioned, the authors obtained support from the National Scientific Council of Taiwan.

We dedicate this book to Evangelina and Veronika Mihova.

Minsk, München, Taipei, Magdeburg
June 2010

Yuri N. Sotskov
Nadezhda Yu. Sotskova
Tsung-Chyan Lai
Frank Werner

Introduction

A schedule is the greatest
production of human spirit
K. Chapek

Scheduling concerns the allocation of given sets of activities (*jobs*) to given resources (*machines*) over time. In real life, machines and jobs may take different forms, e.g., machines in a workshop and operations in a production process, runways at an airport and take-offs and landings of aircrafts, crews at the construction site and stages in the construction project, processing units in a computing environment and executions of computer codes, teachers at the university and student groups. Jobs may differ one from another by processing times, priorities, release dates, due dates, machine types which can execute it, etc.

Mathematical problems arising in scheduling are studied in *scheduling theory* containing two main parts based on deterministic or stochastic models. *Deterministic models* have been introduced for scheduling environments (see [37, 51, 122, 204, 278, 352, 355] among others) in which the processing time (duration) of each job processed by a machine is supposed to be given in advance (before applying a scheduling procedure) and assumed to be a *constant* during the realization of a schedule. Often in real life, however, exact numerical data are not known in advance, and difficulties arise when some job processing times (which were assumed to be known in advance) will vary due to a change in a dynamic environment. Even if all the processing times are fixed before scheduling, one is forced to take into account possible errors within the realization of a schedule, the precision of the equipment for calculating the processing times, round-off errors in the calculation of a schedule on the computer, machine breakdowns, arriving additional jobs with high priority and so on. The inadequacy of a deterministic scheduling problem in modelling real-world situations was emphasized in several publications, e.g., in [24, 59, 89, 100, 144, 152, 155, 159, 193, 235, 240, 241, 242, 243, 247, 264, 269, 273, 292].

More general scheduling settings have been considered using a *stochastic model* [81, 269], where the job processing time is assumed to be a *random variable* with a known probability distribution. In practice, difficulties may still arise in some scenarios. First, one may not have enough prior information to characterize the probability distribution of a random processing time. Second, even if the probability distributions of all random processing times are a priori known, these distributions are really useful for a large number of realizations of similar scheduling environments, but they can be of little practical sense for a unique or small number of similar realizations.

In this book, a model of one of the more realistic scheduling scenarios is considered: It is assumed that in the realization of a schedule a processing time (or other numerical parameter) may take any real value between the *lower* and *upper bounds* given before scheduling. Obviously, a deterministic model is a special case of the model under consideration (namely, if lower and upper bounds are identically given for a processing time). The model considered can also be interpreted as a stochastic one under *strict uncertainty* when there is no sufficient a priori information about the probability distribution of a random processing time (or more precisely, it is only known that the random processing time will fall between the given lower and upper bounds with probability one). In spite of obvious practical importance, the model under such a strict uncertainty has attracted a limited attention in the OR literature so far.

The general scheme of our approach for dealing with uncertainty in scheduling may be described as follows. The whole scheduling problem is decomposed into two (or more) sequential scheduling problems (phases). At the off-line phase, a set of potentially optimal schedules has to be constructed under conditions of *uncertainty* of the given numerical input data. It is assumed that only lower and upper bounds for the activity duration are known at the first, off-line, phase of scheduling. Moreover, the probability distributions of the random activity durations are assumed to be unknown between their lower and upper bounds. For solving a scheduling problem with *uncertain* input data we propose to use a *stability* analysis of an optimal schedule with respect to possible variations of the given parameters. Since the "price" of *time* is not high at the off-line phase of scheduling, a decision-maker can use even a time-consuming algorithm for solving a scheduling problem with uncertain numerical data. When some activities will be realized, a decision-maker will have more reliable information which may be used to find an optimal schedule. So, at the on-line phase of scheduling, it is necessary to choose a schedule (from the set of potentially optimal sched-

ules constructed at the off-line phase) which has to be realized in an optimal way. Such a schedule has to be optimal for the actual activity durations. To solve a scheduling problem at the on-line phase, a decision-maker needs to use fast polynomial-time algorithms. Next, we formally introduce the main models.

We consider a multi-stage processing system (for brevity, a *shop*), which consists of a set of machines $M = \{M_1, M_2, \ldots, M_m\}$ that have to process a set of given jobs $J = \{J_1, J_2, \ldots, J_n\}$. For the shop under consideration, there are assumed five conditions, three of which are as follows.

Condition 1: *At any time, each machine $M_k \in M$ either processes one job from the set J or is idle.*

Condition 2: *At any time, each job $J_i \in J$ is either processed by one machine from set M, waits for processing or is already completed.*

Condition 3: *The machine order $(M_{i_1}, M_{i_2}, \ldots, M_{i_{n_i}})$ for processing each job $J_i \in J$, called the technological or machine route of job J_i, is fixed before scheduling.*

The processing of a job $J_i \in J$ by a machine $M_{i_k} \in M$ at the *stage* $k \in \{1, 2, \ldots, n_i\}$ of the technological route is called an *operation* denoted as O_{ik}. Let Q^J be the set of all operations for processing all jobs from the set J: $Q^J = \{O_{ik} : J_i \in J,\ k = 1, 2, \ldots, n_i\}$. Condition 3 means that processing of a job from the set J includes the processing of the given set of operations in the fixed order, provided that the distribution of the operations Q^J to the machines from the set M is also fixed via the machine routes of the jobs J.

If the technological routes are *identically* given for all jobs from set J, e.g., $(M_1, M_2, \ldots, M_m)$, then we have a **flow shop**, otherwise (if the technological routes may be given *differently* for different jobs), we have a **job shop**. In the former case, each job has to be processed once by each machine while in the latter case, both repetitions and absence of a machine in the technological route of a job are allowed. In both cases each operation is assigned to a certain machine, and the technological route $(M_{i_1}, M_{i_2}, \ldots, M_{i_{n_i}})$ of job $J_i \in J$ defines completely ordered operations (*a sequence*) $(O_{i1}, O_{i2}, \ldots, O_{in_i})$ such that operation O_{ik} has to be processed by machine $M_{i_k} \in M$ after operation $O_{i,k-1}$ processed by machine $M_{i_{k-1}} \in M$ and before operation $O_{i,k+1}$ processed by machine $M_{i_{k+1}} \in M$, $k \in \{2, 3, \ldots, n_i - 1\}$.

For a flow shop, equality $n_i = m$ holds for each job $J_i \in J$ while in the general case of a job shop, the value n_i may be smaller or larger than m or equal to m for a job J_i. For both flow shop and job shop problems, it is assumed that there are no other *precedence constraints* given a priory on the set of operations Q^J except those defined by the technological routes

$(O_{i1}, O_{i2}, \ldots, O_{in_i})$, $J_i \in J$.

Since a flow shop is a special case of a job shop, all algorithms developed for solving a job shop problem may be used for solving a flow shop problem as well. On the other hand, if a special case of a flow shop problem is proven to be NP-hard, the same special case of a job shop problem is NP-hard as well. The following condition also holds for the shop-scheduling problems considered in this book.

Condition 4: *Preemptions of an operation are forbidden.*

Condition 4 means that in any (*feasible*) schedule, operation $O_{ij} \in Q^J$ being started at time s_{ij} has to be processed up to its completion time $c_{ij} = s_{ij} + p_{ij}$, where p_{ij} denotes the processing time of operation O_{ij}.

Let Q_k^J denote the set of all operations from the set Q^J, $Q_k^J \subseteq Q^J$, which have to be processed by machine $M_k \in M$. In a *deterministic model*, the processing times p_{ij} are known in advance for all operations O_{ij}, $J_i \in J$, $j = 1, 2, \ldots, n_i$. Therefore, a *schedule* may be defined as the set of starting times s_{ij} (or completion times c_{ij}) of all operations Q^J provided that Conditions 1 – 4 hold. Such a set of starting (or completion) times of operations Q^J defines a unique *sequence* for processing the operations Q_k^J by each machine M_k, $k = 1, 2, \ldots, m$. Thus, a schedule uniquely defines m sequences (a unique sequence of the operations Q_k^J for each machine $M_k \in M$).

The objective of a scheduling problem is to find such a *schedule* (i.e., to find such m sequences of the operations Q_k^J on the machines M_k, $k = 1, 2, \ldots, m$) for which the value of the given objective function $\Phi(C_1, C_2, \ldots, C_n)$ is minimal. Hereafter, equality $C_i = c_{in_i}$ holds, and so C_i is equal to the completion time of job $J_i \in J$.

If the objective function $\Phi(C_1, C_2, \ldots, C_n)$ is a non-decreasing one, such a criterion is called *regular* [204]. In multi-stage scheduling, the most popular regular criteria are the minimization of maximum flow time (makespan)

$$\Phi(C_1, C_2, \ldots, C_n) = \max\{C_i : J_i \in J\} = \mathcal{C}_{max}$$

and the minimization of mean flow time

$$\Phi(C_1, C_2, \ldots, C_n) = \sum_{i=1}^{n} C_i = \sum \mathcal{C}_i.$$

Scheduling problems are usually classified by a triplet $\alpha/\beta/\gamma$. The α field describes the machine environment and usually contains a single entry (type of the processing system) or double entries (type of the system and number of machines m). The β field provides details of the processing characteristics and may contain no entries, a single entry, or multiple entries (number of

jobs, restrictions on the processing times, etc.). The γ field contains the objective function to be minimized, and it usually contains a single entry. Using such a three-field notation, the deterministic job shop problems considered in Chapter 1 are denoted by $\mathcal{J}//\mathcal{C}_{max}$ and $\mathcal{J}//\sum\mathcal{C}_i$ for the job shop and by $\mathcal{F}//\mathcal{C}_{max}$ and $\mathcal{F}//\sum\mathcal{C}_i$ for the flow shop. The symbols $\mathcal{J}$ and $\mathcal{F}$ are used to indicate a job shop and a flow shop, respectively.

The job shop problem is NP-hard for most criteria considered in scheduling theory even for a small number of machines and jobs. For example, the following shop-scheduling problems are unary NP-hard: $\mathcal{F}3//\mathcal{C}_{max}$, $\mathcal{F}3//\sum\mathcal{C}_i$, $\mathcal{J}3/p_{ij}=1/\mathcal{C}_{max}$ and $\mathcal{J}2/p_{ij}\in\{1,2\}/\mathcal{C}_{max}$ [126, 211, 212]. Problems $\mathcal{F}/n=3/\mathcal{C}_{max}$, $\mathcal{F}/n=3/\sum\mathcal{C}_i$, $\mathcal{J}3/n=2/\mathcal{C}_{max}$ and $\mathcal{J}3/n=2/\sum\mathcal{C}_i$ are binary NP-hard [53, 308, 326]. Therefore, to solve a job shop problem even with moderate numbers of machines and jobs one needs to develop complicated enumerative algorithms.

The job shop problem $\mathcal{J}//\Phi$ is a special case of a **general shop** problem $\mathcal{G}//\Phi$, in which *arbitrary* precedence constraints may be given on the set of operations. Hereafter, $\mathcal{G}$ indicates a general shop, and Φ denotes any given regular criterion: $\Phi=\Phi(C_1,C_2,\ldots,C_n)$. A general shop is defined via a partially ordered set of the given operations. For a general shop, the notion of a job may lose its sense, e.g., for the $\mathcal{C}_{max}$ criterion, and one can use a more simple notation for the operations as follows.

Let $Q=\{1,2,\ldots,q\}$ denote the set of all operations which have to be processed in a general shop, and Q_k denotes the set of all operations from the set Q, $Q_k\subseteq Q$, which have to be processed by machine $M_k\in M$. If $i\in Q_k$, then the non-negative real value p_i denotes the processing time of operation i by machine $M_k\in M$. Since a job shop is a special case of a general shop, we can use the above general shop notations for the job shop and flow shop as follows. Let Q^{J_i} denote all operations belonging to job $J_i\in J$. Using general shop notations, we assume that job J_1 has the operations

$$\{1,2,\ldots,n_1\}=Q^{J_1}, \tag{1}$$

job J_2 has the operations

$$\{n_1+1,n_1+2,\ldots,n_1+n_2\}=Q^{J_2}, \tag{2}$$

and so on, job J_n has the operations

$$\left\{\sum_{j=1}^{n-1}n_j+1,\sum_{j=1}^{n-1}n_j+2,\ldots,\sum_{j=1}^{n}n_j=q\right\}=Q^{J_n}. \tag{3}$$

After such an enumeration of the operations $Q^J = \{O_{ij},\ J_i \in J,\ j = 1, 2, \ldots, n_i\}$, we obtain a one-to-one correspondence between the operations of the sets Q^J and Q (operations of the sets Q_k^J and Q_k). E.g., a completely ordered set of the operations of job J_1 is denoted as $(O_{1,1}, O_{1,2}, \ldots, O_{1n_1})$ using the job shop notations, and it is denoted as $(1, 2, \ldots, n_1)$ using the general shop notations. Whenever it is possible, we shall use the general shop notations (since it is more simple), otherwise, we shall use the job shop notations. Such a principle of "rational sufficiency" is used within the whole book. Some adjectives which may be omitted are given in parentheses, e.g., (undirected) edge in a graph and (directed) arc in a digraph.

Note that the notion of a general shop is used rather seldom in modern scheduling theory (indeed, a general shop may be considered as a job shop with additional precedence constraints given on the set of operations Q^J and assuming that a job $J_k \in J$ may consist of only one operation $\{O_{1,1}\}$). However, we distinguish a general shop problem from a job shop problem since the former is the main subject investigated in Chapters 1 and 2 of this book. (A two-machine job shop and a flow shop are considered in Chapter 3.) Note also that the models and methods used for solving a general shop problem $\mathcal{G}//\Phi$ are close to those used for solving the more general **resource constrained project scheduling problem** (for brevity, RCPSP) widely studied in scheduling theory and widely used in practice. Reviews for the RCPSP can be found in [52, 90, 127, 158, 159, 160, 162]. The generalization of the RCPSP with respect to a general shop problem is connected with the assumptions that in the resource constrained project scheduling problem an operation may require some amount of one or more of the resources during its duration of execution, and each resource has a maximum capacity expressing the total amount of the resource that can be used at any time point by any set of operations. The most common scheduling model considered in this book is the general shop, however, some results presented in Chapters 1 and 2 and the approach to deal with uncertainty in scheduling presented in Chapters 1 – 4 may be applied to the RCPSP with regular criteria based on the objective function of the job completion times and the numbers of resources used for processing the given activities. It should be noted that in resource constrained project scheduling non-regular criteria were also considered (see, e.g., [28, 102, 256, 373]).

The usual assumption, which is used in most theoretical settings of scheduling problems, that durations of all operations are given in advance (before scheduling) and cannot change during the execution of a schedule restricts the usefulness of scheduling theory for practice. In Chapter 1,

we present results for the calculation of the stability radius of an optimal schedule for general and job shops. The *stability radius* denotes the largest quantity of independent variations of the operation durations such that the given schedule remains optimal. The main attention is paid to the results on a stability analysis which are used further in Chapter 2. Some other approaches to stability analysis and related results are discussed in the last section of Chapter 1.

Chapters 2 and 3 deal with a mathematical model for scheduling scenarios in which the processing time of each operation $i \in Q$ may be *uncertain* before applying a scheduling procedure and may take any value between a given *lower bound* $a_i \geq 0$ and an *upper bound* $b_i \geq a_i$. More precisely, in these three chapters we consider a general shop (job shop and flow shop as its special cases) when the *structural input data* are fixed while only a lower bound $a_i \geq 0$ and an upper bound $b_i \geq a_i$ for the processing time of operation $i \in Q$ are given as *numerical input data* before applying a scheduling procedure, i.e., the following condition holds.

Condition 5: *The actual processing time p_i of operation $i \in Q$ may take any real value between given lower and upper bounds:*

$$a_i \leq p_i \leq b_i, \ i \in Q. \tag{4}$$

It should be noted that, while Conditions 1, 2, 3 and 4 are commonly used in scheduling theory, Condition 5 is rather new. The main aim of this book is to introduce Condition 5 into relevant settings of scheduling problems. A general shop problem which satisfies Conditions 1 – 5 will be denoted by $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$. On the one hand, problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$ can be considered as a *stochastic* general shop problem under strict uncertainty, when there is no prior information about the probability distributions of the random processing times of operation $i \in Q$ in the segment $[a_i, b_i]$. On the other hand, if $a_i = b_i$ for each operation $i \in Q$, problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$ turns out to be a *deterministic* general shop problem $\mathcal{G}//\Phi$.

It is clear that not only the duration of the processing of operations may be a source of uncertainty in a practical scheduling. Uncertainty may also arise from machine breakdowns, unexpected arrivals of new jobs with high priorities, late arrivals of raw materials, modifications of setup times, release dates and due dates. As it will be shown in Chapter 1, all these and some other sources of uncertainty may be treated in terms of problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$.

Problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$ seems to be rather realistic, at least, it is not restrictive: Even if there is no prior information on the possible perturbations

of the processing times p_i, one can consider 0 as a lower bound of p_i and a sufficiently large number (e.g., the *planning horizon*) as an upper bound for p_i. Note that for a flow shop problem, fixing the structural input data means only to fix the number n of jobs and the number m of machines. Consequently, any two flow shop problems with the same number n of jobs and the same number m of machines, i.e., problems $\mathcal{F}m/n=k/\Phi$ may differ one from another only in their processing times. Since problem $\mathcal{G}//\Phi$ is a special case of problem $\mathcal{G}/a_i \le p_i \le b_i/\Phi$, NP-hardness of problem $\mathcal{G}//\Phi$ implies NP-hardness of problem $\mathcal{G}/a_i \le p_i \le b_i/\Phi$.

The material of the book is presented in an order from hard scheduling problems to easier scheduling problems. Chapters 1 and 2 deal with general and job shop problems and Chapter 3 with two-machine flow and job shop problems. Such an order of problems allows us to present mathematical results in the opposite order: from evident and simple results to more complicated and deep ones.

We tested algorithms coded in Fortran-77 for a stability analysis, in which an optimal schedule has already been constructed and the question is to determine such maximal variations of the operation processing times, which do not destroy schedule optimality. These computational results are presented in Chapter 1. In Chapter 2, we present computational results for solving randomly generated problems $\mathcal{J}/a_i \le p_i \le b_i/\mathcal{C}_{max}$ and $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$ using algorithms coded in Fortran-77. Chapter 3 contains computational results for solving randomly generated problems $\mathcal{F}2/a_i \le p_i \le b_i/\mathcal{C}_{max}$ and randomly generated two-machine minimum-length flow shop and job shop scheduling problems with limited machine availability. The algorithms developed in Chapter 3 have been coded in C++.

For the convenience of the reader, common notations are summarized in Table 2.22 at the end of Chapter 2 (page 178). Along with common notations, we summarize general shop notations in Table 1.5 at the end of Chapter 1 (page 80). Notations for two-machine flow and job shops are given in Table 3.41 at the end of Chapter 3 (page 288). Each chapter is written mainly as an independent one from the others. In the last section of each chapter and in the conclusion of the book, we summarize the known results with references and outline some topics for future research.

Chapter 1

Stability Radius of an Optimal Schedule

I am a believer in punctuality
though it makes me very lonely
E.V. Lucas

The usual assumption that the processing times of all operations (setup times, release dates, due dates, and other numerical parameters) are exactly known before scheduling restricts practical aspects of modern scheduling theory since it is often not valid for real-world processes. This chapter is devoted to some results obtained for the stability analysis of an optimal schedule which may help to extend the significance of scheduling theory for real production scheduling problems. The terms "stability analysis", "sensitivity analysis" or "post-optimal analysis" are used for the phase of an algorithm at which a solution of an optimization problem has already been found, and additional calculations are performed in order to investigate how this solution depends on the numerical input data.

In this chapter, we prove some results on job and general shop scheduling problems for the calculation of the stability radius of an optimal schedule, when the objective is to minimize mean or maximum flow time or, in general, to minimize any given non-decreasing function of the job completion times. The stability radius of an optimal schedule denotes the largest quantity of independent variations of the processing times of the operations (and other changeable numerical parameters) such that this schedule remains optimal. The extreme values of the stability radius are considered more in detail.

The results presented in this chapter may be considered as a post-optimal analysis of scheduling problems with uncertain numerical data when the aim

is to study the influence of round-off errors or changes of the processing times (and other numerical parameters) on the property of a schedule to be optimal. The main reason for performing such a stability analysis is that in most practical cases the processing times of the operations (and other numerical input data) are inexact or uncertain before applying a scheduling procedure. In such cases, a stability analysis is necessary to investigate the credibility of an optimal schedule at hand.

On the one hand, if possible errors of the numerical parameters are larger than the stability radius of an optimal schedule, this schedule may not be the best in a practical realization, and there is not much sense in likely large efforts to construct an optimal schedule: It may be more advisable to restrict the scheduling procedure to the construction of an approximate or heuristic solution. On the other hand, this is not the case when the possible change of each numerical parameter is less than or equal to the stability radius of an optimal schedule: An a priori constructed optimal schedule will remain optimal (the best for the given objective) in the practical realization.

Another reason for calculating the stability radius is connected with the need to solve a set of similar scheduling problems. In reality, the main characteristics of a shop (such as the number of machines, the technological routes, the range of variations of the processing times and so on) do not change quickly, and it may be possible to use previous computations of an optimal schedule for solving a new similar scheduling problem. Since the majority of scheduling problems is NP-hard, enumeration schemes such as branch-and-bound are often used for finding an optimal schedule. To this end, it is necessary to construct a solution tree, which is often huge. Unfortunately, most of the information contained in the solution tree is lost after having solved the problem. In such a situation, the stability radius of the optimal schedule constructed gives the possibility to use a part of this information for solving further similar scheduling problems.

The rest of this chapter is organized as follows. In Section 1.1, we introduce a mixed (disjunctive) graph model to represent the input data of a general shop scheduling problem $\mathcal{G}//\Phi$. It is shown that various scheduling problems may be represented as extremal problems on mixed graphs, and the only requirement for such a representation is the prohibition of operation preemptions (Condition 4 on page 12). In Section 1.2, we describe how the stability radius of an optimal schedule (digraph) for problem $\mathcal{G}//\Phi$ can be calculated via the reduction to a non-linear mathematical programming problem. The advantage of studying the stability of an optimal digraph instead of the stability of an optimal schedule is demonstrated in Section 1.1

and in Section 1.2. The calculation of the stability radius along with characterizations of its extreme values for problems $\mathcal{G}//\mathcal{C}_{max}$ and $\mathcal{G}//\sum\mathcal{C}_i$ are considered in Section 1.3 and in Section 1.4, respectively. Algorithms for calculating the stability radius are given in Section 1.5. Section 1.6 contains computational results for the calculation of the stability radius of optimal schedules for randomly generated job shop scheduling problems. In Section 1.7, we survey different approaches to stability analysis in combinatorial optimization.

1.1. Mixed Graphs for Modelling a General Shop

We consider a general shop in which the given partially ordered set of operations $Q = \{1, 2, \ldots, q\}$ has to be processed by a set of machines $M = \{M_1, M_2, \ldots, M_m\}$. It is assumed that each operation $j \in Q$ is assigned to one machine from the set M, and at any time each machine can process at most one operation (Condition 1 on page 11).

Let p_j denote the processing time (duration) of operation $j \in Q$ and c_j denote the completion time of operation j. Operation preemptions are not allowed (Condition 4 on page 12): If an operation j starts at time s_j, its processing is not interrupted until operation j is completed (up to time $c_j = s_j + p_j$). The problem of finding an optimal schedule minimizing the given objective function Φ of the job completion times is denoted as $\mathcal{G}//\Phi$. In this book, only regular criteria are considered, i.e., it is assumed that the objective function $\Phi(C_1, C_2, \ldots, C_n)$ is non-decreasing, where C_i denotes the completion time of job J_i.

The set of operations Q is partially ordered by the given precedence constraints (temporal constraints) $\rightarrow$. Given two operations $i \in Q$ and $j \in Q$, the notation $i \rightarrow j$ means that operation i is a predecessor of operation j. In other words, if $i \rightarrow j$, then inequality

$$c_i + p_j \leq c_j \tag{1.1}$$

must hold for any feasible schedule. Let Q_k be the set of operations processed by machine $M_k \in M$ and $\{Q_k : \ k = 1, \ldots, m\}$ be a partition of the set Q:

$$Q = \bigcup_{k=1}^{m} Q_k, \ Q_k \neq \emptyset, \ Q_k \bigcap Q_l = \emptyset, \ \text{ if } \ k \neq l, \ k = 1, \ldots, m, \ l = 1, \ldots, m.$$

We say that the above partition defines capacity constraints (resource constraints). Since at any time each machine $M_k \in M$ can process at most one

operation (Condition 1) and operation preemptions are not allowed (Condition 4), the two inclusions $i \in Q_k$ and $j \in Q_k$ imply one of the following inequalities:

$$c_i + p_j \leq c_j \quad \text{or} \quad c_j + p_i \leq c_i. \tag{1.2}$$

For the case of a job shop problem $\mathcal{J}//\Phi$, along with the above partition, the set of operations Q is also partitioned into n chains (linearly ordered sets):

$$Q = \bigcup_{i=1}^{n} Q^{J_i},\ Q^{J_i} \neq \emptyset,\ Q^{J_i} \bigcap Q^{J_j} = \emptyset, \text{ if } i \neq j,\ i = 1, \ldots, n,\ j = 1, \ldots, n, \tag{1.3}$$

where each chain includes the set Q^{J_i} of operations for processing job J_i, $1 \leq i \leq n$. This chain represents the (technological) route of job J_i. For a job shop, all ordered sets Q^{J_i} are known in advance (before scheduling).

For problem $\mathcal{G}//\Phi$, the processing time p_i of each operation $i \in Q$ is also fixed before scheduling, and therefore, a schedule of the operations Q on the machines M may be defined by the completion times c_i or by the starting times $s_i = c_i - p_i$ of all operations $i \in Q$.

If the operation processing times are not fixed before scheduling (Condition 5 on page 15), it is not possible to define s_i and c_i for all operations $i \in Q$. Therefore, in the general case of problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$, the goal is to determine a sequence for processing the set of operations Q_k on each machine $M_k \in M = \{M_1, M_2, \ldots, M_m\}$. Such a set of m sequences satisfying both the given precedence constraints (1.1) and the given capacity constraints (1.2) may be defined as a schedule for problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$. The general shop problem is to find such a schedule, which minimizes the value of the given non-decreasing objective function $\Phi(C_1, C_2, \ldots, C_n)$.

A mixed (disjunctive) graph is often introduced to model a deterministic scheduling problem. We follow this approach and represent the structural input data for a general shop problem by means of a mixed graph $G = (Q, A, E)$, where

- the set Q of operations is the set of vertices;
- the precedence constraints (1.1) are represented by the set of non-transitive (directed) arcs A: If operation i has to be processed before operation j starts, i.e., precedence constraint $i \to k$ holds and there is no other operation k such that $i \to k$ and $k \to j$ hold, then arc (i, j) has to belong to set A:

$$A = \{(i, j) : i \to j,\ i, j \in Q, \text{ there is no } k \in Q \text{ such that } i \to k \to j\};$$

- the capacity constraints (1.2) are represented by the set E of (undirected) edges $[i,j]$ connecting the unordered operations i and j, which have to be processed by the same machine:

$$E = \{[i,j] \ : \ i \in Q_k, \ j \in Q_k, \quad k = 1, \ldots, m, \quad i \not\to j, \ j \not\to i\}.$$

For a deterministic problem $\mathcal{G}//\Phi$, the processing times p_i of all operations $i \in Q$ are known before scheduling, and we can associate a non-negative weight p_i with each vertex $i \in Q$ in the mixed graph $G = (Q, A, E)$ constructed for problem $\mathcal{G}//\Phi$. As a result, we obtain the weighted mixed graph $G(p) = (Q(p), A, E)$ with $p = (p_1, p_2, \ldots, p_q)$, which represents both the structural and numerical input data for the general shop scheduling problem $\mathcal{G}//\Phi$.

For solving problem $\mathcal{G}//\Phi$ using the mixed graph G, it is necessary to replace the edge $[i,j] \in E$ by an arc incident to the same vertices i and j. Indeed, due to Condition 1 (given on page 11), if edge $[i,j]$ belongs to set E, then for the pair of operations i and j there exist two possibilities: To complete operation $i \in Q_k$ before operation $j \in Q_k$ starts on their common machine $M_k \in M$ and to provide the first inequality from (1.2) (in this case the edge $[i,j]$ has to be replaced by the arc (i,j)), or to complete operation j before operation i starts and to provide the second inequality from (1.2) (in this case the edge $[i,j]$ has to be replaced by the arc (j,i)).

Instead of a mixed graph $G = (Q, A, E)$, often a disjunctive graph is used with the same purpose. Let

$$E^* = \bigcup_{[i,j] \in E} \{(i,j), (j,i)\}.$$

The term "disjunctive graph" is associated with the choice of one of the above two possibilities for each pair of arcs $\{(i,j), (j,i)\} \subseteq E^*$ called disjunctive arcs. This means that for solving problem $\mathcal{G}//\Phi$ using the disjunctive graph (Q, A, E^*), one of these arcs must be added to a subset $E_s \subset E^*$ of chosen arcs, while the other one must be rejected from the disjunctive graph (Q, A, E^*):

(*) Arc (i,j) belongs to set E_s if and only if $(j,i) \in E^* \backslash E_s$.

In terms of a mixed graph $G = (Q, A, E)$, the above choice of arc (i,j) from the set E^* is equivalent to the replacement of the edge $[i,j] \in E$ by the arc (i,j). If such a replacement (choice, respectively) is done for each edge $[i,j] \in E$ (for each pair of disjunctive arcs $\{(i,j), (j,i)\} \subset E^*$), then we obtain the same set of arcs E_s. As a result, the mixed graph $G = (Q, A, E)$

(disjunctive graph (Q, A, E^*)) is transformed into the same resulting digraph $G_s = (Q, A \cup E_s, \emptyset)$.

It is clear that not each of such subsets E_s may be feasible for constructing a feasible schedule for problem $\mathcal{G}//\Phi$ since the set of chosen arcs E_s may cause a contradiction, i.e., the arc set $A \cup E_s$ may imply that 'some operations have to be started before they are completed'. When an arc (i, j) is chosen, all disjunctive arcs that are made redundant due to the transitive and anti-reflexive nature of the precedence constraints, should be excluded. It is easy to convince that a feasible schedule s is defined by a subset $E_s \subset E^*$ such that along with the above condition (*), the following condition (**) has to be satisfied:

(**) Digraph $G_s = (Q, A \cup E_s, \emptyset)$ contains no circuit.

In what follows, we mainly use the terminology based on a mixed graph modelling. The digraph $G_s = (Q, A \cup E_s, \emptyset)$ generated from the mixed graph $G = (Q, A, E)$ by an orientation of all edges of the set E is called *feasible* if and only if condition (**) holds: G_s contains no circuit.

Let $\Lambda(G) = \{G_1, G_2, \ldots, G_\lambda\}$ be set of all feasible digraphs $G_s = (Q, A \cup E_s, \emptyset)$. In other words, a digraph $G_s = (Q, A \cup E_s, \emptyset)$ is generated from the disjunctive graph $G = (Q, A, E^*)$ and satisfies both conditions (*) and (**). We need the following definition of semiactive schedules.

Definition 1.1 *A schedule is called semiactive if no operation $i \in Q$ can start earlier without delaying the processing of some other operation from the set Q or without altering the processing sequence of the operations on any of the machines M.*

For solving problem $\mathcal{G}//\Phi$, we can restrict ourselves to the consideration of the set S of all semiactive schedules only since the following claim holds.

Lemma 1.1 *If the objective function $\Phi(C_1, C_2, \ldots, C_n)$ is non-decreasing, it is sufficient to check only semiactive schedules while solving problem $\mathcal{G}//\Phi$.*

Proof. Let schedule s be optimal for problem $\mathcal{G}//\Phi$. If the schedule s belongs to set S, we are done.

Otherwise, we can construct a semiactive schedule $s' \in S$ from schedule s by starting each operation $i \in Q$ as early as possible without delaying the processing of other operations from the set Q and without altering the processing sequence of the operations on any of the machines M. Since operation preemptions are forbidden (see Condition 4 on page 12), the completion time of each operation $i \in Q$ in schedule s' is less than or equal to

that in the initial schedule s. Since the objective function $\Phi(C_1, C_2, \ldots, C_n)$ is non-decreasing, the optimality of schedule s implies the optimality of the semiactive schedule s'.

$\diamond$

Next, we show that there exists a one-to-one correspondence between all semiactive schedules $S = \{1, 2, \ldots, \lambda\}$ constructed for problem $\mathcal{G}//\Phi$ and all feasible (i.e., circuit-free) digraphs $\Lambda(G) = \{G_1, G_2, \ldots, G_\lambda\}$ generated from the mixed graph G by orienting all edges from the set E.

Lemma 1.2 *Each feasible digraph $G_s = (Q, A \cup E_s, \emptyset) \in \Lambda(G)$ uniquely defines a semiactive schedule $s \in S$ for problem $\mathcal{G}//\Phi$, and vice versa.*

PROOF. On the one hand, given a vector $p = (p_1, p_2, \ldots, p_q)$ of the operation processing times, a feasible digraph $G_s = (Q, A \cup E_s, \emptyset)$, $G_s \in \Lambda(G)$, defines the weighted digraph $G_s(p) = (Q(p), A \cup E_s, \emptyset)$ which uniquely defines the *earliest completion time* $c_i(s)$ of each operation $i \in Q$. Consequently, the weighted digraph $G_s(p)$ defines the unique semiactive schedule

$$s = (c_1(s), c_2(s), \ldots, c_q(s)).$$

On the other hand, each semiactive schedule $s \in S$ defines m sequences of the operations Q_k on the machines M_k, $k = 1, 2, \ldots, m$. These m sequences define a unique digraph $G_s(p) \in \Lambda(G)$.

$\diamond$

Given a vector $p = (p_1, p_2, \ldots, p_q)$, a digraph $G_s \in \Lambda(G)$ and a weighted digraph $G_s(p)$ will be called optimal if and only if schedule $s \in S$ is optimal. Due to Lemma 1.2, we can use a digraph $G_s \in \Lambda(G)$ (an optimal digraph G_s) instead of a schedule $s \in S$ (instead of an optimal schedule s). The digraph $G_s \in \Lambda(G)$ uniquely defines a set of m sequences for processing the operations Q_k by machine $M_k \in M = \{M_1, M_2, \ldots, M_m\}$, and vice versa.

It should be noted that digraph $G_s \in \Lambda(G)$ is more appropriate for solving problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$ and when performing a stability analysis for problem $\mathcal{G}//\Phi$ since it is 'more stable' than the corresponding schedule $s \in S$ with respect to variations of the operation durations (and other numerical parameters involved in the schedule s). It is more useful to consider a stability analysis for an optimal digraph $G_s \in \Lambda(G)$ (defining the set of m optimal sequences for processing the operations Q_k by machine $M_k \in \{M_1, M_2, \ldots, M_m\}$) than for the optimal schedule $s \in S$. Frequently, a production process is controlled by the operation sequences and not the

actual starting and completion times of the operations provided by a schedule. Also, even a small change in a processing time changes an optimal schedule (and other schedules). Therefore, the practical need to reschedule occurs only when the operation sequence changes. Large and more meaningful ranges of numerical parameter changes are obtained when considering the digraph $G_s \in \Lambda(G)$ instead of schedule $s \in S$. Note that the starting and completion times of the operations Q, the value of the objective function and other characteristics of a semiactive schedule s, corresponding to an acyclic weighted digraph $G_s(p)$, can be easily determined using *longest path calculations*.

Given a fixed vector $p = (p_1, p_2, \ldots, p_q)$ of the operation processing times, in order to construct an optimal schedule for problem $\mathcal{G}//\Phi$ using the mixed graph model, one may enumerate (explicitly or implicitly) feasible digraphs $G_1(p), G_2(p), \ldots, G_\lambda(p)$ generated by orienting all edges of the mixed graph G and then select an optimal digraph, i.e., a feasible digraph with minimal value of the objective function. Unfortunately, the number λ of such feasible digraphs (i.e., the number of semiactive schedules) grows exponentially in the number of edges $|E|$, and an overall enumeration of feasible digraphs is practically impossible for large numbers of jobs and machines. Of course, the cardinality of the set $\Lambda(G)$ of feasible (circuit-free) digraphs is less than $2^{|E|}$, but a procedure for testing whether a digraph is circuit-free or not may take also running time. Nevertheless, for some computational experiments, we use an explicit enumeration of the feasible digraphs for rather small job shop problems in order to calculate the stability radii for all the optimal schedules.

Although problem $\mathcal{G}//\Phi$ is unary NP-hard for any given regular criterion Φ considered in modern scheduling theory, the running time of calculating an optimal schedule $s = (c_1(s), c_2(s), \ldots, c_q(s))$ may be restricted by an $O(q^2)$-algorithm after having constructed an optimal digraph $G_s(p)$. Thus, the main difficulty of problem $\mathcal{G}//\Phi$ (in terms of the mixed graph approach) is to construct an optimal digraph $G_s = (Q, A \cup E_s, \emptyset)$, i.e., to find the best set E_s of arcs generated by orienting the edges of the set E. Due to the particular importance of the set E_s, it is called a *signature* of schedule s. Each feasible digraph $G_s = (Q, A \cup E_s, \emptyset)$ is uniquely defined by its signature, i.e., by the set of arcs E_s which replaces the set of edges E.

As it was noted in [35], the mixed graph model "has mostly replaced the solution representation by Gantt charts as described in [124]". We can give the following comments to elaborate this kind of preference. First, while a Gantt chart is useful for the graphical presentation of a particular solution,

the mixed graph model is suitable for the whole scheduling process from the initial mixed graph $G(p)$ (representing the input data) until a final digraph G_s (representing a schedule $s \in S$) has been found. Second, a Gantt chart is a representation of one particular situation when there are no changes both in the a priori known processing times (and other numerical input data) and in the calculated starting times and completion times of all the operations Q. However, such a situation is 'ideal' (at least, it occurs rather seldom in real life). Thus, a Gantt chart seems to be more appropriate 'after realization' of the process (when all the processing times, starting times and completion times are known exactly) while 'before realization' a mixed graph $G(p)$ and a digraph G_s seem to be more useful since they are stable with respect to possible changes of the above 'times'. Third, while a Gantt chart is simply a picture in the plane, a digraph is a mathematical (i.e., abstract) object and can assume different graphical presentations. In particular, one can view a Gantt chart as a diagram of the weighted digraph $G_s(p)$ in the plane.

Next, we will show how a mixed graph model may be introduced in the case of a job shop problem $\mathcal{J}//\Phi$ when an operation of a job $J_i \in J$ is given as O_{ij}. For a more convenient notation for the job shop, we will often use a double subscript designated to operations.

Mixed Graph for a Job Shop Problem

To present the structural input data for problem $\mathcal{J}//\Phi$, one can use a mixed graph (Q^J, A^J, E^J) with the following sets of vertices, arcs and edges:

- $Q^J = \{O_{ij} \,:\, J_i \in J,\ j = 1, 2, \ldots, n_i\}$;
- $A^J = \{(O_{ij}, O_{i,j+1}) \,:\, J_i \in J,\ j = 1, 2, \ldots, n_i - 1\}$;
- $E^J = \{[O_{ij}, O_{uv}] \,:\, O_{ij} \in Q_k,\ O_{uv} \in Q_k,\ J_i \neq J_u\}$.

The set of arcs A^J defines *precedence constraints* (technological routes) as follows. Since each job $J_i \in J$ may be processed by at most one machine from the set M at a time (Condition 2 on page 11) and the technological route is fixed for each job (Condition 3 on page 11), operation O_{ij} has to be completed before operation $O_{i,j+1}$ $(1 \leq j \leq n_i - 1)$ starts: $c_{ij} \leq s_{i,j+1}$. The machine route of job $J_i \in J$ defines linearly ordered operations (a sequence) $(O_{i1}, O_{i2}, \ldots, O_{in_i})$. At the stage $k \in \{1, 2, \ldots, n_i\}$ of the machine route of job J_i, operation O_{ik} has to be processed by machine $M_{i_k} \in M$.

The set of edges E^J defines *capacity constraints* as follows. The set Q_k^J is the set of operations which has to be processed by machine $M_k \in M$. Since

any machine $M_k \in M$ can process at most one operation at a time (Condition 1 on page 11) and operation preemptions are not allowed (Condition 4 on page 12), operation $O_{ij} \in Q_k^J$ has to precede operation $O_{uv} \in Q_k^J$ or vice versa: $c_{ij} \leq s_{uv}$ or $c_{uv} \leq s_{ij}$.

Since a job shop is a special case of a general shop, one can use the notations of a general shop for the job shop as well, assuming that job J_1 consists of the set of operations Q^{J_1} defined in (1), job J_2 of the operations Q^{J_2} defined in (2), and so on, job J_n of the operations Q^{J_n} defined in (3).

It is often more convenient to use the following notation for the above enumeration of the operations. Let

$$w(i) = \sum_{k=0}^{i-1} n_k \tag{1.4}$$

denote the last operation of job J_{k-1} with $k \in \{2, 3, \ldots, n+1\}$ and let $n_0 = 0$. Then job J_1 consists of the linearly ordered operations $Q^{J_1} = (w(1)+1, w(1)+2, \ldots, w(1)+n_1)$, job J_2 of the linearly ordered operations $Q^{J_2} = (w(2)+1, w(2)+2, \ldots, w(2)+n_2)$, and so on, job J_n of the linearly ordered operations $Q^{J_n} = (w(n)+1, w(n)+2, \ldots, w(n)+n_n)$. Due to this enumeration of the operations for problem $\mathcal{J}//\Phi$, one can use the mixed graph $G = (Q, A, E)$ with $Q = \cup_{i=1}^{n} Q^{J_i}$ for modelling a job shop similarly as a general shop.

If operation processing times (or other numerical parameters), which are given before applying a scheduling procedure, may vary in the realization of a schedule, it is not sufficient to construct only an optimal digraph $G_s \in \Lambda(G)$ for solving problem $\mathcal{G}//\Phi$. It is also important to analyze the question of how much the processing times (durations) of the operations (and other numerical parameters) may vary so that the digraph G_s remains optimal.

In the rest of this chapter, we study the following stability analysis question. What are the limits to processing time changes such that the schedule at hand remains optimal? Of course, other numerical parameters of a practical scheduling problem may be also changeable. Next, we show that due to the generality of the above mixed graph model with any given regular objective function, one can analyze other changeable parameters of a general shop scheduling problem (like *release dates, due dates, job weights, setup times* and *removal times*, etc) in terms of the mixed graph model G.

In particular, including in the mixed graph G a dummy operation 0 which proceeds the first operation $w(i)+1$ of a job $J_i \in J$, i.e., $0 \in Q$ and $(0, w(i)+1) \in A$, allows one to consider the *processing time* p_0 of this dummy operation as a *release date* of job J_i. The number $w(i)$ is defined in

(1.4). Let d be a sufficiently large number. Including in the mixed graph G a dummy operation $q+1$ which succeeds the last operation $w(i)+n_i$ of job J_i, i.e., $(q+1) \in Q$ and $(w(i)+n_i,\ q+1) \in A$, allows one to consider the *processing time* $p_{q+1} = d - d_i$ of the dummy operation $q+1$ as a *due date* d_i of job J_i.

If machine $M_k \in M$ needs a sequence-independent setup time $\tau_i \geq 0$ before starting operation $j \in Q_k$ after the completion of operation $i \in Q_k$, $[i,j] \in E$, then the weight (processing time) of operation i has to be increased by the value τ_i ($p_i := p_i + \tau_i$). If job $J_i \in J$ needs a sequence-independent transportation time $\vartheta_i \geq 0$ before starting operation $(i+1) \in Q^{J_i}$ after the completion of operation $i \in Q^{J_i}$, $(i, i+1) \in A$, then the weight (processing time) of operation i has to be increased by the value ϑ_i ($p_i := p_i + \vartheta_i$).

If the given setup times $\tau_{ij} \geq 0$ (or transportation times $\vartheta_{ij} \geq 0$) are sequence-dependent, then we have to consider also weighted arcs and edges in the mixed graph G. Namely, the arc $(i,j) \in A$ must have the weight τ_{ij} (weight ϑ_{ij} respectively). The edge $[i,j] \in E$ must have two weights: the weight τ_{ij} (weight ϑ_{ij}) for a possible orientation $(i,j) \in E_s$ of edge $[i,j]$, and the weight τ_{ji} (weight ϑ_{ji}) for a possible orientation $(j,i) \in E_s$ of this edge. Job weights may be taken into account in the given objective function Φ.

Remark 1.1 The mixed graph model G allows one to consider other parameters (like *release dates, due dates, setup times* and *transportation times*, etc.) than the processing times (weights) of the operations (vertices) from set Q or those of dummy operations included in the set Q.

In the following sections of this chapter, we present some results for the stability ball of an optimal digraph $G_s(p)$, i.e., a closed ball in the space of the numerical input data such that within this ball a schedule s remains optimal. For simplicity, due to Remark 1.1, we will continue to consider operation durations (processing times) as the only changeable parameters. The next section contains a formal definition of the stability radius, which is the maximal value of the radius of such a stability ball.

1.2. Regular Criterion

In the rest of this chapter the main question is as follows. How can one vary the processing times $p_i,\ i \in Q$, simultaneously such that a given schedule $s \in S$, which is optimal for problem $\mathcal{G}//\Phi$ with the processing times $p_i,\ i \in Q$, remains optimal for the new processing times, and how can one calculate the largest quantity of such simultaneous and independent

variations of the processing times?

Any variation $p_i \pm \epsilon, \epsilon > 0$, of a processing time p_i in problem $\mathcal{G}//\Phi$ implies the change of at least one completion time $c_i(s)$ of an operation $i \in Q$ in any feasible semiactive schedule and also in an optimal semiactive schedule $s = (c_1(s), \ldots, c_i(s), \ldots, c_q(s)) \in S$. As a result, the schedule s has to be transformed into another schedule: $(\ldots, c_i(s) + \epsilon, \ldots)$ or $(\ldots, c_i(s) - \epsilon, \ldots)$ in order to be feasible. Fortunately, the optimal digraph $G_s = (Q, A \cup E_s, \emptyset)$ for the new problem $\mathcal{G}//\Phi$ obtained due to such a variation of the processing time p_i may often remain the same if ϵ is sufficiently small, since the signature E_s of an optimal schedule s is more stable with respect to a variation $p_i \pm \epsilon, \epsilon > 0$, of processing time p_i than the optimal semiactive schedule $s = (c_1(s), \ldots, c_i(s), \ldots, c_q(s))$. Note that it is often more important in practice to keep in mind not the calendar times when the operations have to be started and have to be completed, but only m sequences in which the operations $Q_k, k = 1, 2, \ldots, m$, have to be processed on each machine $M_k \in M$ (these sequences are uniquely defined by digraph G_s).

This section is devoted to the stability of an optimal digraph $G_s(p)$ which represents a solution to problem $\mathcal{G}//\Phi$. The above question may be concretized as follows. Under which largest simultaneous and independent changes in the components of the vector $p = (p_1, p_2, \ldots, p_q)$ of the operation processing times remains digraph $G_s(p)$ optimal?

Next, we introduce these notions in a formal way. Let R^q be the space of all q-dimensional real vectors p with the maximum (Chebyshev) metric, i.e., the distance $d(p, p')$ between the vectors $p \in R^q$ and $p' = (p'_1, p'_2, \ldots, p'_q) \in R^q$ is defined as follows:

$$d(p, p') = \max_{i \in Q} |p_i - p'_i|, \tag{1.5}$$

where $|p_i - p'_i|$ denotes the absolute value of the difference $p_i - p'_i$.

Let R^q_+ be the space of all q-dimensional non-negative real vectors:

$$R^q_+ = \{x = (x_1, x_2, \ldots, x_q) : \ x_i \geq 0, \ i \in Q\},$$

and schedule $s \in S$ be optimal for problem $\mathcal{G}//\Phi$ with the non-negative real vector $p \in R^q_+ \subset R^q$ of the operation processing times.

Definition 1.2 *The closed ball $O_\varrho(p)$ with the radius $\varrho \in R^1_+$ and the center $p \in R^q_+$ in the space of q-dimensional real vectors R^q is called a stability ball of schedule $s \in S$ (of digraph $G_s \in \Lambda(G)$) if for any vector $p' \in O_\varrho(p) \cap R^q_+$ of the processing times, schedule s (digraph $G_s(p')$) remains optimal. The maximum value $\varrho_s(p)$ of such a radius ϱ of a stability ball $O_\varrho(p)$ of schedule*

s (of digraph G_s) is called the stability radius of schedule s (of digraph G_s):

$$\varrho_s(p) = \max\{\varrho \in R^1_+ : \ \textit{If } p' \in O_\varrho(p) \cap R^q_+, \ \textit{digraph } G_s \textit{ is optimal}\}. \quad (1.6)$$

We denote the stability radius by $\varrho_s(p)$ for an arbitrarily given regular criterion Φ. For the criterion $\mathcal{C}_{max}$, the stability radius is denoted by $\widehat{\varrho}_s(p)$, and for the criterion $\sum \mathcal{C}_i$ by $\overline{\varrho}_s(p)$. In what follows, we use whenever appropriate the notion "stability radius of an optimal digraph $G_s \in \Lambda(G)$" instead of "stability radius of an optimal schedule $s \in S$". Due to the maximum metric (1.5), the set $O_\varrho(p) \cap R^q_+$ is a polytope for any positive $\varrho \in R^1_+$.

Definition 1.2 implies a general approach for calculating $\varrho_s(p)$, which is discussed in the rest of this section for any regular criterion Φ and which is concretized for $\Phi = \mathcal{C}_{max}$ and for $\Phi = \sum \mathcal{C}_i$ in Section 1.3 and in Section 1.4, respectively. Formulas for calculating the stability radius for the makespan criterion and the characterization of the extreme values of $\widehat{\varrho}_s(p)$ are proven in Section 1.3. The same questions for the mean flow time criterion are considered in Section 1.4.

Next, the calculation of the stability radius $\varrho_s(p)$ is reduced to the solution of a non-linear programming problem. We give this reduction for the general shop problem $\mathcal{G}//\Phi$ provided that the set of all operations $Q = \{1, 2, \ldots, q\}$ is partitioned into n linearly ordered subsets of the operations Q^{J_i} defining the technological routes of the jobs $J_i \in J = \{J_1, J_2, \ldots, J_n\}$ (see partition (1.3)). It should be noted that partition (1.3) does not cause any restriction on the generality of problem $\mathcal{G}//\Phi$ since one can assume that $|Q^{J_i}| = 1$ for a job $J_i \in J$, i.e., this job consists of only one operation and as a result, any precedence constraints may be given on the set of operations Q.

We denote by $\{\mu\}$ the set of vertices which form a path μ in digraph G_k and by $l^p(\mu)$ the weight of this path:

$$l^p(\mu) = \sum_{i \in [\mu]} p_i. \quad (1.7)$$

Let operation $j_i \in Q^{J_i} \subseteq Q$ be the last operation in the technological route of job J_i, $1 \le i \le n$, and $\tilde{H}^i_k$ denote the set of all paths in digraph $G_k = (Q, A \cup E_k, \emptyset) \in \Lambda(G)$ ending in vertex $j_i \in Q^{J_i} \subseteq Q$. Obviously, the completion time $C_i(k) = c_{j_i}(k)$ of job J_i in a semiactive schedule $k \in S$ is equal to the value $\max_{\mu \in \tilde{H}^i_k} l^p(\mu)$ of the largest weight of a path in the set $\tilde{H}^i_k$. While calculating $c_{j_i}(k)$, $J_i \in J$, it is sufficient to consider only a subset of the set $\tilde{H}^i_k$ due to the following dominance relation defined on the set of paths.

Definition 1.3 *The path $\mu \in \tilde{H}_s^i$ is called dominant if there is no path $\nu \in \tilde{H}_s^i$ such that $\{\mu\}$ is a proper subset of set $\{\nu\}$: $\{\mu\} \subset \{\nu\}$. Otherwise, if set $\{\mu\}$ is a proper subset of set $\{\nu\}$, path μ is dominated by path ν.*

The dominance relation given in Definition 1.3 is a *strict order* since both the *transitivity* property and the *anti-reflexivity* property hold. Indeed, the inclusions $\{\mu\} \subset \{\nu\}$ and $\{\nu\} \subset [\tau]$ imply $\{\mu\} \subset [\tau]$ (transitivity) and the inclusion $\{\mu\} \subset \{\mu\}$ does not hold for any path μ (anti-reflexivity). Let H_k^i denote the set of all dominant paths in the set $\tilde{H}_k^i$. Since $p_i \geq 0$ for all operations $i \in Q$, we obtain

$$C_i(k) = c_{j_i}(k) = \max_{\mu \in H_k^i} l^p(\mu).$$

Thus, the value of the objective function $\Phi(C_1, C_2, \ldots, C_n)$ for the semiactive schedule $s = (c_1(s), c_2(s), \ldots, c_q(s)) \in S$ may be calculated as follows:

$$\Phi(\max_{\mu \in H_s^1} l^p(\mu), \max_{\mu \in H_s^2} l^p(\mu), \ldots, \max_{\mu \in H_s^n} l^p(\mu)).$$

Therefore: *The semiactive schedule $s = (c_1(s), c_2(s), \ldots, c_q(s)) \in S$ is optimal for problem $\mathcal{G}//\Phi$ with a regular criterion Φ if and only if*

$$\Phi(\max_{\mu \in H_s^1} l^p(\mu), \ldots, \max_{\mu \in H_s^n} l^p(\mu)) = \min_{k=1,2,\ldots,\lambda} \Phi(\max_{\nu \in H_k^1} l^p(\nu), \ldots, \max_{\nu \in H_k^n} l^p(\nu)). \quad (1.8)$$

For brevity, we denote $\Phi_s^p = \Phi(\max_{\mu \in H_s^1} l^p(\mu), \ldots, \max_{\mu \in H_s^n} l^p(\mu))$.

Let the subset $S^\Phi(p)$ of set S, $S^\Phi(p) \subseteq S$, denote the set of all optimal semiactive schedules for problem $\mathcal{G}//\Phi$ with the vector $p = (p_1, p_2, \ldots, p_q) \in R_+^q$ of the operation processing times and let inclusion $s \in S^\Phi(p)$ hold. Then, due to Definition 1.2, the stability radius $\varrho_s(p)$ may be defined as follows:

$$\varrho_s(p) = \inf\{d(p, x) : x \in R_+^q, s \notin S^\Phi(x)\}. \quad (1.9)$$

While equality (1.6) defines the stability radius $\varrho_s(p)$ as the maximal real number which may be a radius of a stability ball $O_\varrho(p)$, equality (1.9) defines the value of $\varrho_s(p)$ as the infimum of the non-negative real numbers that cannot be a radius of a stability ball $O_\varrho(p)$. From equalities (1.8) and (1.9), it follows that, in order to calculate the stability radius $\varrho_s(p)$, it is sufficient to calculate the optimal value of the objective function $f(x_1, x_2, \ldots, x_q)$ of the following non-linear programming problem:

$$\text{Minimize } f(x_1, x_2, \ldots, x_q) = \max_{i=1,2,\ldots,q} |x_i - p_i| \quad (1.10)$$

subject to

$$\Phi_s^x > \min\{\Phi_k^x \ : \ k = 1, 2, \ldots, \lambda; \ k \neq s\}, \tag{1.11}$$

$$x_i \geq 0, \quad i = 1, 2, \ldots, q. \tag{1.12}$$

If condition (1.11) is not satisfied for any non-negative vector $x \in R_+^q$, then digraph $G_s(p)$ is optimal for all vectors $x \in R_+^q$ of the operation processing times: $s \in S^\Phi(x)$, $x \in R^q$, and we obtain

$$\begin{cases} \Phi_s^x \leq \min\{\Phi_k^x \ : \ k = 1, 2, \ldots, \lambda; \ k \neq s\}, \\ x_i \geq 0, i = 1, 2, \ldots, q. \end{cases}$$

In this case, we say that the stability radius $\varrho_s(p)$ is *infinitely large*. To indicate that the stability radius is infinitely large we write: $\varrho_s(p) = \infty$.

In all other cases, there exists an optimal value f^* of the objective function of problem (1.10) – (1.12): $f^* = \inf \max_{i=1,2,\ldots,q} |x_i - p_i|$, where the infimum is taken over all vectors x satisfying conditions (1.11) and (1.12). To find the value f^*, it is sufficient to calculate a solution $x^0 = (x_1^0, x_2^0, \ldots, x_q^0)$ of the following non-linear programming problem (1.13) – (1.15) which is obtained from problem (1.10) – (1.12) due to the replacement of the sign $>$ in inequality (1.11) by the sign $\geq$:

$$\text{Minimize } f(x_1, x_2, \ldots, x_q) = \max_{i=1,2,\ldots,q} |x_i - p_i| \tag{1.13}$$

subject to

$$\Phi_s^x \geq \min\{\Phi_k^x \ : \ k = 1, 2, \ldots, \lambda; \ k \neq s\}, \tag{1.14}$$

$$x_i \geq 0, \quad i = 1, 2, \ldots, q. \tag{1.15}$$

It is clear that equalities

$$f^* = \max_{i=1,2,\ldots,q} |x_i^0 - p_i| = d(x^0, p) = \varrho_s(p)$$

hold. For any small $\epsilon > 0$, there exists a vector $x^\epsilon = (x_1^\epsilon, x_2^\epsilon, \ldots, x_q^\epsilon) \in R_+^q$ such that $d(x^\epsilon, p) = \varrho_s(p) + \epsilon$ and $s \notin S^\Phi(x^\epsilon)$.

It may occur that the given vector p is itself a solution to the non-linear programming problem (1.13) – (1.15). In the latter case, equalities $\varrho_s(p) = d(p, p) = 0$ hold and it means that the optimality of digraph $G_s(p)$ is *unstable*: For any small real $\epsilon > 0$, there exists a vector $p' \in R_+^q$ such that $s \notin S^\Phi(p')$ and $d(p, p') = \epsilon$. In this case, we say that the optimal digraph $G_s(p)$ is unstable.

If the given vector p is not a solution of problem (1.13) – (1.15), we have the strict inequality $\varrho_s(p) > 0$. In this case, the optimal digraph

$G_s(p)$ is *stable*. The direct use of the solution of problem (1.13) – (1.15) for calculating the stability radius $\varrho_s(p)$ is only possible for a general shop scheduling problem $\mathcal{G}//\Phi$ with a very small size since we have to know all λ feasible semiactive schedules $s \in S$ for this problem and for each digraph G_s, we have to know the sets H_k^i of all dominant paths ending in the last operation j_i of each job J_i.

1.3. Maximum Flow Time

The best studied case of the general shop problem $\mathcal{G}//\Phi$ (and of the job shop and flow shop problems) is the one with $\Phi = \mathcal{C}_{max}$, where the objective is to find an *optimal* schedule for the makespan criterion, i.e., to find a feasible schedule $s = (c_1(s), c_2(s), \ldots, c_q(s))$ with a minimum value of the maximum flow time $\max\{c_i(s) : i \in Q\}$ among all semiactive schedules S. In this section, the stability radius of a solution of problem $\mathcal{G}//\Phi$ with $\Phi = \mathcal{C}_{max}$ is considered.

Let $\tilde{H}_k$ ($\tilde{H}$, respectively) be the set of all paths in digraph $G_k(p) \in \Lambda(G)$ (in digraph $(Q, A, \emptyset)$) constructed for the general shop problem $\mathcal{G}//\mathcal{C}_{max}$. Let H and H_k denote the set of all dominant paths in digraph $(Q, A, \emptyset)$ and in digraph $G_k \in \Lambda(G)$, respectively (see Definition 1.3). Thus, we can write

$$H_k = \{\nu \in \tilde{H}_k : \text{ Inclusion } \{\nu\} \subset \{\mu\} \text{ does not hold for any path } \mu \in \tilde{H}_k\}.$$

The set $H \subseteq \tilde{H}$ is defined similarly. The value of $\max_{i=1}^{n} C_i$ of a schedule s is given by the weight of a maximum-weight path (called a *critical path*) in the weighted digraph $G_s(p)$. Obviously, at least one critical path in $G_s(p)$ is dominant and for any path $\mu \in H$, there exists a path $\nu \in H_s$ such that either ν dominates path μ or inclusion $\mu \in H_s$ holds. Thus, equality (1.8) for problem $\mathcal{G}//\mathcal{C}_{max}$ is converted into the following one:

$$\max_{\mu \in H_s} l^p(\mu) = \min_{k=1,2,\ldots,\lambda} \max_{\nu \in H_k} l^p(\nu). \tag{1.16}$$

Therefore: *The schedule $s = (c_1(s), c_2(s), \ldots, c_q(s)) \in S$ is optimal for problem $\mathcal{G}//\mathcal{C}_{max}$ with the makespan criterion if and only if equality (1.16) holds.*

Zero Stability Radius

Let $H_k(p)$ denote the set of all critical dominant paths in the weighted digraph $G_k(p)$, $G_k \in \Lambda(G)$. Obviously, we have $H_k(p) \subseteq H_k$. To prove necessary and sufficient conditions for the equality $\widehat{\varrho}_s(p) = 0$, we need the following auxiliary claim.

Lemma 1.3 *There exists a real $\epsilon > 0$ such that the set $H_k \backslash H_k(p)$ contains no critical path of digraph $G_k \in P(G)$ for any vector of the processing times $p^\epsilon = (p_1^\epsilon, p_2^\epsilon, \ldots, p_q^\epsilon) \in O_\epsilon(p) \cap R_+^q : H_k(p^\epsilon) \subseteq H_k(p)$.*

PROOF. We can calculate the positive real number

$$\epsilon_k = \min_{\nu \in H_k \backslash H_k(p)} \frac{l_k^p - l^p(\nu)}{2q}. \tag{1.17}$$

Hereafter l_k^p denotes the *critical weight* of digraph $G_k \in P(G)$ which defines the value of the objective function $\mathcal{C}_{max}$ for the schedule k with the vector p of processing times:

$$l_k^p = \max_{\mu \in H_k} l^p(\mu) \tag{1.18}$$

For any real ϵ, which satisfies the inequalities $0 < \epsilon < \epsilon_k$, the difference in the right-hand side of equality (1.17) remains positive when vector p is replaced by any vector $p^\epsilon \in O_\epsilon(p) \cap R_+^q$. Indeed, the number of vertices in any path ν in digraph G_k is at most equal to q and, therefore, the difference $l_k^p - l^p(\nu)$ may not be 'overcome' by a vector p^ϵ if $d(p, p^\epsilon) < \epsilon_k$.

$\diamond$

Theorem 1.1 *For an optimal schedule $s \in S^\Phi(p)$, $\Phi = \mathcal{C}_{max}$, of problem $\mathcal{G}//\mathcal{C}_{max}$, equality $\widehat{\varrho}_s(p) = 0$ holds if and only if there exists another optimal schedule $k \in S^\Phi(p)$, $k \neq s$, and there exists a path $\mu^* \in H_s(p)$ such that there does not exist a path $\nu^* \in H_k(p)$ with $\{\mu^*\} \subseteq \{\nu^*\}$.*

PROOF. *Sufficiency.* We have to prove that, if the conditions of the theorem are satisfied, then we obtain $\widehat{\varrho}_s(p) < \epsilon$ for any given $\epsilon > 0$.

To this end, we construct a vector $p^* = (p_1^*, p_2^*, \ldots, p_q^*)$ with

$$p_i^* = \begin{cases} p_i + \epsilon^*, & \text{if } i \in \{\mu^*\} \\ p_i & \text{otherwise,} \end{cases}$$

where $\epsilon^* = \min\{\epsilon_k, \epsilon_s, \epsilon\}$ with ϵ_s and ϵ_k defined as in (1.17). If

$$\max_{\nu \in H_k} l^{p^*}(\nu) = l^{p^*}(\nu^0),$$

then due to Lemma 1.3 and the inequalities $p_i^* \geq p_i$, $i \in Q$, we obtain

$$\begin{aligned} l^{p^*}(\nu^0) &= \max_{\nu \in H_k(p)} l^{p^*}(\nu) \\ &= l^p(\nu^*) + \epsilon^* |\{\mu^*\} \cap \{\nu^*\}| = l^p(\mu^*) + \epsilon^* |\{\mu^*\} \cap \{\nu^*\}|. \end{aligned} \tag{1.19}$$

Since inclusion $\{\mu^*\} \subseteq \{\nu^*\}$ does not hold for any $\nu^* \in H_k(p)$, inequality $|\{\mu^*\} \cap \{\nu^*\}| < |\{\mu^*\}|$ holds and we can continue (1.19) in the following way:

$$l^p(\mu^*) + \epsilon^* |\{\mu^*\} \cap \{\nu^*\}| < l^p(\mu^*) + \epsilon^* |\{\mu^*\}| = l^{p^*}(\mu^*) = \max_{\mu \in H_s} l^{p^*}(\mu).$$

Thus, we obtain $l_k^{p^*} < l_s^{p^*}$ and $s \notin S^\Phi(p^*)$, which imply inequality $\widehat{\varrho}_s(p) < \epsilon$ because of the condition $d(p, p^*) = \epsilon^* \leq \epsilon$.

Necessity. We prove necessity by contradiction. Suppose that $\widehat{\varrho}_s(p) = 0$ but the conditions of the theorem are not satisfied. We consider two cases i) and ii) of violating these conditions.

i) Assume that there does not exist another optimal makespan schedule: $S^\Phi(p) = \{s\}$. Then we calculate the real number

$$\epsilon^0 = \frac{1}{2q} \min\{l_t^p - l_s^p : t = 1, 2, \ldots, \lambda;\ t \neq s\}.$$

Since s is the only optimal schedule, we obtain $\epsilon^0 > 0$. For any positive real $\epsilon < \epsilon^0$, the difference $l_t^p - l_s^p$ remains positive when vector p is replaced by an arbitrary vector $p^0 \in O_\epsilon(p) \cap R_+^q$. So, we can conclude that digraph G_s remains optimal for any vector p^0 of the processing times. Therefore, we have $\widehat{\varrho}_s(p) \geq \epsilon > 0$ which contradicts the assumption $\widehat{\varrho}_s(p) = 0$.

ii) Assume that $|S^\Phi(p)| > 1$ and for any schedule $k \in S^\Phi(p),\ k \neq s$, and for any path $\mu^* \in H_s(p)$, there exists a path $\nu_k^* \in H_k(p)$ such that $\{\mu^*\} \subseteq \{\nu_k^*\}$. In this case, we can take any ϵ that satisfies the inequalities

$$0 < \epsilon < \min\left\{ \min_{k \in \phi^{max}(p)} \epsilon_k,\ \frac{1}{2q} \min\{l_t^p - l_s^p : t = 1, 2, \ldots, \lambda;\ t \notin S^\Phi(p)\} \right\}.$$

From Lemma 1.3, due to inequality $\epsilon < \epsilon_s$, we get equality

$$l_s^{p^0} = \max_{\mu \in H_s(p^0)} l^{p^0}(\mu) = \max_{\mu \in H_s(p)} l^{p^0}(\mu) \tag{1.20}$$

for any vector $p^0 \in O_\epsilon(p) \cap R_+^q$. Since inequalities $\epsilon < \epsilon_s$ and $\epsilon < \epsilon_k$ hold, and there exists a path $\nu_k^* \in H_k(p),\ k \in S^\Phi(p),\ k \neq s$, for any path $\mu^* \in H_s(p)$ such that $\{\mu^*\} \subseteq \{\nu_k^*\}$, we obtain inequality

$$\max_{\mu \in H_s(p)} l^{p^0}(\mu) \leq \max_{\nu \in H_k(p)} l^{p^0}(\nu). \tag{1.21}$$

Thus, due to (1.20) and (1.21), we obtain

$$l_s^{p^0} \leq \max_{\nu \in H_k(p)} l^{p^0}(\nu) \tag{1.22}$$

for any optimal schedule $k \in S^{\Phi}(p)$, $k \neq s$. Since inequality

$$\epsilon < \frac{1}{2q} \min\{l_t^p - l_s^p : t = 1, 2, \ldots, \lambda; t \notin S^{\Phi}(p)\}$$

holds, condition $t \notin S^{\Phi}(p)$ implies $t \notin S^{\Phi}(p^0)$. Taking into account (1.22) and the latter implication, we can conclude that $s \in S^{\Phi}(p^0)$ for any vector $p^0 \in R_+^q$ with $d(p, p^0) \leq \epsilon$. Consequently, we obtain $\widehat{\varrho}_s(p) \geq \epsilon > 0$, which contradicts the assumption $\widehat{\varrho}_s(p) = 0$.

$\diamond$

Obviously, the conditions of Theorem 1.1 are violated if $H_s(p) \subseteq H$. Therefore, the following corollary holds.

Corollary 1.1 *If s is an optimal schedule for problem $\mathcal{G}//\mathcal{C}_{max}$ and $H_s(p) \subseteq H$, then $\widehat{\varrho}_s(p) > 0$.*

For the following corollary from Theorem 1.1, it is not necessary to know the set $H_s(p)$.

Corollary 1.2 *If s is a unique optimal schedule for problem $\mathcal{G}//\mathcal{C}_{max}$, then $\widehat{\varrho}_s(p) > 0$.*

Infinite Stability Radius

Next, we prove the following characterization of an infinitely large stability radius.

Theorem 1.2 *For an optimal schedule $s \in S^{\Phi}(p)$, $\Phi = \mathcal{C}_{max}$, of problem $\mathcal{G}//\mathcal{C}_{max}$, the stability radius $\widehat{\varrho}_s(p)$ is infinitely large if and only if for any path $\mu \in H_s \backslash H$ and for any digraph $G_t(p) \in \Lambda(G)$, there exists a path $\nu \in H_t$ such that $\{\mu\} \subseteq \{\nu\}$.*

Proof. *Necessity.* Following the contradiction method, we suppose that $\widehat{\varrho}_s(p) = \infty$ but there exist a path $\mu \in H_s \backslash H$ and a digraph $G_t \in P(G)$ such that for any $\nu \in H_t$ relation $\{\mu\} \subseteq \{\nu\}$ does not hold.

We set $\epsilon' = \max_{i=1}^{q} p_i$ and consider the vector $p' = (p'_1, p'_2, \ldots, p'_q) \in R_+^q$, where

$$p'_i = \begin{cases} \epsilon', & \text{if } i \in \{\mu\} \\ 0 & \text{otherwise.} \end{cases}$$

For any path $\nu \in H_t$, we have $l^{p'}(\nu) = \epsilon'|\{\mu\} \cap \{\nu\}|$. Since relation $\{\mu\} \subseteq \{\nu\}$ does not hold, we obtain $l^{p'}(\nu) < l^{p'}(\mu)$. Therefore, $l_t^{p'} < l^{p'}(\mu) = l_s^{p'}$ and hence $s \notin S^{\Phi}(p')$. We get a contradiction:

$$\hat{\varrho}_s(p) < d(p', p) \leq \max_{i=1}^{q} p_i < \infty.$$

Sufficiency. Let ϵ be a positive number as large as desired. We take any vector $p^0 \in O_\epsilon(p) \cap R_+^q$ and suppose that $l_s^{p^0} = l^{p^0}(\mu)$. If $\mu \in H$, then inequality $l^{p^0}(\mu) \leq l_t^{p^0}$ holds for every $t = 1, 2, \ldots, \lambda$. If $\mu \in H_s \backslash H$, then, due to the conditions of the theorem, for any feasible schedule t, there exists a path $\nu \in H_t$ such that $\{\mu\} \subseteq \{\nu\}$. Therefore, we get $l^{p^0}(\mu) \leq l^{p^0}(\nu) \leq l_t^{p^0}$. Thus, in both cases we obtain $s \in S^{\Phi}(p^0)$.

$\diamond$

Directly from the above proof of necessity we obtain the following corollary which gives a simple upper bound for the stability radius $\hat{\varrho}_s(p)$.

Corollary 1.3 *If $\hat{\varrho}_s(p) < \infty$, then $\hat{\varrho}_s(p) \leq \max_{i=1}^{q} p_i$.*

Due to Theorem 1.2, one can identify a general shop problem $\mathcal{G}//\mathcal{C}_{max}$ whose optimal schedule is implied only by the precedence constraints given on the set of operations Q and by the given distribution of the operations Q to the machines from the set M, but independent of the processing times $p \in R_+^q$ of the operations Q. However, because of the generality of problem $\mathcal{G}//\mathcal{C}_{max}$, it is practically difficult to check the conditions of Theorem 1.2.

Next, it is shown that for a job shop problem $\mathcal{J}//\mathcal{C}_{max}$, there are necessary and sufficient conditions for $\hat{\varrho}_s(p) = \infty$ which can be verified in $O(q^2)$ time. To present the latter conditions, we need the following notations.

Let A_k (B_k, respectively) be the set of all operations $i \in Q$ such that $i \to j$ ($j \to i$) and $j \in Q_k,\ i \notin Q_k$:

$$A_k = \{i \ : \ i \to j,\ i \in Q \cap Q_k,\ j \in Q_k\};$$

$$B_k = \{j \ : \ i \to j,\ j \in Q \cap Q_k,\ i \in Q_k\}.$$

For a set B of operations, let $n(B)$ denote the number of jobs in the set J having at least one operation in the set B.

Theorem 1.3 *For problem $\mathcal{J}//\mathcal{C}_{max}$, there exists an optimal digraph $G_s(p)$ with an infinitely large stability radius if and only if the following two conditions hold:*
1) Inequality $\max\{|A_k|, |B_k|\} \leq 1$ holds for any machine M_k with $n(Q_k) > 1$;
2) If there exist two operations $g \in A_k$ and $f \in B_k$ of job J_l, then there exists a path from vertex f to vertex g in digraph $(Q, A, \emptyset)$ (possibly $f = g$).

PROOF. *Necessity.* Let $\hat{\varrho}_s(p) = \infty$, but $|A_k| > 1$.

If $n(A_k) > 1$, then there exist at least two jobs J_u and J_v, $u \neq v$, which have to be processed by different machines and after that, both jobs have to be processed by the same machine M_k. We can choose operations $i \in Q_k \cap Q^{(u)}$, $f \in A_k \cap Q^{(u)}$, $j \in Q_k \cap Q^{(v)}$ and $g \in A_k \cap Q^v$ such that both relations $f \to i$ and $g \to j$ hold. Since $\hat{\varrho}_s(p) = \infty$, digraph G_s has to be optimal for any vector $p \in R^q_+$ of operation processing times.

In particular, digraph G_s has to be optimal for vector p with $p_f = p_j = 1$ and $p_h = 0$ for the operations $h \in \{1, 2, ..., q\} \setminus \{f, j\}$. Since operations f and j are processed by different machines, we obtain $\mathcal{C}_{max}(s) = 1$. Hence, arc (j, i) has to belong to set E_s: $(j, i) \in E_s$.

On the other hand, if we consider vector p' with $p'_g = p'_i = 1$ and $p'_h = 1$ for the operations $h \in \{1, 2, ..., q\} \setminus \{g, i\}$, then similarly, we obtain that arc (i, j) has to belong to set E_s: $(i, j) \in E_s$. Thus, we obtain a contradiction, which implies that there is no digraph $G_s \in \Lambda(G)$ which remains optimal for any vector $p \in R^q_+$ of operation processing times. In this case, inequality $\hat{\varrho}_s(p) < \infty$ must hold for any digraph $G_s \in \Lambda(G)$.

Next, we have to consider the case when $|A_k| > 1$, but $n(A_k) = 1$. It is clear that the set of all operations A_k has to belong to the same job, say, job J_w.

(a) Since $|Q_k/J| > 1$, there exist an operation $j \in Q_k \setminus Q^{J_w}$ and an operation $b \in Q^{J_w} \cap Q_k$ such that in digraph $(Q, A, \emptyset)$, there exists a path of the form $(g, ..., b, ..., f)$ with two operations $\{g, f\}$, which do not belong to set Q_k and inclusion $[j, b] \in E$ holds.

Let $G_s \in \Lambda(G)$ be an optimal digraph with an infinitely large stability radius: $\hat{\varrho}_s(p) = \infty$. If we set $p_g = p_j = 1$ and we set $p_h = 0$ for any other $h \in \{1, 2, ..., q\} \setminus \{g, j\}$, then for such a vector $p \in R^q_+$ of the operation durations arc (j, b) has to belong to set E_s. On the other hand, if we consider vector p' with $p'_j = p'_f = 1$ and with $p'_h = 0$ for any other $h \in \{1, 2, ..., q\} \setminus \{j, f\}$, then arc (b, j) has to belong to set E_s. Hence, it is impossible to construct a digraph $G_s \in \Lambda(G)$ which remains optimal for any vector $p \in R^q_+$ of operation processing times: $\hat{\varrho}_s(p) < \infty$.

Thus, condition $\hat{\varrho}_s(p) = \infty$ implies inequality $|A_k| \leq 1$. Similarly, we can prove that equality $\hat{\varrho}_s(p) = \infty$ implies $|B_k| \leq 1$.

(b) If there exists a job $J_l \in J$ such that $A_k \cap Q^{J_l} = g$ and $B_k \cap Q^{J_l} = f$ and there exists a path from vertex g to vertex f in digraph $(Q, A, \emptyset)$, then similarly as in case (a), one can prove that $\hat{\varrho}_s(p) < \infty$.

Sufficiency. Obviously, the set of machines satisfying the conditions of Theorem 1.3 can be partitioned into the following five subsets:

1) $|A_k| = |B_k| = 0$;
2) $|A_k| = 1$, $|B_k| = 0$;
3) $|A_k| = 0$, $|B_k| = 1$;
4) $|A_k| = |B_k| = 1$ and there exists a job J_l such that there exists a path from vertex $f \in B_k \cap Q^{J_l}$ to vertex $g \in A_k \cap Q^{J_l}$ in digraph $(Q, A, \emptyset)$ (possibly $f = g$);
5) $|A_k| = |B_k| = 1$ and there is no set Q^{J_l} containing both sets A_k and B_k.

Now, we can consider any mixed graph G in which for any machine $M_k \in M$ with $n(Q_k) > 1$ one of the conditions 1) – 5) holds. Then we construct a digraph $G_s \in \Lambda(G)$ using the following algorithm.

Algorithm $RAD\_\widehat{\varrho}_s(p) = \infty$

Input: Mixed graph $G = (Q, A, E)$ for problem $\mathcal{J}//\mathcal{C}_{max}$.
Output: Digraph G_s with $\widehat{\varrho}_s(p) = \infty$ (if such a digraph exists).

Step 1: **IF** for machine M_k condition 1) holds **THEN** orient all edges incident with operations Q_k arbitrarily without circuit appearance.

Step 2: **IF** for machine M_k condition 4) holds **THEN** partition the set $Q^{J_l} \cap Q^k$ into two subsets R and L, where set R (set L) contains each vertex h, if there exists a path from vertex h (vertex g) to vertex g (vertex h) in digraph $(Q, A, \emptyset)$.

Step 3: Orient the edges of set E^k incident to operations from set Q_k as follows:
IF an edge is incident with vertices from set R (the set L, respectively)
THEN the generated arcs are leaving (entering) the vertices from set R (set L). The remaining edges have to be oriented arbitrarily without circuit appearance.

Step 4: **IF** for machine M_k one of the conditions 2), 3) or 5) holds
THEN the edges incident with Q_k have to be oriented as follows: Edges incident with vertex Q^L, $Q^L \cap A_k \neq \emptyset$, have to be replaced by the arcs entering the operations Q^{J_l}; edges incident with vertex Q^L, $Q^L \cap A_k \neq \emptyset$, have to be replaced

by the arcs leaving the operations Q^{J_l}.
The remaining edges have to be oriented
arbitrarily without circuit appearance. **STOP**

Obviously, inclusion $G_s \in \Lambda(G)$ holds. It is also easy to convince that for any path μ in digraph G_s, either there exists a machine $M_k \in M$ such that $\{\mu\} \subseteq Q_k$ or there exists a job $J_l \in J$ such that $\{\mu\} \subseteq Q^{J_l}$. Therefore, for the digraph G_s constructed by Algorithm $RAD\_\hat{\varrho}_s(p) = \infty$, equality

$$C_{max} = \max\left\{\max_k \sum_{i \in Q_k} p_i,\ \max_l \sum_{i \in Q^{J_l}} p_i\right\}$$

holds. Consequently, digraph G_s is optimal. This digraph remains optimal for any vector $p \in R^q_+$ of operation processing times, and so $\hat{\varrho}_s(p) = \infty$.

$\diamond$

From Theorem 1.3 it follows that there are job shop problems $\mathcal{J}//\mathcal{C}_{max}$ with an optimal schedule having an infinitely large stability radius for any given number of jobs n and number of machines m. Next, we show that testing the conditions of Theorem 1.3 takes $O(q^2)$ time.

Let the input data given for problem $\mathcal{J}//\mathcal{C}_{max}$ include the set of jobs $J = \{J_1, J_2, ..., J_n\}$, the sets $Q_1, Q_2, ..., Q_m$ and the technological routes of jobs J. For testing inequality $|A_k| > 2$ (inequality $|B_k| > 2$) for each $k = 1, 2, ..., m$, it is sufficient for each operation $i \in Q_k$ to find the operation which follows after i. It is easy to see that such a testing needs $O(\sum_{k=1}^{n}(q|Q_k| + q))$ time, i.e., $O(q^2)$ time. If $|A_k| \leq 2$ and $|B_k| \leq 2$ for each k, then it is sufficient to consider machine M_k such that equalities $|A_k| = |B_k| = 1$ hold and to test the condition of Theorem 1.3 which needs $O(q)$ time. Hence, $O(q^2)$ time is sufficient to test whether a problem $\mathcal{J}//\mathcal{C}_{max}$ has a solution with an infinitely large stability radius.

Note that for a flow shop problem, such a schedule can exist only if n or m is equal to 1. Indeed, for problem $\mathcal{F}//\mathcal{C}_{max}$, we have $n(A_k) > 1$ for any machine M_k with $k \geq 2$ provided that $n \geq 2$ and $m \geq 2$. Thus, Theorem 1.3 implies the following corollary.

Corollary 1.4 *For problem $\mathcal{F}//\mathcal{C}_{max}$ with $n \geq 2$ and $m \geq 2$, inequality $\hat{\varrho}_s(p) > 0$ holds.*

In the next section, it is shown that there does not exist an optimal schedule s with $\varrho_s(p) = \infty$ for a problem $\mathcal{J}//\Phi$ with all other regular criteria

Φ which are considered in contemporary scheduling theory (these criteria are given, e.g., in [204]).

Formula for Calculating the Stability Radius

Next, we derive a formula for calculating the stability radius $\widehat{\varrho}_s(p)$. This calculation is reduced to the solution of an extremal problem on the set of digraphs $\Lambda(G) = \{G_1, G_2, \ldots, G_\lambda\}$ with a variable vector of weights assigned to the vertices of digraph $G_k \in \Lambda(G)$. The main objects for the calculation are the sets of dominant paths $H_k,\ k = 1, 2, \ldots, \lambda$.

Assume that inequality $\widehat{\varrho}_s(p) < \infty$ holds for a given optimal schedule $s \in S^\Phi(p),\ \Phi = \mathcal{C}_{max}$, of problem $\mathcal{G}//\mathcal{C}_{max}$. Using equality (1.16), we can conclude that equality (1.9) for criterion $\Phi = \mathcal{C}_{max}$ is converted into the following one:

$$\widehat{\varrho}_s(p) = \inf\{d(p,x) : x \in R^q_+,\ \max_{\mu\in H_s} l^x(\mu) > \min_{k=1,2,\ldots,\lambda;\ k\neq s}\ \max_{\nu\in H_k} l^x(\nu)\}.$$

Therefore, to find the stability radius $\widehat{\varrho}_s(p)$ it is sufficient to construct a vector $x \in R^q_+$ that satisfies the following three conditions.

(a) There exists a digraph $G_k \in \Lambda(G),\ k \neq s$, such that $l^x_s = l^x_k$, i.e.,

$$\max_{\mu\in H_s} l^x(\mu) = \max_{\nu\in H_k} l^x(\nu). \tag{1.23}$$

(b) For any given real $\epsilon > 0$, which may be as small as desired, there exists a vector $p^\epsilon \in R^q_+$ such that $d(x, p^\epsilon) = \epsilon$ and inequality

$$\max_{\mu\in H_s} l^{p^\epsilon}(\mu) > \max_{\nu\in H_k} l^{p^\epsilon}(\nu) \tag{1.24}$$

is satisfied for at least one digraph $G_k \in \Lambda(G)$.

(c) The distance $d(p, x)$ achieves its minimal value among the distances between the vector p and the other vectors in the vector space R^q_+ satisfying both the above conditions (a) and (b).

After having constructed such a vector $x \in R^q_+$, one can define the stability radius of digraph G_s as follows: $\widehat{\varrho}_s(p) = d(p, x)$, since the critical path in digraph G_s becomes larger than that of digraph G_k for any vector $p^\epsilon \in R^q_+$ with a positive real ϵ, which may be as small as desired (see condition (b)), and so digraph G_s is no longer optimal, while for any vector from the intersection of the ball $O_{d(p,x)}(p) \subset R^q$ with the set R^q_+ digraph G_s remains optimal (see condition (c)). Digraph G_k, which satisfies conditions (a), (b) and (c) is called a *competitive* digraph for the optimal digraph G_s.

Thus, the calculation of the stability radius may be reduced to a rather sophisticated extremal problem on the given set of digraphs $\Lambda(G) = \{G_1, G_2, \ldots, G_\lambda\}$ with a variable vector of weights assigned to the vertices of each digraph $G_k \in \Lambda(G)$. As it follows from (1.23) and (1.24), the main objects for such a calculation are the sets of dominant paths $H_k,\ k = 1, 2, \ldots, \lambda$.

To satisfy conditions (a), (b) and (c), we look next for a vector $x = p(r) = (p_1(r), p_2(r), \ldots, p_q(r)) \in R^q_+$ with the components $p_i(r) \in \{p_i, p_i + r, p_i - r\}$ on the basis of a direct comparison of the paths from the set H_s and the paths from the sets H_k, where $k = 1, 2, \ldots, \lambda$ and $k \neq s$.

Let the value $l^p(\nu)$ be greater than the length of a critical path in an optimal digraph G_s. To satisfy equality (1.23), the length of a path $\nu \in H_k$ has to be not greater than that of at least one path $\mu \in H_s$ and there must be a path $\nu \in H_k$ with a length equal to the length of a critical path in G_s. Thus, if we have calculated

$$r_\nu = \min_{\mu \in H_s} \frac{l^p(\nu) - l^p(\mu)}{|\{\mu\} \cup \{\nu\}| - |\{\mu\} \cap \{\nu\}|}, \tag{1.25}$$

we obtain equality

$$\max_{\mu \in H_s} l^{p(r)}(\mu) = l^{p(r)}(\nu) \tag{1.26}$$

for the vector $p(r) = p(r_\nu)$ with the components

$$p_i(r) = p_i(r_\nu) = \begin{cases} p_i + r_\nu, & \text{if } i \in \{\mu\}, \\ p_i - r_\nu, & \text{if } i \in \{\nu\} \backslash \{\mu\}, \\ p_i, & \text{if } i \notin \{\mu\} \cup \{\nu\}. \end{cases} \tag{1.27}$$

Further, we shall use the following remark: *Due to (1.25), vector $p(r)$ calculated in (1.27) is the closest one to the given vector p among all vectors for which equality (1.26) holds.*

To reach equality (1.23) for the whole digraph G_k, we have to repeat calculation (1.25) for each path $\nu \in H_k$ with inequality $l^p(\nu) > l^p_s$. Thus, instead of vector $p(r_\nu)$ we have to consider the vector $p(r) = p(r_{G_k})$ calculated according to formula (1.27), where

$$r_{G_k} = \min_{\mu \in H_s} \max_{\nu \in H_k;\, l^p(\nu) > l^p_s} \frac{l^p(\nu) - l^p(\mu)}{|\{\mu\} \cup \{\nu\}| - |\{\mu\} \cap \{\nu\}|}. \tag{1.28}$$

Next, we consider inequality (1.24). Since the processing times are non-negative, this inequality may not be valid for a vector $p^\epsilon \in R^q_+$ if path μ is dominated by path ν. Thus, we can restrict our consideration to the subset

H_{sk} of the set H_s of all paths, which are not dominated by paths from the set H_k:

$$H_{sk} = \{\mu \in H_s : \text{ There is no path } \nu \in H_k \text{ such that } \{\mu\} \subseteq \{\nu\}\}.$$

Hence, we can replace H_s in equality (1.28) by H_{sk}. To obtain the desired vector $x \in R^q_+$, we have to use equality (1.28) for each digraph $G_k \in \Lambda(G), k \neq s$. Let r denote the minimum of such a value r_{G_k}:

$$r = r_{G_{k^*}} = \min\{r_{G_k} : G_k \in \Lambda(G), k \neq s\}$$

and let $\nu^* \in H_{k^*}$ and $\mu^* \in H_{sk^*}$ be paths at which value $r_{G_{k^*}}$ has been reached:

$$r_{G_{k^*}} = r_{\nu^*} = \frac{l^p(\nu^*) - l^p(\mu^*)}{|\{\mu^*\} \cup \{\nu^*\}| - |\{\mu^*\} \cap \{\nu^*\}|}.$$

Taking into account (1.27), we note that, if $r_{\nu^*} \leq p_i$ for each $i \in \{\nu^*\} \backslash \{\mu^*\}$, vector $p(r) = p(r_{\nu^*})$ does not contain negative components, i.e., $p(r) \in R^q_+$. Due to the remark given after formula (1.27), we have obtained a lower bound for the stability radius:

$$\begin{aligned} &\widehat{\varrho}_s(p) \geq r \\ &= \min_{k=1,2,\ldots,\lambda;\, k\neq s} \quad \min_{\mu \in H_{sk}} \quad \max_{\nu \in H_k;\, l^p(\nu) > l^p_s} \quad \frac{l^p(\nu) - l^p(\mu)}{|\{\mu\} \cup \{\nu\}| - |\{\mu\} \cap \{\nu\}|}. \end{aligned} \tag{1.29}$$

The bound (1.29) is tight: If $\widehat{\varrho}_s(p) \leq p_i$ for each $i \in \{\nu^*\} \setminus \{\mu^*\}$, then $\widehat{\varrho}_s(p) = r$. For example, we have $\widehat{\varrho}_s(p) = r$ in (1.29) if $\widehat{\varrho}_s(p) \leq \min\{p_i : i \in Q\}$. To obtain the exact value of $\widehat{\varrho}_s(p)$ in the general case, we construct a vector $x = p^*(r) = (p^*_1(r), p^*_2(r), ..., p^*_q(r))$ with the components

$$p^*_i(r) = \begin{cases} p_i + r, & \text{if } i \in \{\mu\}, \\ \max\{0, p_i - r\}, & \text{if } i \in \{\nu\} \setminus \{\mu\}, \\ p_i, & \text{if } i \notin \{\mu\} \cup \{\nu\} \end{cases}$$

instead of vector $p(r)$ defined in (1.27). As it follows from the remark given after formula (1.27), such a vector $p^*(r)$ is the closest one to vector p among all vectors $x \in R^q_+$ which satisfy both conditions (a) and (b).

For calculating the maximal value r for vector $p^*(r)$, we can consider the operations of the set $\{\nu\} \setminus \{\mu\}$ in non-decreasing order of their processing times. Let $p^{\nu\mu}_{(0)}$ be equal to zero and let

$$(p^{\nu\mu}_{(0)}, p^{\nu\mu}_{(1)}, \ldots, p^{\nu\mu}_{(\omega_{\nu\mu})}) \tag{1.30}$$

denote a non-decreasing sequence of the processing times of the operations from the set $\{\nu\} \setminus \{\mu\}$, where $w_{\nu\mu} = |\{\nu\} \setminus \{\mu\}|$. Now, it is easy to obtain the following assertion.

Theorem 1.4 *If G_s is an optimal digraph for problem $\mathcal{G}//\mathcal{C}_{max}$, $s \in S^{\Phi}(p)$, $\Phi = \mathcal{C}_{max}$, and inequality $\widehat{\varrho}_s(p) < \infty$ holds, then*

$$\widehat{\varrho}_s(p) = \min_{k=1,2,\ldots,\lambda; k\neq s} \widehat{r}_{ks}, \tag{1.31}$$

where

$$\widehat{r}_{ks} = \min_{\mu\in H_{sk}} \quad \max_{\nu\in H_k,\ l^p(\nu)\geq l^p_s} \quad \max_{\beta=0,1,\ldots,w_{\nu\mu}} \frac{l^p(\nu) - l^p(\mu) - \sum_{\alpha=0}^{\beta} p^{\nu\mu}_{(\alpha)}}{|\{\mu\} \cup \{\nu\}| - |\{\mu\} \cap \{\nu\}| - \beta}. \tag{1.32}$$

Equalities (1.31) – (1.32) mean that one has to compare an optimal digraph $G_s(p)$ with all other feasible digraphs $G_k(p)$. Note that the formulas in Theorem 1.4 turn into $\widehat{\varrho}_s(p) = \infty$ if $H_{sk} = \emptyset$ for any $k = 1, 2, \ldots, \lambda,\ k \neq s$ (see Theorem 1.2). Moreover, if only a subset of the processing times (say, $P \subseteq \{p_1, p_2, \ldots, p_q\}$) can be changed but the other ones cannot be changed, formula (1.29) and Theorem 1.4 remain valid provided that the difference $|\{\mu\}\cup\{\nu\}|-|\{\mu\}\cap\{\nu\}|$ is replaced by $|\{\{\mu\}\cup\{\nu\}\}\cap P|-|\{\{\mu\}\cap\{\nu\}\}\cap P|$.

In Section 2.3, it will be shown how it is possible to restrict this enumeration and the comparisons (see Corollary 2.5 on page 116). We coded these formulas in Fortran-77 and tested them on randomly generated job shop problems $\mathcal{J}//\mathcal{C}_{max}$.

1.4. Mean Flow Time

In this section, we consider the stability radius $\overline{\varrho}_s(p)$ of an optimal schedule for problem $\mathcal{G}//\sum\mathcal{C}_i$ with the criterion $\sum\mathcal{C}_i$. If $\Phi = \sum\mathcal{C}_i$, conditions (1.8) and (1.9) for the general shop problem $\mathcal{G}//\Phi$ are converted into the following conditions (1.33) and (1.34):

$$\sum_{i=1}^{n} \max_{\mu\in H^i_s} l^p(\mu) = \min_{k=1,2,\ldots,\lambda} \sum_{i=1}^{n} \max_{\nu\in H^i_k} l^p(\nu), \tag{1.33}$$

$$\overline{\varrho}_s(p) = \inf\{d(p,x) \ :\ x \in R^q_+,\ \sum_{i=1}^{n} \max_{\mu\in H^i_s} l^x(\mu) > \min_{k=1,2,\ldots,\lambda;\ k\neq s} \sum_{i=1}^{n} \max_{\nu\in H^i_k} l^x(\nu)\}. \tag{1.34}$$

Here set $H^i_k,\ H^i_k \subseteq \tilde{H}^i_k$, is the set of all dominant paths in the digraph G_k ending in the *fixed* vertex $u \in Q^{J_i}$ (where $u = w(i) + n_i$ is the last

operation in the technological route of job J_i), and starting from *different* vertices $v \in Q^{J_r},\ r = 1, 2, \ldots, n$ (where $v = w(r) + 1$ is the first operation in the technological route of job J_r).

Obviously, the value C_i for a digraph $G_s(p)$ is equal to the largest weight of a path from the set H_s^i, and hence, to solve problem $\mathcal{G}//\sum \mathcal{C}_i$, it is sufficient to find a digraph $G_s(p)$ such that equality (1.33) holds.

Due to equality (1.34), to find the stability radius $\overline{\varrho}_s(p)$, it is sufficient to construct a vector $x \in R_+^q$ that satisfies the following three conditions.

(a') There exists a digraph $G_k(p) \in \Lambda(G), k \neq s$, such that

$$\sum_{i=1}^{n} \max_{\mu \in H_s^i} l^x(\mu) = \sum_{i=1}^{n} \max_{\nu \in H_k^i} l^x(\nu). \tag{1.35}$$

(b') For any given real $\epsilon > 0$, which may be as small as desired, there exists a vector $p^\epsilon \in R_+^q$ such that $d(x, p^\epsilon) = \epsilon$ and inequality

$$\sum_{i=1}^{n} \max_{\mu \in H_s^i} l^{p^\epsilon}(\mu) > \sum_{i=1}^{n} \max_{\nu \in H_k^i} l^{p^\epsilon}(\nu) \tag{1.36}$$

is satisfied for at least one digraph $G_k(p) \in \Lambda(G)$.

(c') The distance $d(p, x)$ achieves its minimal value among the distances between the vector p and the other vectors in the vector space R_+^q which satisfy both above conditions (a') and (b').

Similarly as in the previous section (see conditions (a), (b) and (c)), after having constructed such a vector $x \in R_+^q$, one can calculate the stability radius of digraph $G_s(p)$ as follows: $\overline{\varrho}_s(p) = d(p, x)$. Thus, due to (1.35) and (1.36), the calculation of the stability radius may be reduced again to an extremal problem on the set of weighted digraphs $\Lambda(G)$. However, in this case we are forced to consider sets of representatives of the family of sets $H_k^i,\ 1 \leq i \leq n$, which may be defined as follows.

Let Ω_k^u be a set of representatives of the family of sets $(H_k^i)_{1 \leq i \leq n}$. More precisely, the set Ω_k^u includes exactly one path from each set $H_k^i, 1 \leq i \leq n$. Since $H_k^i \cap H_k^j = \emptyset$ for each pair of different jobs J_i and J_j, we have $|\Omega_k^u| = n$ and there exist $\omega_k = \prod_{i=1}^{n} |H_k^i|$ different sets of representatives for each digraph G_k, namely: $\Omega_k^1, \Omega_k^2, \ldots, \Omega_k^{\omega_k}$. For each set Ω_k^u, we can calculate the integer vector $n(\Omega_k^u) = (n_1(\Omega_k^u), n_2(\Omega_k^u), \ldots, n_q(\Omega_k^u))$, where $n_j(\Omega_k^u), j \in Q = \{1, 2, ..., q\}$, is equal to the number of paths in Ω_k^u which include vertex j. Since a path $\nu \in H_k^i$ includes vertex $j \in Q$ at most once, the value $n_j(\Omega_k^u)$ is equal to the number of copies of vertex j contained in the multiset $\{\{\nu\} : \nu \in \Omega_k^u\}$. Similarly to the proof of Theorem 1.4, one can find a vector x satisfying conditions (a') and (b') in the form $x(r) = (x_1(r), x_2(r), \ldots,$

$x_q(r))$ with the components $x_i(r)$ from the set $\{p_i, p_i + r, p_i - r\}$ on the basis of a direct comparison of the set $\Omega_s^u \in \Omega_{s,k}$ of representatives of the family of sets $(H_s^i)_{1\le i\le n}$, and the set Ω_k^u of representatives of the family of sets $(H_k^i)_{1\le i\le n}$, where $k = 1, 2, \ldots, \lambda,\ k \neq s$, and

$$\Omega_{sk} = \{\Omega_s^v : \text{ There does not exist a set } \Omega_k^u \text{ such that } n_i(\Omega_s^v) \le n_i(\Omega_k^u) \text{ for each } i = 1, 2, \ldots, q\}.$$

As a result, the following lower bound for the stability radius has been obtained:

$$\overline{\varrho}_s(p) \ge r = \min_{k=1,2,\ldots,\lambda;\ k\neq s} \min_{\Omega_s^v\in\Omega_{s,k}} \max_{u\in\{1,2,\ldots,\omega_k\}} r_{\Omega_k^u,\Omega_s^v}, \tag{1.37}$$

where

$$r_{\Omega_k^u,\Omega_s^v} = \frac{\sum_{\nu\in\Omega_k^u} l^p(\nu) - \sum_{\mu\in\Omega_s^v} l^p(\mu)}{\sum_{i=1}^q |n_i(\Omega_k^u) - n_i(\Omega_s^v)|}.$$

This bound is tight. Indeed, we obtain $\overline{\varrho}_s(p) = r$ in (1.37) if $r \le \min\{p_i : i \in Q\}$. It is easy to see that it may happen that the above vector $x(r) \in R^q$ does not belong to set R_+^q since subtracting value r from some component of vector p may result in a negative number. To obtain the exact value of $\overline{\varrho}_s(p)$, one can use the vector $x^*(r) = (x_1^*(r), x_2^*(r), \ldots, x_q^*(r))$ with

$$x_i^*(r) = \begin{cases} p_i + r, & \text{if } n_i(\Omega_k^u) < n_i(\Omega_s^v), \\ \max\{0, p_i - r\}, & \text{if } n_i(\Omega_k^u) > n_i(\Omega_s^v), \\ p_i, & \text{if } n_i(\Omega_k^u) = n_i(\Omega_s^v). \end{cases}$$

In contrast to vector $x(r)$, vector $x^*(r)$ necessarily has no negative components. Let the set of operations Q be ordered in the following way:

$$i_1, i_2, \ldots, i_m, i_{m+1}, \ldots, i_q, \tag{1.38}$$

where $n_{i_\alpha}(\Omega_k^u) \le n_{i_\alpha}(\Omega_s^v)$ for each $\alpha = 1, 2, \ldots, m$ and $n_{i_\alpha}(\Omega_k^u) > n_{i_\alpha}(\Omega_s^v)$ for each $\alpha = m+1, m+2, \ldots, q$. For sequence (1.38), the inequalities $p_{i_{m+1}} \le p_{i_{m+2}} \le \ldots \le p_{i_q}$ have to be satisfied. Using the sequence of operations (1.38), it is easy to derive the following formula for calculating $\overline{\varrho}_s(p)$.

Theorem 1.5 *If G_s is an optimal digraph for problem $\mathcal{G}//\sum \mathcal{C}_i$, then*

$$\overline{\varrho}_s(p) = \min_{k=1,2,\ldots,\lambda;k\neq s} \overline{r}_{ks}, \tag{1.39}$$

where

$$\overline{r}_{ks} = \min_{\Omega_s^v\in\Omega_{s,k}} \max_{u=1,2,\ldots,\omega_k} \max_{\beta=0,1,\ldots,q-m} \frac{\sum_{\alpha=1}^{m+\beta} p_{i_\alpha} |n_{i_\alpha}(\Omega_k^u) - n_{i_\alpha}(\Omega_s^v)|}{\sum_{\alpha=1}^{m+\beta} |n_{i_\alpha}(\Omega_k^u) - n_{i_\alpha}(\Omega_s^v)|}. \tag{1.40}$$

If only a subset of the processing times can be changed but the other ones cannot be changed, formulas similar to (1.39) and (1.40) can be derived (see the remark after Theorem 1.4).

Zero Stability Radius

Next, we consider the case of $\overline{\varrho}_s(p) = 0$. Similarly to the notions of a critical path and a critical weight, which is important for problem $\mathcal{G}//\mathcal{C}_{max}$ (see Section 1.3), we introduce the notions of a critical set of paths $\Omega_k^{u^*}$ and a critical sum of weights for problem $\mathcal{G}//\sum\mathcal{C}_i$. The set $\Omega_k^{u^*}, u^* \in \{1, 2, \ldots, \omega_k\}$, is called a *critical set* if the value of the objective function

$$L_k^p = \max_{u\in\{1,2,\ldots,\omega_k\}} \sum_{\nu\in\Omega_k^u} l^p(\nu) \tag{1.41}$$

for the weighted digraph $G_k(p)$ is reached on this set:

$$\sum_{\nu\in\Omega_k^{u^*}} l^p(\nu) = \max_{u\in\{1,2,\ldots,\omega_k\}} \sum_{\nu\in\Omega_k^u} l^p(\nu) = L_k^p.$$

The value L_k^p defined in (1.41) is called a *critical sum of weights* for digraph $G_k(p)$. Obviously, a critical set $\Omega_k^{u^*}$ may include a path $\nu \in H_k^i$, $i = 1, 2, \ldots, n$, if and only if $l^p(\nu) = \max_{\mu\in H_k^i} l^p(\mu)$ and so for different vectors $p \in R_+^q$ of the processing times, different sets Ω_k^u, $u \in \{1, 2, \ldots, \omega_k\}$, may be critical. Let $\Omega_k(p)$ denote the set of all critical sets $\Omega_k^{u^*}$ of digraph $G_k(p)$ for the vector $p = (p_1, p_2, \ldots, p_q) \in R_+^q$ of operation processing times and let Ω_k denote the set $\{\Omega_k^u : u = 1, 2, \ldots, \omega_k\}$. To prove necessary and sufficient conditions for $\overline{\varrho}_s(p) = 0$, we need the following auxiliary claim.

Lemma 1.4 *There exists a real number $\epsilon > 0$ such that the set $\Omega_k \backslash \Omega_k(p)$ contains no critical set of digraph $G_k(p) \in \Lambda(G)$ for any vector $p' \in O_\epsilon(p) \cap R_+^q$ of the processing times, i.e., $\Omega_k(p') \subseteq \Omega_k(p)$.*

Proof. After having calculated the value

$$\epsilon_k = \frac{1}{2qn} \min\left\{ L_k^p - \sum_{\nu\in\Omega_k^u} l^p(\nu) \ : \ \Omega_k^u \in \Omega_k \backslash \Omega_k(p) \right\}, \tag{1.42}$$

one can verify that for any real ϵ, which satisfies the inequalities $0 < \epsilon < \epsilon_k$, the difference in the right-hand side of equality (1.42) remains positive when vector p is replaced by any vector $p' \in O_\epsilon(p) \cap R_+^q$. Indeed, for any $u \in \{1, 2, \ldots, \omega_k\}$, the cardinality of set Ω_k^u may be at most equal to qn. Thus, it is not possible to 'overcome' the difference $L_k^p - \sum_{\nu\in\Omega_k^u} l^p(\nu)$ using any vector p' if inequality $d(p, p') < \epsilon_k$ holds.

$\diamond$

Next, we prove necessary and sufficient conditions for equality $\overline{\varrho}_s(p) = 0$.

Theorem 1.6 *Let G_s be an optimal digraph for problem $\mathcal{G}//\sum\mathcal{C}_i$ with positive processing times $p_i > 0$ of all operations $i \in Q$. The equality $\overline{\varrho}_s(p) = 0$ holds if and only if the following three conditions hold:*
1) There exists another optimal schedule $k \in S^{\Phi}(p)$, $\Phi = \sum\mathcal{C}_i$, $k \neq s$;
2) There exists a set $\Omega_s^{v^*} \in \Omega_s(p)$ such that for any set $\Omega_k^{u^*} \in \Omega_k(p)$, there exists an operation $i \in Q$ for which condition

$$n_i(\Omega_s^{v^*}) \geq n_i(\Omega_k^u),\ \Omega_k^u \in \Omega_k(p), \tag{1.43}$$

holds (or condition

$$n_i(\Omega_s^{v^*}) \leq n_i(\Omega_k^u),\ \Omega_k^u \in \Omega_k(p), \tag{1.44}$$

holds);
3) Inequality (1.43) (or inequality (1.44), respectively) is satisfied as a strict one for the set $\Omega_k^{u^*}$.

Proof. We prove *necessity* by contradiction. Assume that $\overline{\varrho}_s(p) = 0$ but the conditions of the theorem are not satisfied. We consider the three cases (j), (jj) and (jjj) of violating these conditions.

(j) Assume that there does not exist another optimal schedule, i.e., we have $S^{\Phi}(p) = \{s\}$ with $\Phi = \sum\mathcal{C}_i$. Then we consider a real number ϵ such that inequalities

$$0 < \epsilon < \frac{1}{2qn} \min_{t \neq s}\{L_t^p - L_s^p\}$$

hold. Similarly to the proof of Lemma 1.4, we can show that digraph G_s remains optimal for any vector $p^0 = (p_1^0, p_2^0, \ldots, p_q^0) \in R_+^q$ of the processing times provided that $d(p, p^0) \leq \epsilon$. Therefore, we obtain the inequalities $\overline{\varrho}_s(p) \geq \epsilon > 0$ which contradict the assumption $\overline{\varrho}_s(p) = 0$.

(jj) We assume that $|S^{\Phi}(p)| > 1$ and for any optimal schedule $k \in S^{\Phi}(p)$ with $k \neq s$ and for any set $\Omega_s^v \in \Omega_s(p)$, there exists a set $\Omega_k^{u^*} \in \Omega_k(p)$ such that $n_i(\Omega_s^v) = n_i(\Omega_k^{u^*})$ for any operation $i \in Q$.

In this case, we can take any ϵ that satisfies the inequalities

$$0 < \epsilon < \min\left\{\epsilon_s, \epsilon_k, \frac{1}{2qn} \min_{t \notin \phi^{\Sigma}(p)}\{L_t^p - L_s^p\}\right\}. \tag{1.45}$$

From Lemma 1.4, due to inequality $\epsilon < \epsilon_s$, we get that equality

$$L_s^{p^0} = \max_{\Omega_s^v \in \Omega_s(p)} \sum_{\mu \in \Omega_s^v} l^{p^0}(\mu) \tag{1.46}$$

holds for any vector $p^0 \in O_\epsilon(p) \cap R^q_+$. Since there exists a set $\Omega^{u^*}_k \in \Omega_k(p)$ for any set $\Omega^v_s \in \Omega_s(p)$ and any $k \in S^\Phi(p),\ k \neq s$, such that $n_i(\Omega^v_s) = n_i(\Omega^{u^*}_k),\ i \in Q$, we obtain the inequality

$$\max_{\Omega^v_s \in \Omega_s(p)} \sum_{\mu \in \Omega^v_s} l^{p^0}(\mu) \leq \max_{\Omega^u_k \in \Omega_k(p)} \sum_{\nu \in \Omega^{u^*}_k} l^{p^0}(\nu),$$

because of $\epsilon < \epsilon_s$ and $\epsilon < \epsilon_k$. Therefore, due to (1.46), we have

$$L^{p^0}_s \leq \max_{\Omega^u_k \in \Omega_k(p)} \sum_{\nu \in \Omega^{u^*}_k} l^{p^0}(\nu) \tag{1.47}$$

for any optimal schedule $k \in S^\Phi(p),\ k \neq s$. Since $\epsilon < \frac{1}{2qn} \min_{t \notin S^\Phi(p)}\{L^p_t - L^p_s\}$, condition $t \notin S^\Phi(p)$ implies $t \notin S^\Phi(p^0)$. Thus, taking into account (1.45) and the latter implication, we conclude that $s \in S^\Phi(p^0)$ for any vector $p^0 \in R^q_+$ provided that $d(p, p^0) \leq \epsilon$. Consequently, we obtain the inequalities $\overline{\varrho}_s(p) \geq \epsilon > 0$, which contradict the assumption $\overline{\varrho}_s(p) = 0$.

(jjj) We assume that $|S^\Phi(p)| > 1$ and for any optimal schedule $k \in S^\Phi(p), k \neq s$, and for any set $\Omega^v_s \in \Omega_s(p)$, there exists a set $\Omega^{u^*}_k \in \Omega_k(p)$ such that for any operation $i \in Q$ with $n_i(\Omega^v_s) > n_i(\Omega^{u^*}_k)$, there exists a set $\Omega^{u^0}_k \in \Omega_k(p)$ such that $n_i(\Omega^v_s) < n_i(\Omega^{u^0}_k)$. Arguing in the same way as in case (jj), we can show that $\overline{\varrho}_s(p) \geq \epsilon > 0$, where ϵ is as in (1.45) since for any vector $p^0 \in O_\epsilon(p) \cap R^q_+$, the value $\sum_{\mu \in \Omega^{v^*}_s} l^{p^0}(\mu)$ is less than or equal to the value $\sum_{\nu \in \Omega^{u^*}_k} l^{p^0}(\nu)$ or value $\sum_{\nu \in \Omega^{u^0}_k} l^{p^0}(\nu)$.

Sufficiency. We show that, if the conditions of Theorem 1.6 are satisfied, then $\overline{\varrho}_s(p) < \epsilon$ for any given $\epsilon > 0$.

We construct a vector $p^* = (p^*_1, p^*_2, \ldots, p^*_q) \in R^q_+$ with the components $p^*_i \in \{p_i, p_i + \epsilon^*, p_i - \epsilon^*\}$, where $\epsilon^* = \min\{\epsilon_k, \epsilon, \min_{i \in Q} p_i\}$ using the following rule: For each $\Omega^{u^*}_k \in \Omega_k(p)$ mentioned in Theorem 1.6, we set $p^*_i = p_i + \epsilon^*$, if inequalities (1.43) hold, or we set $p^*_i = p_i - \epsilon^*$, if inequalities (1.44) hold. Note that $\epsilon^* > 0$ since $p_i > 0, i \in Q$.

After changing at most $|\Omega_k(p)|$ components of vector p according to the above rule, we obtain a vector p^* of processing times for which inequality

$$\sum_{\mu \in \Omega^{v^*}_s} l^{p^*}(\mu) > \sum_{\nu \in \Omega^{u^*}_k} l^{p^*}(\nu)$$

holds for each set $\Omega^{u^*}_k \in \Omega_k(p)$. Due to inequality $\epsilon^* \leq \min_{i \in Q} p_i$, we obtain $p^* \in R^q_+$. Since $\epsilon^* \leq \epsilon_k$, we have

$$\begin{aligned} L^{p^*}_k &= \max_{u \in \{1,2,\ldots,\omega_k\}} \sum_{\nu \in \Omega^u_k} l^{p^*}(\nu) \\ &= \max_{\Omega^u_k \in \Omega_k(p)} \sum_{\nu \in \Omega^u_k} l^{p^*}(\nu) = \sum_{\nu \in \Omega^{u^*}_k} l^{p^*}(\nu) < \sum_{\mu \in \Omega^{v^*}_s} l^{p^*}(\mu) \leq L^{p^*}_s. \end{aligned}$$

We conclude that $s \notin S^{\Phi}(p^*)$ with $d(p, p^*) = \epsilon^*$ which implies $\overline{\varrho}_s(p) < \epsilon^* \leq \epsilon$.
$\diamond$

Theorem 1.6 directly implies the following assertion.

Corollary 1.5 *If $s \in S$ is a unique optimal schedule for problem $\mathcal{G}//\sum \mathcal{C}_i$, then $\overline{\varrho}_s(p) > 0$.*

It is easy to prove the following upper bound for the stability radius of an optimal schedule for problem $\mathcal{G}//\sum \mathcal{C}_i$.

Theorem 1.7 *If $s \in S$ is an optimal schedule for problem $\mathcal{G}//\sum \mathcal{C}_i$ with $\lambda > 1$ and $p_i > 0$ for at least one operation $i \in Q$, then $\overline{\varrho}_s(p) \leq \max_{i \in Q} p_i$.*

PROOF. We consider the vector $p^0 \in R^q_+$ with zero components: $p^0_i = 0$ for each operation $i \in Q$. For this vector of processing times, each feasible digraph $G_t \in \Lambda(G)$ is optimal and each set of representatives Ω^u_t is critical. We can take a schedule $k \in S^{\Phi}(p^0)$ which has only one arc $(j, i) \in E_k$ different from the arcs in set E_s, i.e., $(i, j) \in E_s$ and $E_s \backslash \{(i, j)\} = E_k \backslash \{(j, i)\}$. It is easy to see that there exist sets $\Omega^v_s \in \Omega_s(p)$ and $\Omega^u_k \in \Omega_k(p)$ such that $n_i(\Omega^v_s) > n_i(\Omega^u_k)$. Setting $p^{\epsilon}_i = \epsilon > 0$ and $p^{\epsilon}_l = 0$ for each operation $l \in Q \backslash \{i\}$, we obtain $s \notin S^{\Phi}(p^{\epsilon})$ and $d(p^{\epsilon}, p) < \max\{p_i : i \in Q\}$.
$\diamond$

Remark 1.2 As follows from Theorem 1.7, problem $\mathcal{J}//\sum \mathcal{C}_i$ with $\lambda > 1$ cannot have an optimal schedule with an infinitely large stability radius in contrast to problem $\mathcal{J}//\mathcal{C}_{max}$ (see Section 1.3).

Note that all results presented in this section and in Section 1.3 remain valid for any general shop scheduling problem. However, we use the partition of the set of operations Q into n chains $Q^{(i)}, i = 1, 2, \ldots, n$, (which is necessary for the job shop and flow shop but is not necessary for the general shop) for a better presentation of the results.

In scheduling theory, the **open shop** problem is also considered when the technological routes for processing the jobs are not fixed before scheduling, i.e., for each job $J_i \in J$ only the set of operations Q^{J_i} is given but the order of these operations is not fixed before scheduling. While solving an open shop problem $\mathcal{O}//\Phi$, it is necessary to find the optimal orders of the operations Q^{J_i} for each job $J_i \in J$ along with the optimal orders of the operations on each machine $M_k \in M$. Similarly to the flow shop problem, each job $J_i \in J$ has to be processed by each machine exactly once. The open shop problem is a special case of the general shop problem and therefore, all

results obtained above for the stability analysis remain valid for the open shop as well. The classical job shop is often considered in scheduling theory for which each job has to be processed exactly once on each machine. For our considerations, this restriction is not important. In fact, we consider the job shop problem $\mathcal{J}m//\Phi$ with *recirculation*, which may occur when a job may visit a machine from the set M more than once.

1.5. Calculation of the Stability Radius

This section is devoted to the calculation of the stability radius of an optimal schedule for a job shop problem when the objective is to minimize mean or maximum flow times. The used approach may be regarded as an *a posteriori* analysis. We investigate the influence of errors and possible changes of the processing times on the property of a schedule to be optimal. To this end, extensive numerical experiments with randomly generated job shop problems are performed. Due to the developed software, we have the possibility to compare the values of the stability radii, the numbers of optimal schedules and some other 'numbers' for the two criteria $\mathcal{C}_{max}$ and $\sum \mathcal{C}_i$. The main question we try to answer is how large the stability radius is, on average, for randomly generated job shop problems. The formulas for calculating the stability radii $\hat{\varrho}_s(p)$ and $\overline{\varrho}_s(p)$ of an optimal digraph $G_s(p)$, derived in [44, 311, 339] (see Chapter 1), were coded in Fortran-77. Due to these formulas (1.31), (1.32) and (1.39), (1.40), the calculation of the stability radii based on a direct comparison of the paths of an optimal digraph G_s and of each feasible digraph $G_k \in \Lambda(G), k \neq s$, for C_{max} and subsets of paths of G_s and of $G_k \in \Lambda(G), k \neq s$, for $\sum C_i$ is very complicated and time-consuming. Nevertheless, such an 'unpractical' calculation for sample problems allows us to derive some properties of job shop problems, which may be used in practically efficient methods for determining lower and/or upper bounds for the stability radii. Next, we present the formal algorithm for calculating the stability radius $\hat{\varrho}_s(p)$ on the basis of the coded formulas (1.31) and (1.32). We calculate the set of stability radii $\widehat{\mathcal{R}} = \{\hat{\varrho}_1(p), \hat{\varrho}_2(p), \ldots, \hat{\varrho}_{opt}(p)\}$ for the set of all optimal digraphs $G_1(p), G_2(p), \ldots, G_{opt}(p)$ from the set $\Lambda(G)$ generated from a weighted mixed graph (Q, A, E). Here *opt* indicates the number of optimal schedules.

Algorithm $RAD\_\hat{\varrho}_s(p)$

Input: Mixed graph (Q^J, A^J, E^J) with the vector $p \in R^q_+$

of job processing times.

Output: Set $\widehat{\mathcal{R}}$ of the stability radii for all optimal digraphs.

Step 1: Construct the set of feasible digraphs $\{G_1(p), G_2(p), \ldots, G_{opt}(p), \ldots, G_\lambda(p)\}$ generated from (Q^J, A^J, E^J) and index them in non-decreasing order of their makespans: $l^p_1 = l^p_2 = \ldots = l^p_{opt} < l^p_{opt+1} \leq l^p_{opt+2} \leq \ldots \leq l^p_\lambda$. Set $\widehat{\mathcal{R}} = \emptyset$. **IF** $opt = 1$ **THEN** $s = 1$ **GOTO** *Step 4*.

Step 2: **FOR** $s = 1$ **TO** *opt* **DO**
BEGIN

Step 3: **IF** there exists a path $\mu^* \in H_s(p)$ such that for digraph $G_k(p), k \neq s, k \leq opt$, $l^p_s = l^p_k$, there does not exist a path $\nu^* \in H_k(p)$ with $[\mu^*] \subseteq [\nu^*]$
THEN $\hat{\varrho}_s(p) = 0$; $\widehat{\mathcal{R}} := \widehat{\mathcal{R}} \cup \{\hat{\varrho}_s(p)\}$
IF $s < opt$ **THEN GOTO** *Step 2* **ELSE STOP**
ELSE

Step 4: $\hat{\varrho}_s(p) := \infty$
IF the conditions of Theorem 1.3 hold for digraph $G_s(p)$
THEN $\widehat{\mathcal{R}} := \widehat{\mathcal{R}} \cup \{\hat{\varrho}_s(p)\}$
IF $s < opt$ **THEN GOTO** *Step 2* **ELSE STOP**
ELSE

Step 5: **FOR** $k = 1, k \neq s$ **TO** λ **DO**
BEGIN

Step 6: Construct the set $H_{sk} = \{\mu \in H_s :$ There is no path $\nu \in H_k$ such that $[\mu] \subseteq [\nu]\}$.
IF $H_{sk} = \emptyset$

Step 7: **IF** $k = \lambda$
IF $H_{st} = \emptyset$ for each digraph $G_t(p), t \neq s$, $t \in \{1, 2, \ldots, \lambda\}$
THEN $\widehat{\mathcal{R}} := \widehat{\mathcal{R}} \cup \{\hat{\varrho}_s(p)\}$
IF $s < opt$ **THEN GOTO** *Step 2* **ELSE STOP**
ELSE
ELSE GOTO *Step 5*
ELSE $r_k = 0$

Step 8: **FOR** $\mu \in H_{sk}$ **DO**
BEGIN

Step 9: **FOR** $\nu \in H_k : \ l^p(\nu) \geq l_s^p$ **DO**
BEGIN

Step 10: Set $r_\beta = 0$.
Construct a sequence $(p^{\nu\mu}_{(0)}, p^{\nu\mu}_{(1)}, \ldots, p^{\nu\mu}_{(w_{\nu\mu})})$, where $p^{\nu\mu}_{(0)} = 0$ and $(p^{\nu\mu}_{(1)}, p^{\nu\mu}_{(2)}, \ldots, p^{\nu\mu}_{(w_{\nu\mu})})$ is a non-decreasing sequence of the processing times of the operations from the set $[\nu] \setminus [\mu]$ with $w_{\nu\mu} = |[\nu] \setminus [\mu]|$.
FOR $\beta = 0$ **TO** $\omega_{\nu\mu}$ **DO**
BEGIN
$r_\beta = \max\{r_\beta, \frac{l^p(\nu) - l^p(\mu) - \sum_{\alpha=0}^{\beta} p^{\nu\mu}_{(\alpha)}}{|[\mu]+[\nu]|-\beta}\}$
END
END
$r_k := \max\{r_k, r_\beta\}$
END

Step 11: $\widehat{\varrho}_s(p) := \min\{\widehat{\varrho}_s(p), r_k\}$

Step 12: **FOR** $k := k + 1$ **TO** $\lambda + 1$ **Do**
BEGIN
IF $\widehat{\varrho}_s(p) > \frac{l_k^p - l_s^p}{q}$ **THEN GOTO** *Step 5*
END
$\widehat{\mathcal{R}} := \widehat{\mathcal{R}} \cup \{\widehat{\varrho}_s(p)\}$
IF $s <$ *opt* **THEN GOTO** *Step 2* **ELSE STOP**
END
END STOP

Computational results for randomly generated job shop problems are presented in this chapter. The stability radii have been calculated for more than 10,000 randomly generated job shop problems.

At the worst, the calculation of $\widehat{\varrho}_s(p)$ (in just the same way as the calculation of $\overline{\varrho}_s(p)$) implies not only to have an optimal digraph $G_s(p)$, which is already an NP-hard problem, but to construct all feasible digraphs $G_1(p), G_2(p), \ldots, G_\lambda(p)$ (see Step 1) and for each of them, which has to be compared with the optimal digraph, Algorithm $RAD\_\widehat{\varrho}_s(p)$ finds all dominant paths (see Step 6) introduced in Definition 1.3. We can avoid such a time-consuming comparison in the two following cases.

First, if there are two or more optimal digraphs it is possible that the stability radius of one of them or the radii of both are equal to zero (see Theorem 1.1). In Step 3, we check such a condition for $\widehat{\varrho}_s(p) = 0$. Second,

there are two cases of an infinitely large stability radius $\widehat{\varrho}_s(p) = \infty$. One of them follows from the graph construction: There is identified a problem class whose optimal solutions are implied only by the given structural input data and even independently from the numerical input data (see Theorem 1.3). Thus, the necessary and sufficient conditions of Theorem 1.3 for an infinitely large stability radius $\mathcal{J}//\mathcal{C}_{max}$ can be verified in polynomial time $O(q^2)$ in Step 4 (q is the number of operations, $q = |Q|$). The second condition for an infinitely large stability radius follows directly from Theorem 1.2 and it is checked in Step 7. More exactly, from Theorem 1.2 and the definition of the set H_{sk} it follows that, if $H_{sk} = \emptyset$ for each feasible digraph $G_k(p), k \neq s$, then $\widehat{\varrho}_s(p) = \infty$. From Step 7 to Step 11, we calculate the value $\widehat{r}_{ks}$ according to formula (1.32). In Algorithm $RAD\_\widehat{\varrho}_s(p)$, the value

$$\widehat{r}_{ks} := \widehat{\varrho}_s(p) := \min\{\widehat{\varrho}_s(p), r_k\}$$

is finally defined in Step 11. To restrict the number of digraphs G_k with which an optimal digraph has to be compared, Algorithm $RAD\_\widehat{\varrho}_s(p)$ uses the bounds (2.34) in Step 12. (In Chapter 2, these bounds will be proven for the more general case of the relative stability radius.) Note that Steps 5 and 7 are rather complicated. So, for scheduling problems with a small size of the input data the program starts with generating all feasible digraphs $\Lambda(G)$ (see Step 1). Algorithm $RAD\_\widehat{\varrho}_s(p)$ compares each optimal digraph $G_s(p), 1 \leq s \leq opt$, consecutively with the digraphs $G_k(p), k \neq s$, from the set $\Lambda(G)$. After that, Theorem 1.4 is used for calculating the stability radius $\widehat{\varrho}_s(p)$. Using Algorithm $RAD\_\widehat{\varrho}_s(p)$, we construct a set $\widehat{\mathcal{R}} = \{\widehat{\varrho}_1(p), \widehat{\varrho}_2(p), \ldots, \widehat{\varrho}_{opt}(p)\}$ of the stability radii. As it will follow from Remark 2.4, this algorithm is more effective than Algorithm $SOL\_\mathcal{C}_{max}(2)$. So, if $\widehat{\mathcal{R}}$ is not a single-element set, then a decision-maker can use one of the optimal digraphs $G_s(p), s = 1, 2, \ldots, opt$, which is more stable, i.e., a schedule with the largest value of the stability radius $\widehat{\varrho}_1(p) \in \widehat{\mathcal{R}}$. Next, we present the formal algorithm for the calculation of the stability radii $\overline{\varrho}_s(p)$, which uses the formulas (1.39) and (1.40) derived for the job shop problem.

Algorithm $RAD\_\overline{\varrho}_s(p)$

Input: Mixed graph (Q^J, A^J, E^J) with the vector $p \in R^q_+$ of job processing times.

Output: Set $\overline{\mathcal{R}} = \{\overline{\varrho}_1(p), \overline{\varrho}_2(p), \ldots, \overline{\varrho}_{opt}(p)\}$ of the stability radii for all optimal digraphs.

Step 1: Construct the set of feasible digraphs $\{G_1(p), G_2(p), \ldots, G_{opt}(p), \ldots, G_\lambda(p)\}$ generated from (Q^J, A^J, E^J) and index them in non-decreasing order of the objective function values: $L_1^p = L_2^p = \ldots = L_{opt}^p < L_{opt+1}^p \leq L_{opt+2}^p \leq \ldots \leq L_\lambda^p$. Set $\overline{\mathcal{R}} = \emptyset$. **IF** $opt = 1$ **THEN** set $s = 1$ **GOTO** *Step 4*.

Step 2: **FOR** $s = 1$ **TO** *opt* **DO**
BEGIN

Step 3: **IF** there exists a set $\Omega_s^{v^*} \in \Omega_s(p)$ such that for any set $\Omega_k^{u^*} \in \Omega_k(p)$, there exists an operation $O_{ij} \in Q^J$ such that condition $n_{ij}(\Omega_s^{v^*}) \geq n_{ij}(\Omega_k^{u^*})$ (or condition $n_{ij}(\Omega_s^{v^*}) \leq n_{ij}(\Omega_k^{u^*})$) holds and this inequality has the sign $>$ (or $<$) for at least one set $\Omega_k^{u^0} \in \Omega_k(p)$ **THEN** set $\overline{\varrho}_s(p) = 0$ and $\overline{\mathcal{R}} := \overline{\mathcal{R}} \cup \{\overline{\varrho}_s(p)\}$
IF $s <$ *opt* **THEN GOTO** *Step 2* **ELSE STOP**
ELSE

Step 4: Set $\overline{\varrho}_s(p) := \infty$.

Step 5: **FOR** $k = 1, k \neq s$ **TO** λ **DO**
BEGIN

Step 6: Construct the set $\Omega_{sk} = \{\Omega_s^v :$ There does not exist a set Ω_k^u such that $n_{ij}(\Omega_s^v) \leq n_{ij}(\Omega_k^u)$ for each $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n_i\}$. Set $r_k = 0$.

Step 7: **FOR** $v = 1$ **TO** ω_s **DO**
BEGIN

Step 8: **FOR** each $\Omega_k^u \in \Omega_k, u = 1, 2, \ldots, \omega_k$, with $\sum_{\nu \in \Omega_k^u} l^p(\nu) \geq L_s^p$
DO
BEGIN

Step 9: Set $r_\beta = 0$. Order the set of operations Q^J: $O_{ij_{(1)}}, O_{ij_{(2)}}, \ldots, O_{ij_{(m)}}, O_{ij_{(m+1)}}, \ldots, O_{ij_{(q)}}$, where for each $\alpha = 1, 2, \ldots, m$ inequality $n_{ij_{(\alpha)}}(\Omega_k^u) \leq n_{ij_{(\alpha)}}(\Omega_s^v)$ holds and for each $\alpha \in \{m+1, m+2, \ldots, q\}$ inequalities $n_{ij_\alpha}(\Omega_k^u) > n_{ij_\alpha}(\Omega_s^v)$; $p_{ij_{(m+1)}} \geq p_{ij_{(m+2)}} \geq \ldots \geq p_{ij_{(q)}}$ hold.
FOR $\beta = 0$ **TO** $q - m$ **DO**
BEGIN

$$r_\beta = \max\left\{r_\beta, \frac{\sum_{\alpha=1}^{m+\beta} p_{ij_{(\alpha)}}(n_{ij_{(\alpha)}}(\Omega_k^u) - n_{ij_{(\alpha)}}(\Omega_s^v))}{\sum_{\alpha=1}^{m+\beta} |n_{ij_{(\alpha)}}(\Omega_k^u) - n_{ij_{(\alpha)}}(\Omega_s^v)|}\right\}$$

END
END
Set $r_k := \max\{r_k, r_\beta\}$
END

Step 10: Set $\overline{\varrho}_s(p) := \min\{\overline{\varrho}_s(p), r_k\}$.

Step 11: **FOR** $k := k+1$ **TO** $\lambda+1$ **DO**
BEGIN
IF $\overline{\varrho}_s(p) > \frac{L_k^p - L_s^p}{nq-n}$ **THEN GOTO** *Step 5*
END
$\overline{\mathcal{R}} := \overline{\mathcal{R}} \cup \{\overline{\varrho}_s(p)\}$
IF $s <$ *opt* **THEN GOTO** *Step 2* **ELSE STOP**
END
END STOP

If there exist at least two optimal schedules, i.e., if $opt > 1$, we verify in Step 3 the condition for a zero stability radius on the basis of Theorem 1.6. In Step 4, we set $\overline{\varrho}_s(p) := \infty$ (note that $\overline{\varrho}_s(p) < \infty$ due to Theorem 1.7 and Remark 1.2). Theorem 1.5 is used for the calculation of the stability radius $\overline{\varrho}_s(p)$, $0 < \overline{\varrho}_s(p) < \infty$, for each optimal digraph $G_s, s = 1, 2, \ldots, opt$, (see Steps 6 - 10). As it will be shown in Lemma 2.8 in Chapter 2, we can reduce the set of digraphs in the considerations in Step 11.

Both above formal algorithms were coded in Fortran-77. So, for a small problem size the program starts with generating all feasible digraphs and for each of them, which has to be compared with the optimal digraph, it finds dominant paths (see Definition 1.3). Then formulas (1.31) and (1.32) are used for calculating $\hat{\varrho}_s(p)$ and formulas (1.39) and (1.40) from Section 1.4 are used for calculating $\overline{\varrho}_s(p)$. To restrict the number of digraphs G_k with which an optimal digraph has to be compared, one can use the bound (2.63) (see Chapter 2) for the mean flow time criterion.

Note that the software developed is rather general. In principle, it allows us to calculate the exact or approximate values of $\hat{\varrho}_s(p)$ and $\overline{\varrho}_s(p)$ for most scheduling problems (since there exists a possibility to represent them as extremal problems on a mixed graph, see Section 1.1). The only 'theoretical' requirement for such problems is the prohibition of preemptions of operations (see Assumption 2). However, in the simulation study we are forced to take into account also 'practical' requirements: The running time and the memory of the computers. Remind that the most critical parameter of the problem under consideration is the number of edges in the mixed graph G because the whole number of feasible (without a circuit) and infeasible

(with circuit) digraphs generated from G is equal to $2^{|E|}$. Moreover, for each feasible digraph G_k, we have to find all dominant paths for C_{max} and (what is essentially larger) all subsets of the set of dominant paths for $\sum C_i$.

1.6. Experimental Design and Results

In this section, computations were restricted to job shop problems. We considered three different levels of the simulation study in dependence on running time and memory limits. The stability region of an optimal digraph G_s (the whole set of non-negative q-dimensional vectors, for which G_s is optimal) is a closed cone [354, p. 326]. Indeed, if G_s is optimal for the vector $p \in R^q_+$ of processing times, it remains optimal for the processing times $\alpha p_{11}, \alpha p_{12}, \ldots, \alpha p_{nn_n}$ with any real $\alpha > 0$ (obviously, the stability radius is the largest radius of a stability ball, which is fully contained in the stability region). So, when considering the influence of 'load leveling' factors (numbers and distributions of operations per machines and per jobs) on the stability radius, we consider the same range of variations of the processing times for the problems of the first level: The processing times of the operations are uniformly distributed real numbers (with four digits after the decimal point) between the same bounds 10 and 100.

First, we generated small instances with 12 operations in each case, for which the exact values of the stability radii $\widehat{\varrho}_s(p)$ and $\overline{\varrho}_s(p)$ may be calculated on a PC 386 usually within some seconds using only internal memory of the computer. For each combination of the number of jobs from 3 to 7 and of the number of machines from 4 to 8, we randomly generated and solved 50 instances. Moreover, at the first level simulation includes four different types of problems in dependence on the distribution of the number of operations to the machines (evenly or randomly) and the operations, distributed to the same machine, to the jobs (evenly or randomly). Thus, we consider at the first level problems of the four types:

EE (evenly, evenly), ER (evenly, randomly),
RE (randomly, evenly), RR (randomly, randomly).

At the first level, we calculated the stability radii for 5000 job shop problems ($5000 = 4 \cdot 5 \cdot 5 \cdot 50 = 4$ (types) $\cdot$ 5 (combinations of the number of jobs n) $\cdot$ 5 (combinations of the number of machines m) $\cdot$ 50 (randomly generated instances in each series)) with 12 operations in each instance. Note that, if there were two or more optimal schedules for a sample problem, we calculated the stability radius for each of them. After solving the above problems

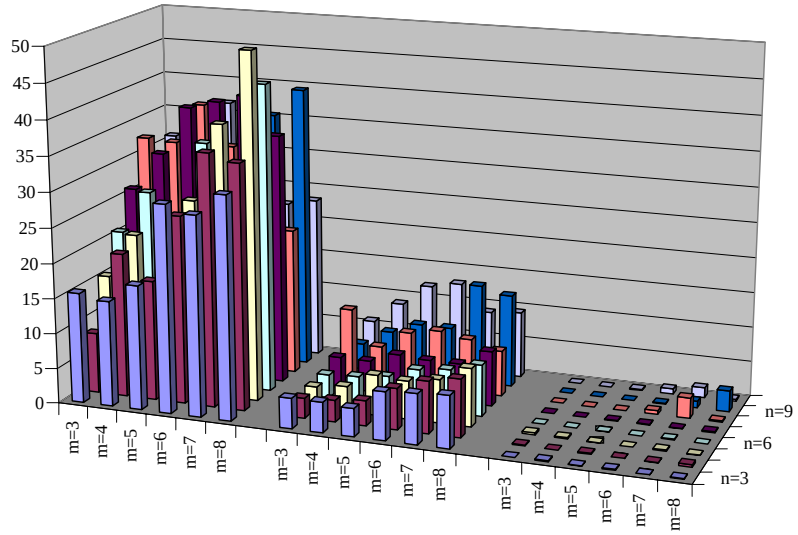
(without using external memory on a hard disk), we considered series of instances for each combination of the number of jobs from 8 to 10 and of the number of machines from 4 to 8, and for each combination of the number of jobs from 3 to 10 and $m = 3$. The number of operations in each instance was equal to 12. Since the number of edges in the mixed graph exceeded 20 (and so the number of generated feasible and infeasible digraphs exceeded $2^{20} = 1,048,576$), we had to use external memory on a hard disk for such instances and the running time for some of them achieved one or even two hours on a PC 486. So, we were forced to restrict the number of considered instances in the most difficult series for such combinations of the numbers of jobs and the numbers of machines to 10.

On the basis of the obtained information within the first level of experiments (for the instances with 12 operations), we designed the second and third ones. First, we decided to consider only instances generated for an evenly distributed number of operations to the machines and evenly distributed operations on the same machine to the jobs (i.e., type EE). At the second level, we calculated the exact values of the stability radii for job shop problems with 16 and 20 operations, considering 10 instances in each series while considering the influence of 'load leveling' factors. Note that for some of the instances at the second level, the CPU time of a Pentium PC exceeded 10 hours. Along with 'load leveling' factors, other ones also influence the complexity and stability of scheduling problems, e.g., the variability of $p_{ij}, O_{ij} \in Q$, across the entire shop and the variability of the average processing time from job to job or from machine to machine are also important factors of the complexity of shop scheduling problems (remind the famous job shop problem with 10 jobs and 10 machines given in [119], which was so difficult to attain due to a special processing time variability). Therefore, at the second level we also investigated the influence of the latter factors for random modifications of the processing times of the job shop problem with the same mixed graph G.

More precisely, for the same randomly generated mixed graph G at the second level of the simulation study, we considered six different ranges of variations of the given processing times, namely: [1, 10], [1, 100], [1, 1000], [10, 100], [10, 1000] and [100, 1000]. Obviously, the intervals [10, 100] and [100, 1000] may be obtained from the interval [1, 10] after multiplying with 10 and 100, respectively. However, the number of optimal schedules, and the number of problems with a zero value of the stability radius may be different for these three intervals since we consider all real numbers with a fixed number of decimal places. Due to the same reason, we consider the

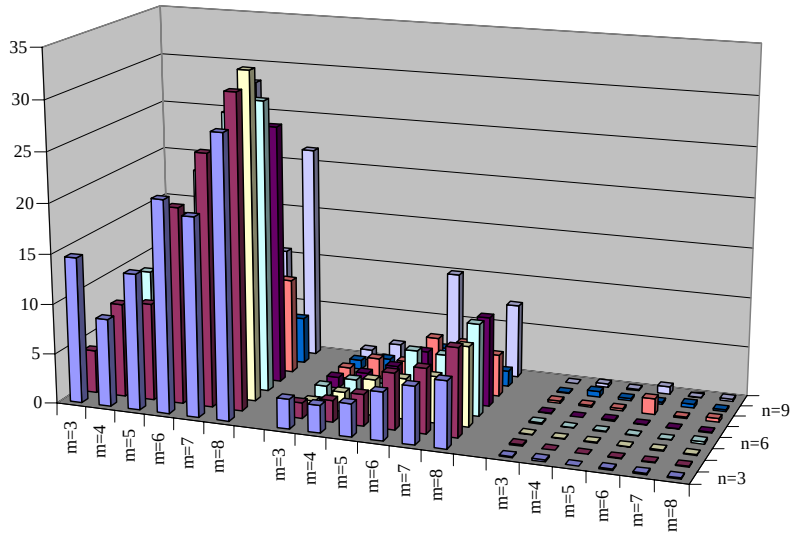
intervals [1, 100] and [10, 1000]. For the above segments, we calculated $\widehat{\varrho}_s(p)$ and $\overline{\varrho}_s(p)$ for each optimal schedule s in series with 50 instances. Moreover, we investigated instances in which different jobs had different ranges of the variations of the given processing times. At the third level, we considered a well-known job shop test problem with 6 jobs and 6 machines from [119] with different ranges of the variations of the given processing times across the entire shop and across different jobs. In Figures 1.1 and 1.2, we present the maximal, average and minimal values of the stability radii for each combination of the number of jobs n and the number of machines m, considered at the first level when the number of operations are evenly distributed to the machines and the operations on a machine are evenly distributed to the jobs (type EE). While the processing times are real numbers between 10 and 100, the stability radii are approximately between 0.001 and 50 for $\mathcal{C}_{max}$ and between 0.001 and 35 for $\sum \mathcal{C}_i$. Similar data for the other three types of distributing the operations are given in Figures 1.3 and 1.4 (types ER, RE and RR). The largest value of $\widehat{\varrho}_s(p)$ was about 90, and the largest value of $\overline{\varrho}_s(p)$ was about 70. For all types EE, ER, RE and RR, the average value of $\widehat{\varrho}_s(p)$ was larger than that of $\overline{\varrho}_s(p)$. An obvious conclusion from these diagrams is that an optimal makespan schedule (Figures 1.1 and 1.3) is more stable than an optimal mean flow time schedule (Figures 1.3 and 1.4). An important issue from Figures 1.1 - 1.4 is also that for each series of instances the smallest value of $\widehat{\varrho}_s(p)$ and $\overline{\varrho}_s(p)$ is greater than zero.

The results for the sample problems of the second level for 'load leveling' factors are presented in Table 1.1, where the minimal (MIN), average (AVE) and maximal (MAX) values of the stability radius divided by the maximal processing times (p_{MAX}) are given in columns 2, 3, and 4, and similar values divided by the average processing times (p_{AVE}) are given in columns 5, 6, and 7. During our experiments, we also determined the largest number γ of competitive digraphs in the sequence $(G_{i_1}, G_{i_2}, \ldots, G_{i_\gamma}, \ldots, G_{i_{\lambda'}}, \ldots, G_{i_\lambda})$ (where the digraphs are ordered according to non-decreasing objective function values) and the number λ' of the digraph, which was the last considered one in this sequence while calculating the exact value of the stability radius. Column 8 contains the average values of the percentage of digraphs, which may be a competitive digraph for the optimal one ($100t/\lambda$). For the set of instances presented in Table 1.1 with the mean flow time criterion, these values are bounded by 2.42%. When minimizing the makespan, these values are larger, but the latter results are mostly due to the large numbers of optimal makespan schedules (the average and maximal numbers NOS of optimal semiactive schedules for an instance are given in columns 9 and 10,



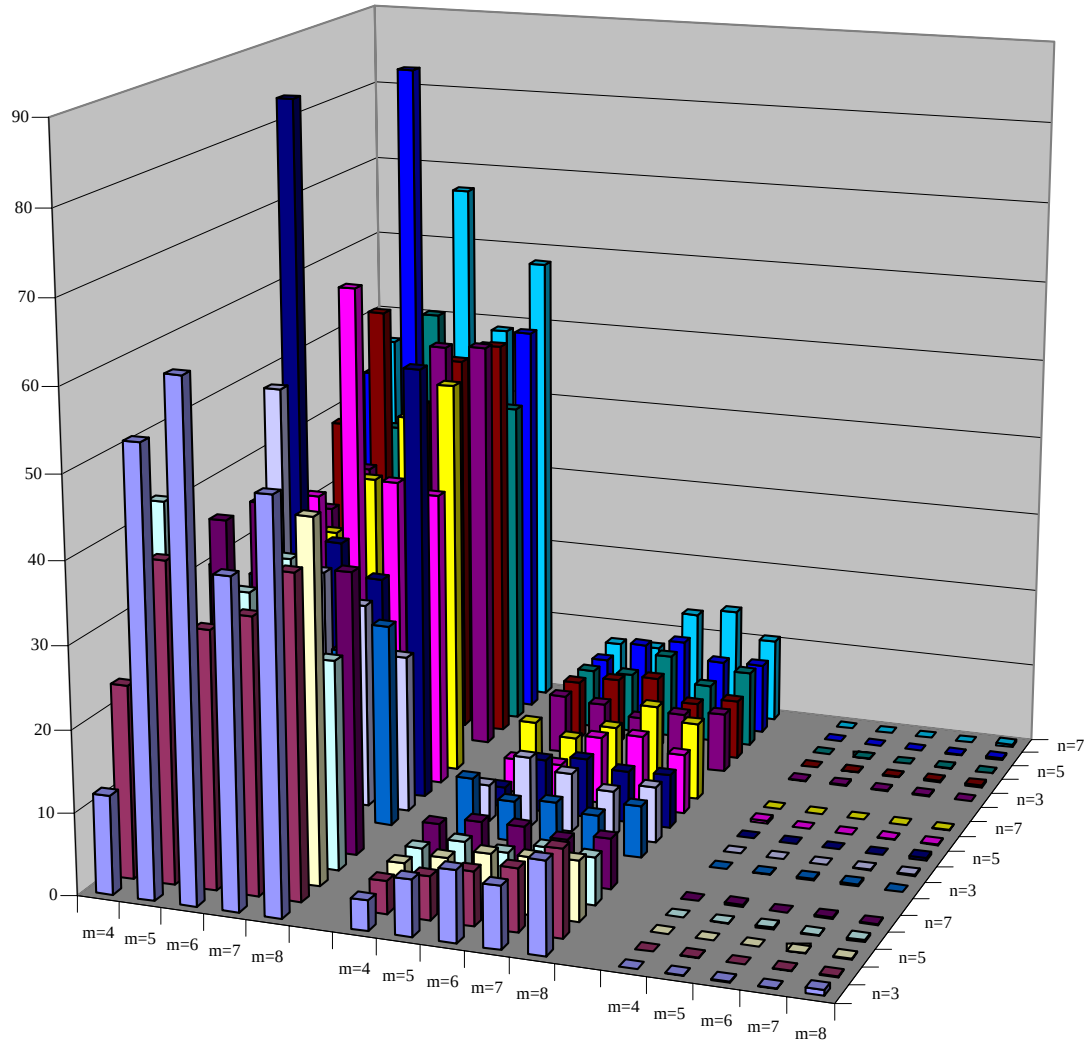
	m=3	m=4	m=5	m=6	m=7	m=8		m=3	m=4	m=5	m=6	m=7	m=8		m=3	m=4	m=5	m=6	m=7	m=8
n=3	15,71	15	17,71	29,48	28,4	31,56		4,33	4,3	4	6,83	7,12	7,43		0,01	0,14	0,03	0,15	0,01	0,08
n=4	8,6	20,49	17,04	26,74	35,86	34,85		2,84	3,11	3,58	5,86	7,38	8,2		0,14	0,01	0,16	0,03	0,06	0,34
n=5	15,65	22,04	21,67	27,76	38,84	49,27		3,02	3,61	5,72	5,41	6,15	8,27		0,3	0,18	0,01	0	0,18	0,04
n=6	20,96	27,11	21,63	34,9	30,88	43,82		3,34	3,58	4,13	5,62	6,13	7,32		0	0,01	0,03	0	0,12	0
n=7	26,16	31,69	38,68	39,84	40,99	35,63		4,47	4,44	5,88	5,57	5,57	7,83		0	0	0,08	0,11	0,05	0,15
n=8	32,69	32,4	38,18	32,47	14	20,91		10,2	5,18	7,71	8,45	7,74	6,53		0	0	0,01	0,53	2,89	0,17
n=9	23,07	22,07	22,33	22,31	36,44	40,39		3,74	6,01	7,59	7,56	14,25	13,27		0	0	0	0,12	0,97	2,98
n=10	31,13	28,14	36,7	28,14	22,31	23,19		5,84	8,94	12,02	12,83	9,08	9,49		0	0	0,14	0,75	1,44	0,29

Figure 1.1: *Maximal, average and minimal values of* $\widehat{\varrho}_s(p)$ *for the problems of type EE*



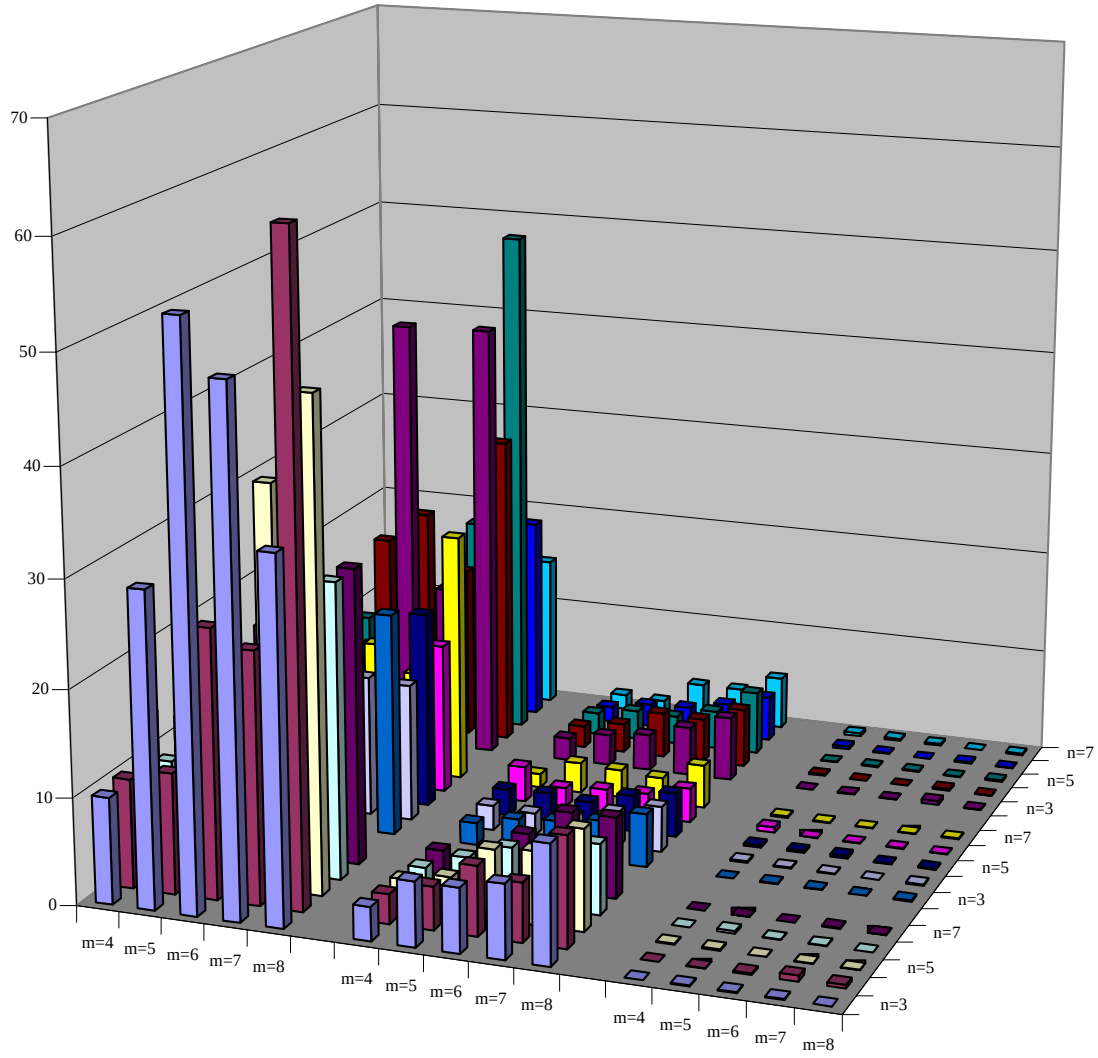
	m=3	m=4	m=5	m=6	m=7	m=8		m=3	m=4	m=5	m=6	m=7	m=8		m=3	m=4	m=5	m=6	m=7	m=8
n=3	14,57	8,76	13,58	21,15	19,77	28,04		2,95	2,73	3,26	4,78	5,74	6,63		0,09	0,2	0	0,13	0,15	0,13
n=4	4,3	9,35	9,71	19,61	25,17	31,23		1,55	2,17	3,16	5,62	6,45	8,83		0,14	0	0,02	0,05	0,07	0,09
n=5	2,27	5,29	12,33	15,13	16,32	32,7		0,69	1,95	3,59	4,04	4,63	7,95		0,07	0,04	0,08	0,04	0,13	0,09
n=6	3,27	10,93	7,16	21,72	27,74	29,16		1,22	2,2	2,22	5,86	5,77	9,21		0,15	0,04	0,09	0,05	0,04	0,23
n=7	2,68	8,59	16,15	20,28	20,88	25,96		1,11	1,89	2,96	4,8	5,08	8,92		0,07	0,01	0,14	0,06	0,03	0,08
n=8	2,82	5,32	5,57	13,32	18,89	9,57		1,11	2,41	2,47	5,23	5,06	4,19		0,22	0,16	0,21	1,58	0,12	0,36
n=9	2,57	4,13	10,46	6,52	2,97	4,67		0,93	1,4	2,39	3,24	1,53	1,57		0,07	0,56	0,2	0,21	0,39	0,19
n=10	4,27	3,97	2,5	28,14	10,67	21,55		1,05	1,96	1,24	10,01	3,48	7,43		0,01	0,32	0,14	0,75	0,03	0,24

Figure 1.2: *Maximal, average and minimal values of* $\overline{\varrho}_s(p)$ *for the problems of type EE*



	m=4	m=5	m=6	m=7	m=8		m=4	m=5	m=6	m=7	m=8		m=4	m=5	m=6	m=7	m=8
n=3	12,02	54,07	62,11	39,75	49,58		3,67	6,86	8,68	7,55	11,22		0,05	0,07	0,08	0,1	0,66
n=4	23,51	38,96	31,35	33,52	39,21		4,01	5,3	6,58	7,67	10,67		0	0,07	0	0,05	0,12
n=5	20,85	20,68	28,29	29,73	44,23		4,2	5,56	6,76	6,97	7,33		0,02	0,03	0,03	0,01	0,14
n=6	42,53	26,58	32,69	37,25	25,61		4,1	5,58	4,95	6,25	5,8		0	0,03	0,26	0,07	0,25
n=7	41,6	39,35	41,96	34,49	34,72		5,21	6,17	6,25	5,36	6,15		0	0,23	0,01	0,19	0,09
n=3	30,34	29,67	23,96	21,33	24,99		7,19	4,98	5,49	4,57	6,38		0	0,04	0,17	0,23	0,01
n=4	22,34	51,36	29,06	25,46	19,47		4,52	8,68	7,33	5,73	6,89		0	0	0	0,11	0,18
n=5	23,97	85,56	31,31	27,22	53,93		2,45	6,62	7,35	6,46	6,65		0	0	0	0,01	0,31
n=6	19,45	35,31	61,9	38,04	36,88		4,25	4,23	8,35	9,1	7,41		0,34	0	0	0,07	0,12
n=7	25,1	29,22	36,58	44,9	49,43		7,32	5,89	7,9	11,22	9,64		0	0	0	0	0,05
n=3	28,97	34,73	29,5	51,46	51,8		7,35	6,91	5,68	6,8	7,45		0,05	0,15	0,1	0,14	0,02
n=4	38,89	53,89	42,31	48,45	50,81		7,54	8,49	9,29	6,51	7,48		0	0,03	0,08	0,18	0,25
n=5	31,22	37,56	52,86	37,46	41,39		7,47	7,57	10,59	7,23	9,57		0	0,05	0	0,09	0,06
n=6	43,12	83,36	43,61	47,98	50,2		7,29	9,92	10,92	8,76	8,95		0	0	0	0,08	0,23
n=7	46,22	47,29	67,11	48,91	58,15		7,93	7,93	13,04	14	10,63		0	0	0	0	0,3

Figure 1.3: *Maximal, average and minimal values of $\widehat{\varrho}_s(p)$ for the problems of types ER, RE and RR*



	m=4	m=5	m=6	m=7	m=8		m=4	m=5	m=6	m=7	m=8		m=4	m=5	m=6	m=7	m=8
n=3	9,98	29,41	53,81	48,64	33,93		3,15	6,06	6,03	6,94	11,11		0,01	0,11	0,09	0,11	0,03
n=4	10,27	11,33	25,18	23,58	61,41		2,8	4,06	6,51	5,53	10,35		0,02	0,03	0,17	0,54	0,32
n=5	14,85	15,13	24,29	37,5	45,89		2,64	3,4	6,52	6,85	9,49		0,08	0,15	0,04	0,01	0,03
n=6	9,18	16,54	19,5	13,04	27,82		2,25	3,74	5,2	4,8	6,63		0,02	0,24	0,03	0,06	0,01
n=7	11	12,42	21,58	36,63	27,88		2,39	1,13	5,01	7,57	7,62		0,02	0,01	0,01	0,11	0,03
n=3	10,67	12,73	14,42	17,59	21,14		2,12	2,97	3,32	3,87	5,06		0	0,08	0,06	0,03	0,13
n=4	6,99	9,11	11,18	13,41	13,09		2,34	2,14	2,32	3,3	4,32		0,15	0,09	0,07	0,01	0,12
n=5	11,21	14,45	8,78	11,67	18,76		2,56	2,67	2,35	3,48	4,18		0,21	0,21	0,27	0,03	0,11
n=6	5,29	5,15	8,77	9,51	14,44		3,4	1,82	2,15	2,25	3,27		0,51	0,02	0,03	0,06	0,04
n=7	3,17	10,25	12,48	9,94	23,97		1,3	2,9	2,7	2,43	4,15		0,14	0,09	0,06	0,04	0,11
n=3	7,54	9,4	42,06	16	42,27		2,21	3	3,49	4,75	6,2		0,01	0,07	0,05	0,38	0,08
n=4	9,47	19,1	22,12	16,73	30,19		2,11	2,8	4,39	4,27	5,71		0,12	0,01	0,01	0,17	0,07
n=5	9,45	9,11	11,83	20,48	49,67		2,15	2,83	2,74	3,75	6,13		0,06	0,1	0,11	0,05	0,17
n=6	4,4	8,5	7,52	14,2	19,77		1,49	2,2	2,37	3,18	4,39		0,26	0,07	0	0,13	0,07
n=7	3,86	2,87	9,56	14,75	14,68		1,46	1,31	3,45	3,45	5,07		0,3	0,21	0,17	0,03	0,16

Figure 1.4: *Maximal, average and minimal values of $\overline{\varrho}_s(p)$ for the problems of types ER, RE and RR*

respectively). Note that for some 6 x 6 instances (i.e., those with 6 jobs and 6 machines), 7 x 7 instances and 8 x 8 instances, the number λ of all semiactive schedules was not calculated in our experiments, and therefore, the values of $100\gamma/\lambda$ are not presented for these series. If there is more than one optimal schedule, we calculate the differences of their stability radii. The average and maximal values of these differences (DIFF) are presented in columns 11 and 12, respectively. We can also note that for the mean flow time criterion, an optimal schedule is usually uniquely determined, and even if there are two optimal mean flow time schedules, they have often the same stability radius. Consequently, for the mean flow time criterion we have not much need to look for an optimal schedule with the largest stability radius. Next, we present the randomly generated mixed graph G for the job shop

Table 1.1: *Randomly generated problems*

n x m	RADIUS / p_{MAX}			RADIUS / p_{AVE}			$\frac{100\gamma}{\lambda}$	NOS		DIFF	
	MIN	AVE	MAX	MIN	AVE	MAX		AVE	MAX	AVE	MAX
1	2	3	4	5	6	7	8	9	10	11	12
Maximum flow time											
6 x 6	0.01	0.62	4.26	0.02	1.09	7.40	-	21.50	78	1.60	3.83
7 x 7	0.07	1.76	11.16	0.12	3.45	23.59	-	15.60	43	2.71	9.72
8 x 8	0.07	3.43	12.66	0.13	6.00	17.80	-	17.00	70	4.91	12.20
9 x 9	0.00	3.97	11.52	0.00	6.91	22.14	4.43	28.90	144	6.07	11.38
10 x 10	0.18	3.33	21.90	0.32	5.97	41.38	1.54	12.40	48	5.68	18.84
Mean flow time											
6 x 6	0.33	1.28	5.10	0.59	2.25	8.67	2.27	1.10	2	0.00	0.00
7 x 7	0.23	1.33	6.57	0.40	2.32	11.19	2.42	1.20	2	0.00	0.00
8 x 8	0.26	1.86	6.54	0.51	3.28	11.85	0.03	1.20	2	0.00	0.00
9 x 9	0.60	2.20	4.41	1.10	3.84	8.22	0.10	1.10	2	0.00	0.00
10 x 10	0.46	3.83	8.05	0.75	6.79	13.69	0.57	1.00	1	0.00	0.00

problem $\mathcal{J}6/n = 4/\Phi$ with 4 jobs and 6 machines, which is used for the simulation study of the influence of the variability of the processing times:

$$Q = \{O_{1,1}, O_{1,2}, \ldots, O_{4,4}\};\ E = \{[O_{1,1}, O_{2,1}], [O_{1,1}, O_{3,4}], [O_{2,1}, O_{3,4}],$$
$$[O_{1,2}, O_{2,3}], [O_{1,3}, O_{2,2}], [O_{1,3}, O_{4,2}], [O_{2,2}, O_{4,2}], [O_{1,4}, O_{3,2}], [O_{1,4}, O_{4,1}],$$
$$[O_{3,2}, O_{4,1}], [O_{2,4}, O_{3,1}], [O_{2,4}, O_{4,4}], [O_{3,1}, O_{4,4}], [O_{3,3}, O_{4,3}]\}$$

(see Figure 1.5). Computational results for this mixed graph are given in Tables 1.2 and 1.3. Table 1.2 presents the computational results for different ranges of the processing times for the same mixed graph G, which is described above. Both criteria $\mathcal{C}_{max}$ and $\sum \mathcal{C}_i$ are considered for the same 50 examples for which the obtained results are presented in the corresponding rows of Table 1.2 (row i for $\mathcal{C}_{max}$ corresponds to row $i + 6$ for $\sum \mathcal{C}_i$).

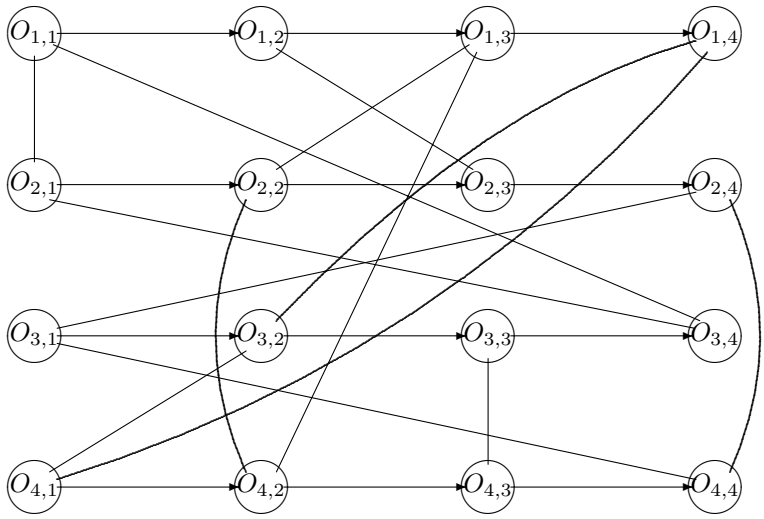


Figure 1.5: *Randomly generated mixed graphs* (Q^J, A^J, E^J) *for problem* $\mathcal{J}6/n=4/\Phi$

Table 1.2: *Problem* $\mathcal{J}6/n=4/\Phi$, $\Phi \in \{\mathcal{C}_{max}, \sum \mathcal{C}_i\}$, *with different ranges of the variations of the job processing times* p_{ij}

Bounds for p_{ij}		RADIUS/p_{MAX}			RADIUS/p_{AVE}			$\frac{100\gamma}{\lambda}$	$\frac{100\lambda'}{\lambda}$	NOS		NMO	DIFF	
LB	UB	MIN	AVE	MAX	MIN	AVE	MAX			AVE	MAX		AVE	MAX
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Maximum flow time														
1	10	0.07	2.37	8.48	0.13	4.40	17.15	0.48	2.93	2.80	12	34	0.15	0.75
1	100	0.02	2.31	12.11	0.04	4.91	27.38	0.71	5.02	3.58	21	34	2.35	10.53
1	1000	0.00	3.63	13.83	0.00	8.33	36.71	0.73	8.66	4.26	36	32	30.93	132.28
10	100	0.13	2.52	10.78	0.26	4.64	21.18	0.50	4.71	2.74	12	31	2.36	9.53
10	1000	0.01	3.08	13.06	0.01	6.23	31.36	0.62	8.07	4.60	30	41	26.90	123.57
100	1000	0.04	2.88	11.89	0.07	5.20	24.85	0.53	4.18	2.30	12	25	20.63	108.90
Mean flow time														
1	10	0.06	2.56	10.17	0.11	4.76	17.89	0.30	3.87	1.02	2	1	0.13	0.13
1	100	0.07	2.47	9.90	0.13	5.06	20.54	0.34	4.44	1.02	2	1	0.00	0.00
1	1000	0.03	2.07	10.21	0.05	3.95	16.96	0.30	3.40	1.04	2	2	1.80	3.59
10	100	0.12	2.41	7.30	0.21	4.51	15.63	0.30	3.42	1.00	1	0	-	-
10	1000	0.17	2.26	8.67	0.37	4.52	16.29	0.32	3.40	1.12	2	6	6.18	22.48
100	1000	0.05	2.53	11.43	0.11	4.48	19.78	0.33	4.22	1.04	2	2	16.37	32.75

Table 1.3 presents computational results for different ranges of the values p_{ij} of the operations of different jobs. Along with the columns defined for Table 1.1, we also present the percentage of considered digraphs while calculating the exact value of the stability radius (column 10 in Table 1.2 and column 16 in Table 1.3) and the number NMO of problems with two or more optimal schedules (column 13 in Table 1.2 and column 19 in Table 1.3).

For the problems considered at the second level, the 'superiority' of the stability radius for the makespan criterion is lost in most cases. At least the minimal values of $\overline{\varrho}_s(p)$ became larger than those of $\widehat{\varrho}_s(p)$. Of course,

the large number of optimal makespan schedules has influenced this relation essentially, but even on average, we cannot find a large superiority of the stability radius of one criterion over the other (for the considered classes of randomly generated job shop problems). Next, we discuss some questions on the basis of our experimental calculation of the stability radii of the optimal schedules for small randomly generated job shop problems.

How often is the stability radius equal to zero? In the experiments at the first and the second levels, we obtained only once a stability radius equal to zero for criterion $\mathcal{C}_{max}$ and never for criterion $\sum \mathcal{C}_i$ although it takes not much effort to construct such an example by hand (see Theorem 1.1 for $\mathcal{C}_{max}$ and Theorem 1.6 for $\sum \mathcal{C}_i$). So, in principle, to find an optimal schedule for almost all problems generated in our experiments has sense. On the other hand, in many series there are instances with very small values of the stability radius (even less than 0.001). So, if for such an instance the precision of the processing times is not sufficiently high, we have no guarantee that the (a priori) constructed optimal schedule will be indeed the best one in its practical realization.

May the stability radius be infinitely large? From theoretical results it follows that for any given n and m, there exist job shop problems with an optimal makespan schedule s, which remains optimal for any feasible variation of the processing times, i.e., $\widehat{\varrho}_s(p) = \infty$ (see Theorem 1.2 and Theorem 1.3). In particular, an easily verifiable characterization of such a schedule has been derived for criterion $\mathcal{C}_{max}$ (see Theorem 1.3). In contrast, it was shown that for mean flow time, we have $\overline{\varrho}_s(p) \leq \max_{O_{ij} \in Q}\{p_{ij}\}$ for any job shop problem (see Theorem 1.7). Although in [192] a practical example of an infinitely large stability radius was presented (for a traffic-light on the intersection of two roads), nevertheless such a shop appears to be rather artificial for large numbers of jobs and machines. Surprisingly, in our randomly generated job shop problems with the makespan criterion an infinitely large stability radius was obtained not seldom, at least essentially more often than a zero stability radius (of course, we did not include infinite stability radii while calculating the average and maximal values of $\widehat{\varrho}_s(p)$). So, our experiments indicate that the results derived in [192, 311] will have not only theoretical significance.

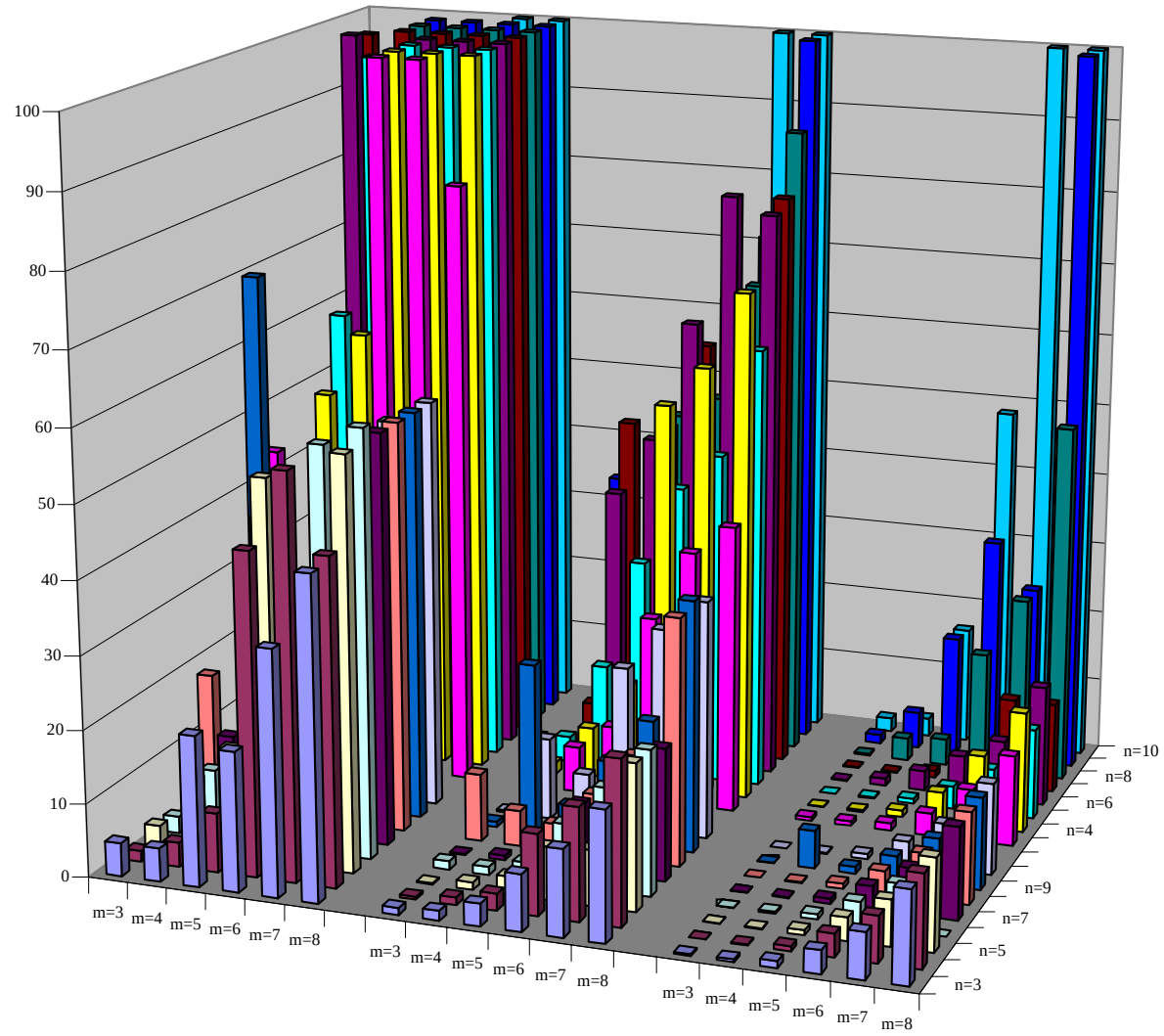
How many 'best' schedules do we need to consider? As already mentioned, we also determined the number γ of competitive digraphs and the number λ' of considered digraphs while calculating the exact value of the stability radius. For the problems of the first level, the diagrams for the percentage of the numbers γ and for the percentage of the numbers λ' for the problems

of type EE are presented in Figure 1.6 (Figure 1.7) for criterion $\mathcal{C}_{max}$ (for criterion $\sum \mathcal{C}_i$, respectively). In the front part of the diagrams in Figures 1.6 and 1.7, the minimal, average and maximal values of the percentages $100\gamma/\lambda$ are presented while in the background of these diagrams the minimal, average and maximal values of the percentages $100\lambda'/\lambda$ are presented. As it follows from Figures 1.6 and 1.7, the value $100\gamma/\lambda$ may be smaller than 1 % and it is not greater than 73 % for $\mathcal{C}_{max}$ and not greater than 56 % for $\sum \mathcal{C}_i$.

Table 1.3: *Problem* $\mathcal{J}6/n=4/\Phi$, $\Phi \in \{\mathcal{C}_{max}, \sum \mathcal{C}_i\}$, *with variability of* p_{ij} *from job to job*

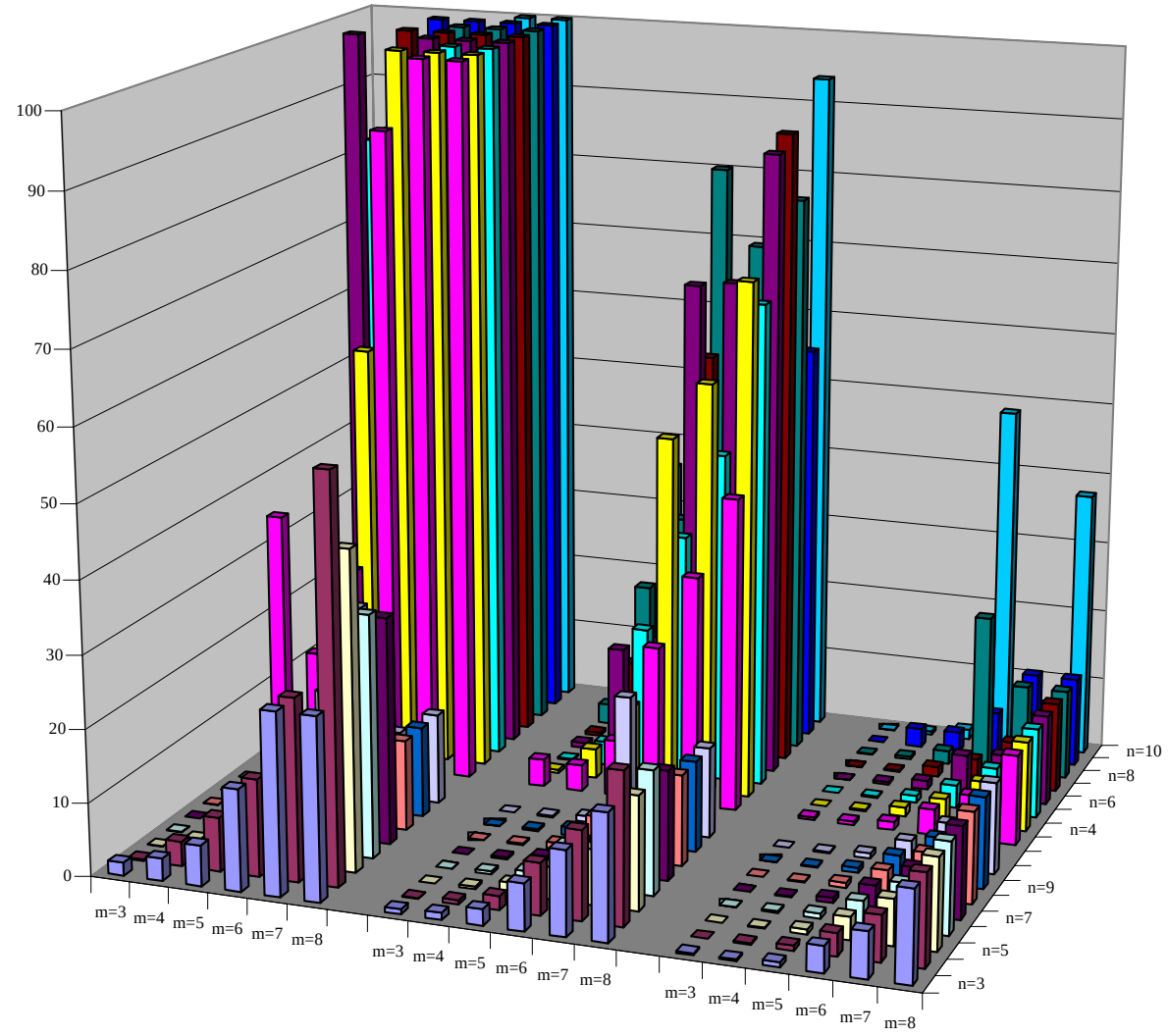
Lower bound L_i and upper bound B_i for p_{ij}								$\frac{\varrho_s}{p_{MAX}}$			$\frac{\varrho_s}{p_{AVE}}$			$\frac{100\gamma}{\lambda}$	$\frac{100\lambda'}{\lambda}$	NOS		NMO	DIFF	
L_1	U_1	L_2	U_2	L_3	U_3	L_4	U_4	MI	AV	MA	MI	AV	MA			AV	MA		AV	MA
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Maximum flow time																				
10	20	30	40	70	80	90	100	0.01	0.87	3.81	0.01	1.59	6.92	3.83	9.15	30	54	50	0.5	3.2
10	20	40	50	60	70	90	100	0.00	0.98	4.37	0.01	1.79	8.06	1.10	5.86	8	16	50	0.6	3.5
10	30	30	50	60	80	80	100	0.01	1.41	6.62	0.02	2.56	12.01	1.70	7.95	15	32	50	1.3	5.1
10	40	30	60	50	80	10	100	0.00	1.66	8.56	0.01	2.99	15.59	1.21	6.97	8	22	50	1.4	6.9
10	60	30	60	50	80	50	100	0.06	1.84	8.03	0.09	3.32	15.27	0.84	4.25	6	24	49	1.0	7.1
10	60	10	60	50	100	50	100	0.00	3.30	12.05	0.01	6.02	22.56	1.30	12.52	12	24	50	3.8	10.6
10	40	40	70	10	100	10	100	0.01	4.28	14.81	0.01	8.31	31.23	0.65	18.38	16	58	50	6.3	14.0
10	30	10	50	60	80	60	100	0.01	1.78	9.82	0.02	3.20	18.20	1.24	6.81	11	32	50	1.6	9.3
10	20	30	50	60	80	80	100	0.00	1.03	4.69	0.00	1.87	8.85	2.36	9.54	19	54	50	1.1	4.1
10	25	25	75	25	75	50	100	0.09	2.73	14.34	0.16	5.30	25.56	0.81	7.93	4	12	42	2.0	14.1
Mean flow time																				
10	20	30	40	70	80	90	100	0.08	1.08	2.97	0.14	1.95	5.36	0.27	1.56	1	1	0	0	0
10	20	40	50	60	70	90	100	0.04	1.21	2.82	0.07	2.21	5.22	0.26	1.28	1	1	0	0	0
10	30	30	50	60	80	80	100	0.02	1.15	4.11	0.04	2.08	7.33	0.31	1.64	1	1	0	0	0
10	40	30	60	50	80	70	100	0.09	1.54	4.85	0.15	2.78	8.02	0.31	1.78	1	1	0	0	0
10	60	30	60	50	80	50	100	0.05	1.87	6.14	0.09	3.35	10.85	0.35	2.39	1	1	0	0	0
10	60	10	60	50	100	50	100	0.01	1.45	4.41	0.01	2.62	7.53	0.32	2.02	1	3	1	0	0
10	40	40	70	70	100	10	100	0.07	2.32	7.83	0.12	4.23	13.62	0.33	3.40	1	2	2	0	0
10	30	10	50	60	80	60	100	0.04	1.31	4.66	0.08	2.34	7.97	0.33	1.51	1	1	0	0	0
10	20	30	50	60	80	80	100	0.01	0.94	3.26	0.01	1.70	5.95	0.25	1.14	1	1	0	0	0
10	25	25	75	25	75	50	100	1.01	2.18	5.51	0.02	4.17	9.92	0.31	3.27	1	1	0	0	0

It should be noted that for the case of a large number of machines and a small number of operations (at the first level when $q = 12$), there often exist only a few feasible semiactive schedules which make the relative values of γ and λ' rather large. Moreover, for criterion $\mathcal{C}_{max}$, we have a relatively large number of optimal schedules which also enlarges the relative values of γ and λ'. Thus, calculating the exact value of the stability radius on the basis of bounds (2.34) and (2.63) may require to consider the whole set $\Lambda(G)$ of digraphs for the problems considered at the first level of our simulation study. From Tables 1.2 and 1.3, it follows that the competitive digraphs are within 3.83 % of the whole set of feasible digraphs for criterion



	m=3	m=4	m=5	m=6	m=7	m=8		m=3	m=4	m=5	m=6	m=7	m=8		m=3	m=4	m=5	m=6	m=7	m=8
n=3	4,55	4,55	20,5	19,05	33,33	43,75		0,96	1,33	3,05	7,71	11,76	17,66		0,23	0,35	0,93	3,13	6,25	12,5
n=4	1,5	3,36	8,05	44	55	44,44		0,39	1,03	2,33	11,05	15,33	22,4		0,03	0,15	0,69	3,13	6,25	12,5
n=5	3	9,9	15,03	52,08	15,63	56,25		0,16	1,03	2,49	7,24	7,57	19,9		0,04	0,15	0,69	3,13	6,25	12,5
n=6	2,3	9,5	11,11	27,78	55,56	58,33		1,15	1,11	2,41	8,41	13,97	19,78		0,03	0,25	0,69	3,13	6,25	0
n=7	2,63	12,35	43,21	35,94	40,63	56,25		0,13	0,64	2,67	9,21	11,15	18,1		0,03	0,15	0,69	3,13	6,25	12,5
n=8	18,56	9,76	14,24	27,08	31,25	56,25		9,28	4,88	3,78	8,56	14,6	33,92		0,03	0,15	0,69	3,13	6,25	12,5
n=9	6,83	72,73	17,01	26,56	53,13	56,25		0,71	23,31	4,42	11,23	17,47	34,55		0,02	5,17	0,93	3,13	6,25	12,5
n=10	2,8	22,48	25,35	26,56	53,13	56,25		0,3	11,16	6,85	22,41	28,36	32,81		0,02	0,15	0,69	3,13	6,25	12,5
n=3	44,16	27,27	37,76	100	100	83,33		7,26	6,35	9,88	25,95	35,74	39,85		0,49	0,65	1,04	3,13	7,14	12,5
n=4	8,8	51,47	60,37	100	100	100		1,67	7,28	14,22	54,26	59,87	70,53		0,09	0,28	0,93	4,17	10	16,67
n=5	19	61,52	98,04	100	100	100		3,58	14,5	30,07	41,17	46,32	61,5		0,04	0,22	0,69	3,13	6,25	12,5
n=6	3,24	100	94,44	100	100	100		1,19	38	46,19	63,03	81,06	78,83		0,11	1,01	2,78	5,56	8,33	16,67
n=7	24,99	99,31	100	100	100	100		4,91	46,78	46,55	58,75	62,13	80,13		0,14	0,15	0,93	3,13	12,5	12,5
n=8	22,71	67,28	100	100	100	100		7,22	24,74	46,92	49,95	66,46	88,19		0,24	3,19	3,7	16,67	25	50
n=9	87,09	72,73	100	100	100	100		35,3	23,31	47,5	62,97	72,19	100		1,24	5,17	16,67	31,25	25	100
n=10	23,47	77,78	66,67	50	100	100		11,48	24,83	45,44	49,69	100	100		1,94	2,55	16,32	48,44	100	100

Figure 1.6: *Percentage of the number γ (and λ') of competitive (considered) digraphs for the problems of type EE for $\widehat{\varrho}_s(p)$*



	m=3	m=4	m=5	m=6	m=7	m=8		m=3	m=4	m=5	m=6	m=7	m=8		m=3	m=4	m=5	m=6	m=7	m=8
n=3	1,77	3,07	5,6	13,89	25	25		0,68	0,96	2,21	6,41	11,53	17,19		0,23	0,25	0,69	3,57	6,25	12,5
n=4	0,23	3,38	7,41	13,33	25	55,56		0,11	0,59	1,9	7,1	12,19	20,75		0,03	0,23	0,69	3,13	6,25	12,5
n=5	0,08	1,79	8,73	10,94	15,63	43,75		0,05	0,37	1,55	4,43	7,57	15,5		0,02	0,15	0,69	3,13	6,25	12,5
n=6	0,1	0,88	2,47	11,11	22,22	33,33		0,04	0,41	1,11	5,43	9,73	16,96		0,02	0,25	0,69	3,13	6,25	12,5
n=7	0,03	0,82	2,31	9,38	15,63	31,25		0,02	0,28	1,18	3,92	7,31	15		0,02	0,15	0,69	3,13	6,25	12,5
n=8	0,08	0,31	1,74	12,5	12,5	12,5		0,04	0,22	0,97	5,42	7,08	12,5		0,03	0,15	0,69	3,13	6,25	12,5
n=9	0,06	0,63	2,78	4,69	6,25	12,5		0,04	0,28	1,12	3,29	6,25	12,5		0,02	0,15	0,69	3,13	6,25	12,5
n=10	0,04	0,51	1,04	26,56	9,38	12,5		0,02	0,24	0,82	18,26	6,56	12,5		0,02	0,15	0,69	3,13	6,25	12,5
n=3	34,63	15,34	28	90	100	100		3,82	3,68	7,81	21,73	32,21	43,66		0,36	0,51	1,14	3,57	6,25	12,5
n=4	2,16	22,27	58,02	100	100	100		0,49	4,12	11,01	49,57	57,61	72		0,03	0,23	1,22	3,13	6,25	12,5
n=5	0,84	15,24	86,51	93,75	100	100		0,19	3,32	20,35	34,19	46,32	67,75		0,05	0,21	0,98	3,13	6,25	12,5
n=6	2,51	100	87,04	100	100	100		0,71	15,18	16,84	68,29	69,01	87,08		0,04	0,38	1,23	5,56	6,25	12,5
n=7	0,95	64,75	100	100	100	100		0,48	11,44	27,42	57	62	88,88		0,04	0,21	1,39	3,13	6,25	12,5
n=8	9,57	58,13	74,54	100	100	100		2,87	21,09	31,69	82,29	71,88	78,75		0,11	0,39	1,85	21,88	12,5	12,5
n=9	8,51	81,19	100	100	100	100		2,29	18,45	31,34	58,44	59,06	56,25		0,05	2,65	2,78	6,25	12,5	12,5
n=10	9,99	72,61	54,86	50	100	100		2,99	36,02	29,87	49,69	80,63	93,75		0,02	0,46	1,39	48,44	6,25	37,5

Figure 1.7: *Percentage of the number γ (and λ') of competitive (considered) digraphs for the problems of type EE for $\overline{\varrho}_s(p)$*

$\mathcal{C}_{max}$ and within 0.35 % for criterion $\sum \mathcal{C}_i$, and the percentage of digraphs which have been considered while calculating the stability radius is no more than 18.38 % for criterion $\mathcal{C}_{max}$ and no more than 4.44 % for criterion $\sum \mathcal{C}_i$. So, it is not necessary to construct the whole set of feasible digraphs for calculating the stability radius of an optimal digraph for these types of problems. After studying the obtained results at the first and second levels of our experiments, we enlarged the size of problems, which are still suitable for calculating the exact value of the stability radius (or at least its upper bound). For calculating the stability radius for instances of larger size, we constructed for each of them only the k best schedules (with $k = 100$ in most cases) by a direct enumeration of the whole set of feasible digraphs. Then, considering only these k best digraphs, we intended to calculate the stability radius of an optimal digraph (or optimal digraphs). If this process has stopped before the whole k best digraphs were compared with the optimal one, we have obtained the exact value of the stability radius due to the bounds (2.34) or (2.63), otherwise we have obtained at least an upper bound for the stability radius. Moreover, to shorten the running time, we used the branch-and-bound method for calculating the k best digraphs.

How can one combine this approach with the branch-and-bound method? The following approach to stability analysis for scheduling problems seems to be practically efficient. Using a branch-and-bound method (e.g. [62]), one can construct not only one optimal but the k best schedules (as it has been shown for the traveling salesman problem [228, 230] and for the binary linear programming problem [365], the running time of such a variant of a branch-and-bound algorithm grows rather slowly with k). In particular, in our computational study we used a branch-and-bound algorithm with the conflict resolution strategy. Due to an implicit enumeration of the feasible mixed graphs $G_{(s)}(p) = (Q, A_{(s)}, E_{(s)})$, we construct the k best ones and calculate the exact value or an upper bound for the stability radius of an optimal schedule in the same manner as described in the above paragraph "How many 'best' schedules do we need to consider?".

Note that, while an explicit enumeration of the digraphs $G_1(p)$, $G_2(p), \ldots, G_\lambda(p)$ gives the exact value of $\widehat{\varrho}_s(p)$ for $|E| \leq 30$, the branch-and-bound algorithm gives the possibility to calculate $\widehat{\varrho}_s(p)$ for $|E| \leq 100$ (often within the same CPU time). In particular, at the third level of the experiments, we considered the well-known classical job shop problem from [119] with 6 jobs and 6 machines. The assignment of the operations $Q = \{O_{1,1}, O_{1,2}, \ldots, O_{6,6}\}$ to the set of machines $M = \{M_1, M_2, \ldots, M_6\}$ is as follows:

$$
\begin{aligned}
Q_1 &= \{O_{1,2}, O_{2,5}, O_{3,4}, O_{4,2}, O_{5,5}, O_{6,4}\},\\
Q_2 &= \{O_{1,3}, O_{2,1}, O_{3,5}, O_{4,1}, O_{5,2}, O_{6,1}\},\\
Q_3 &= \{O_{1,1}, O_{2,2}, O_{3,1}, O_{4,3}, O_{5,1}, O_{6,6}\},\\
Q_4 &= \{O_{1,4}, O_{2,6}, O_{3,2}, O_{4,4}, O_{5,6}, O_{6,2}\},\\
Q_5 &= \{O_{1,6}, O_{2,3}, O_{3,6}, O_{4,5}, O_{5,3}, O_{6,5}\},\\
Q_6 &= \{O_{1,5}, O_{2,4}, O_{3,3}, O_{4,6}, O_{5,4}, O_{6,3}\}.
\end{aligned}
$$

For this problem, each job has to be processed on each machine exactly once and hence, we have $q = 6$ x $6 = 36$ and $|E| = 6$ x $\binom{6}{2} = 90$. By the branch-and-bound algorithm we constructed $k = 150$ best schedules: 22 of them are optimal with $\mathcal{C}_{max} = 55$ and 54 other schedules have a makespan value equal to 56, and at least 74 schedules have a makespan value equal to 57. We calculated an upper bound for $\widehat{\varrho}_s(p)$ for each optimal makespan schedule. It turned out that 14 of them have a zero stability radius and the other 8 optimal schedules have an upper bound for $\widehat{\varrho}_s(p)$ equal to 0.08333. The existence of unstable optimal schedules for this test problem is implied mainly by the fact that its processing times are integers from 1 to 10.

We also randomly generated 50 instances with 6 jobs, 6 machines and 36 operations. Again, each job has to be processed on each machine exactly once (i.e., we considered classical job shop problems), but in contrast to the problem from [119], the processing times were uniformly distributed real numbers between 1 and 10. For each generated problem with 36 operations, we constructed 50 best schedules (for the makespan criterion) on the basis of the branch-and-bound algorithm and calculated upper bounds for $\widehat{\varrho}_s(p)$ for each optimal makespan schedule which was constructed. Note that 45 of these instances had more than one optimal makespan schedule and among them, 7 instances had 50 or even more optimal makespan schedules. The average value of the stability radius $\widehat{\varrho}_s(p)$ was equal to 0.12939, and for all calculated optimal makespan schedules s the following bounds were satisfied: $0.001 \leq \widehat{\varrho}_s(p) \leq 0.87455$. We also calculated the differences between $\widehat{\varrho}_s(p)$ for different optimal makespan schedules $s \in S$ of the same instance (if this instance had two or more optimal makespan schedules). The maximum of this difference was equal to 0.84636, the average difference was 0.11709 and some optimal makespan schedules had the same stability radius. Among the 50 instances, there was no optimal schedule with a zero stability radius.

To investigate the influence of the variability of the processing times p_{ij} on the stability radius, we considered again the test problem with 6 jobs and 6 machines given in [119], but with different distributions of the processing times to the operations. More precisely, the mixed graph $G = (Q, A, E)$ was

Table 1.4: *Test problem $\mathcal{J}6/n=6/\mathcal{C}_{max}$ with variability of p_{ij}*

p_{ij}		RADIUS/p_{MAX}			RADIUS/p_{AVE}			NOS		NMO	DIFF	
LB	UB	MIN	AVE	MAX	MIN	AVE	MAX	AVE	MAX		AVE	MAX
1	2	3	4	5	6	7	8	9	10	11	12	13
Common bounds for p_{ij} for different jobs												
1	10	0.0067	0.1843	0.8744	0.0138	0.3374	1.5340	13.5	52	8	0.0106	0.0393
1	100	0.0077	0.3265	1.2092	0.0163	0.6158	2.0630	26.7*	100*	10	0.1705	1.1278
1	1000	0.0749	0.6679	2.4344	0.1639	1.4461	5.1843	31.5	90	10	6.3276	22.6764
10	100	0.0440	0.7733	3.9813	0.0820	1.4277	7.4289	17.5*	100*	9	0.5540	3.6507
10	1000	0.0070	0.4436	1.7260	0.0116	0.8298	3.2587	27.4*	100*	10	3.5290	10.9608
100	1000	0.0308	0.5694	1.9779	0.0564	1.0109	3.8182	17.0	54	10	7.2888	18.3886
Different bounds for p_{ij} for different jobs												
LB_i^1	UB_i^1	0.0000	0.6429	3.9997	0.0000	1.1009	7.2044	41.2*	100*	10	0.9175	3.1636
LB_i^2	UB_i^2	0.0216	0.5046	1.3379	0.0383	0.8487	2.3764	5.2	12	9	0.0000	0.0000
LB_i^3	UB_i^3	0.0000	1.0051	4.2719	0.0000	1.7247	7.4870	74.6*	100*	10	1.7654	4.1433
LB_i^4	UB_i^4	0.0031	0.9716	9.2608	0.0053	1.7292	16.4208	82.2*	100*	10	2.1819	8.6476

defined in accordance with [119], but the processing times were randomly generated real numbers with the same lower and upper bounds for all jobs (see rows 1 - 6 in Table 1.4) and with different lower and upper bounds for different jobs (rows 7 - 10 in Table 1.4). Each row in Table 1.4 presents the results obtained for a series of 10 instances. For each instance, we calculated the stability radius using 100 best schedules generated by the branch-and-bound algorithm. For row 7 in Table 1.4, the lower bound LB_i^1 and the upper bound UB_i^1 for job J_i are as follows:

$$\begin{array}{llll} LB_1^1 = 10, & UB_1^1 = 40; & LB_2^1 = 20, & UB_2^1 = 50; \\ LB_3^1 = 30, & UB_3^1 = 60; & LB_4^1 = 50, & UB_4^1 = 80; \\ LB_5^1 = 60, & UB_5^1 = 90; & LB_6^1 = 70, & UB_6^1 = 100. \end{array}$$

For row 8, these bounds are:

$$\begin{array}{llll} LB_1^2 = 10, & UB_1^2 = 60; & LB_2^2 = 30, & UB_2^2 = 60; \\ LB_3^2 = 40, & UB_3^2 = 60; & LB_4^2 = 50, & UB_4^2 = 70; \\ LB_5^2 = 50, & UB_5^2 = 80; & LB_6^2 = 50, & UB_6^2 = 100. \end{array}$$

For row 9, these bounds are:

$$\begin{array}{llll} LB_1^3 = 10, & UB_1^3 = 40; & LB_2^3 = 20, & UB_2^3 = 50; \\ LB_3^3 = 40, & UB_3^3 = 70; & LB_4^3 = 60, & UB_4^3 = 90; \\ LB_5^3 = 70, & UB_5^3 = 100; & LB_6^3 = 10, & UB_6^3 = 100. \end{array}$$

For row 10, these bounds are:

$$\begin{array}{llll} LB_1^4 = 10, & UB_1^4 = 30; & LB_2^4 = 20, & UB_2^4 = 40; \\ LB_3^4 = 30, & UB_3^4 = 50; & LB_4^4 = 60, & UB_4^4 = 80; \\ LB_5^4 = 70, & UB_5^4 = 90; & LB_6^4 = 80, & UB_6^4 = 100. \end{array}$$

In Table 1.4, we marked the series of instances, for which the number of optimal schedules is larger than 100 by an asterisk. Since we calculated only 100 best schedules for each instance, we had not the exact number of optimal semiactive makespan schedules. The developed software did not allow us to find $\overline{\varrho}_s(p)$ for most of the above instances with 36 operations and 90 edges since the calculation of the stability radius for the mean flow time criterion is essentially more time-consuming than for the makespan.

How to use this approach for problems of practical size? For large instances, for which a direct enumeration of all feasible digraphs was practically impossible, we constructed only a subset of feasible digraphs, selected then the best digraph G_s among them and calculated an upper bound for the 'stability radius' of G_s by a comparison with all other digraphs that have been constructed. This variant of the implementation of the software may be useful for some practical problems. Indeed, in reality OR workers have at most one or only a few feasible schedules (usually without an exact information about their quality). In the case when a set of feasible schedules is known, we can investigate the stability radius of the best of them in comparison with the others at hand. Even if we do not have the possibility to find an optimal schedule by a branch-and-bound method and only an approximate schedule (with an information about its quality) or a heuristic schedule has been constructed, we can investigate the 'stability radius' of this schedule in comparison with the other $k-1$ schedules that have been constructed. The main issue from our experiments is that an optimal schedule is usually stable: Its stability radius is not equal to zero, and so there exists a ball with the center p of the processing times in the space R_+^q of the input data within which the schedule remains optimal. Thus, such a radius may be useful as a measure of the stability of an optimal schedule. Moreover, on the basis of the above computational experiments (though limited problem sizes), one can give the conclusion that an optimal schedule for criterion $\mathcal{C}_{max}$ is often more stable than an optimal schedule for criterion $\sum \mathcal{C}_i$ when the size of the problem is small. Moreover, our approach gives not only the exact value or a bound for the stability radius but also competitive schedules (competitive digraphs) which along with an optimal schedule have to be considered as candidates for the practical realization when the stability radius or its upper bound is less than the possible error of the processing times known in advance. Note that the problem of calculating the stability radius of the digraph $G_s(p)$ is NP-hard even provided that an optimal schedule s is known. It is even NP-hard to find the 'tolerances' of a single processing time p_{ij}, which do not violate the optimality of an optimal digraph. The

latter result follows from [276] since the problem considered in that paper may be presented as a special case of the job shop problem. Another insight is that an optimal mean flow time schedule is usually uniquely determined while two or more optimal makespan schedules are very usual (at least in our simulation study). So, in the latter case it makes sense to look for an optimal makespan schedule with the largest value of the stability radius (the difference of the stability radii for different optimal schedules of the same problem may be very large for the makespan criterion). Such a schedule has a better chance to be makespan optimal in its practical realization. However, this is not valid for the mean flow time criterion, for which one can be satisfied by the first constructed optimal schedule because, even if there are two or more optimal mean flow time schedules, they usually have the same value (or close values) of the stability radii.

Moreover, there exist shops for which we can look for an optimal makespan schedule with an infinitely large stability radius. In particular, if one can influence the properties of the shop (i.e., technological routes of the jobs, the number of used machines and the distribution of the operations to the machines, etc.), one can design a shop that has an optimal makespan schedule with an infinitely large stability radius (see Theorem 1.3). In this case, the variations of the processing times have no influence on such a schedule to be optimal. For some scheduling problems, such a property may be practically important.

Since a zero stability radius of the optimal schedule s is rather seldom, there exists an $\epsilon > 0$ such that s will remain optimal for any variations $p_{ij} \pm \epsilon$ of the processing times. In particular, this is true for almost all problems with the mean flow time criterion, which were considered in our experiments since for these problems an optimal schedule is often uniquely determined, and as a result, it has a strictly positive stability radius. On the other hand, it has a practical sense to make the error in the determination of the processing times as small as possible in order to guarantee the real optimality of a schedule at hand: Almost in all series there were schedules with very small (but strictly positive) values of the stability radii. After the analysis of the influence of possible changes of the given processing times of the operations, i.e., the largest quantity of independent variations (stability radius) within which an optimal schedule of problems $\mathcal{J}//\mathcal{C}_{max}$ and $\mathcal{J}//\sum\mathcal{C}_i$ remains optimal, we perform experimental investigations of job shop problems with *uncertain* processing times $\mathcal{J}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ and $\mathcal{J}/a_i \leq p_i \leq b_i/\sum\mathcal{C}_i$, which satisfy Assumptions 1, 2 and 3 (see Introduction). These computational results are described in Section 2.7.

1.7. Comments and References

In this section, the main attention is paid to a stability analysis. We present references, where the results of this chapter have been originally obtained. Then we discuss different aspects of stability and sensitivity analysis for scheduling problems and outline how the classical questions of sensitivity analysis are considered in this book. A sketch of some other close results is also presented in order to emphasize the generality and difference between the approach presented in this book and others presented in the OR literature. Finally, we describe some trends in scheduling research which aim to make it more relevant to production scheduling in real-world processes.

1. Surveys on stability analysis in sequencing and scheduling were given in [93, 161, 162, 310, 316, 317, 322, 325, 336, 337, 338, 339]. An annotated bibliography for stability analysis in integer programming and combinatorial optimization was given in [145, 146]. This chapter was based on the definitions of the stability radius of an optimal schedule being introduced in [17, 304, 306, 307, 311, 319].

Any shop-scheduling problem with a regular criterion and forbidden operation preemptions can be represented as an extremal problem on a mixed (disjunctive) graph [1, 51, 204, 271, 282, 314, 315, 348, 355]. Moreover, there is a one-to-one correspondence between circuit-free digraphs generated by a mixed graph and semiactive schedules. As it was proven in [132], the set of semiactive schedules contains at least one optimal schedule provided that the criterion is regular. It should also be noted that such a digraph is more stable than the corresponding schedule with respect to variations of the job processing times. The advantage of studying the stability of an optimal digraph instead of the stability of an optimal schedule is justified in Section 1.2. That is why the mixed graph model (described in Section 1.1) is used for shop-scheduling problems in Chapters 1 and 2 of this book.

The papers [44, 192, 203, 311, 334] were devoted to the stability of an optimal digraph $G_s(p)$ which represents an optimal solution to problem $\mathcal{G}//\Phi$. Section 1.2 was based on the papers [306, 311]. In [311], the calculation of the stability radius $\varrho_s(p)$ of an optimal schedule s was reduced to a nonlinear programming problem. Formulas for calculating the stability radius of an optimal schedule with the $\mathcal{C}_{max}$ criterion and the characterization of the extreme values of $\widehat{\varrho}_s(p)$ were proven in [311] (see Section 1.3). The same questions for the mean flow time criterion were considered in [44]. Necessary and sufficient conditions for equality $\widehat{\varrho}_s(p) = 0$ and the characterization of an infinitely large stability radius were proven in [311]. In [191, 192], it

was shown that for problem $\mathcal{J}//\mathcal{C}_{max}$, there exist necessary and sufficient conditions for $\hat{\varrho}_s(p) = \infty$ which can be verified in $O(q^2)$ time. In [192], the analogies to Theorems 1.2 and 1.3 for the job shop problem $\mathcal{J}//\mathcal{L}_{max}$ with minimizing maximum lateness (with respect to the due dates given for the jobs $J_i \in J$) were proven. It was also shown that there does not exist an optimal schedule s with $\varrho_s(p) = \infty$ for all other regular criteria, which are considered in scheduling theory (see [204] for the definitions of different regular criteria). The extreme values of $\overline{\varrho}_s(p)$ were studied in [44, 337]. Necessary and sufficient conditions for equality $\overline{\varrho}_s(p) = 0$ were derived in [44] (see Section 1.4). The stability of an optimal schedule for a job-shop problem with two jobs was investigated in [332, 333], where polynomial geometric algorithms developed in [5, 50, 157, 305, 308, 350] were used.

Within this book, we mainly consider two regular criteria: $\mathcal{C}_{max}$ and $\sum \mathcal{C}_i$. It is useful to give the following comments about them. Although makespan minimization is widely considered in scheduling theory, its importance for production scheduling seems to be less than mean flow time minimization. Indeed, $\mathcal{C}_{max}$ aims for minimizing the schedule duration (length), however, in industry this duration is often defined on the *periodicity* of the process: A working day, a working week or another planning interval. On the other hand, the criterion $\sum \mathcal{C}_i$ is more adequate to such a periodical scheduling environment. Note also that it is often more complicated to analyze $\sum \mathcal{C}_i$ than $\mathcal{C}_{max}$, e.g., the two-machine problems $\mathcal{F}2//\mathcal{C}_{max}$, $\mathcal{J}2/n_i \leq 2/\mathcal{C}_{max}$ and $\mathcal{O}2//\mathcal{C}_{max}$ are polynomially solvable (i.e., *there exist polynomial-time algorithms for them*) but problems $\mathcal{F}2//\sum \mathcal{C}_i$ and $\mathcal{O}2//\sum \mathcal{C}_i$ are NP-hard (see, e.g., [204, 355]), which means that *up to now, there do not exist efficient (polynomial-time) algorithms for their solution and moreover, the construction of a polynomial-time algorithm for at least one of them in the future is unlikely.* The differences in the complexity of the $\mathcal{C}_{max}$ and $\sum \mathcal{C}_i$ criteria, probably, is one of the reasons why in academic research criterion $\mathcal{C}_{max}$ is mainly considered for multi-stage systems (job, flow and open shop) while criterion $\sum \mathcal{C}_i$ for single-stage systems, i.e., systems with only one machine or with a set of *parallel* (usually identical) machines.

In this chapter, we studied the main stability analysis question: *What are the limits to processing time changes such that the schedule at hand remains optimal?* Of course, other numerical parameters of a practical scheduling problem may also be changeable. It is easy to see that, due to the generality of the mixed graph model with any regular objective function presented in Section 1.1, one can analyze other changeable parameters (like *release times, due dates, deadlines, job weights, set-up* and *removal times*, etc.) in terms

of the mixed graph G. (E.g., introduction of a dummy operation in G which proceeds the first operation of a job allows one to consider the *processing* time of this dummy operation as a *release* time of this job.) Nevertheless, for simplicity in what follows, we shall continue to consider operation durations as the only changeable parameters. It is assumed that all the processing times are simultaneously and independently changeable. In order to study specific changes of different numerical parameters (e.g., the case when only one parameter is changeable) in Chapter 2, we introduce and study the more general notion of the so-called *relative stability radius*. Next, we briefly discuss two other classical sensitivity analysis questions which may have both theoretical and practical importance (see [155]). *Given a specific change of one numerical parameter of a scheduling problem, what is the new optimal value of the objective function?* Answering this question identifies the effect of parameter changes on the objective function value. This answer is a solution to the *evaluation version* of a scheduling problem, while an answer to the following classical sensitivity analysis question is a solution to the *optimization version* of a scheduling problem: *Given a specific change of a numerical parameter, what is the new optimal schedule?* It is obvious that a solution to the latter question (which is a subject of Chapters 2 and 3) implies a solution to the former question, however, the converse is not true. Regarding the latter question, how a schedule changes may be of interest. Similarly, as for the stability radius, both these questions may be used when several numerical parameters of a scheduling problem change simultaneously. In Chapter 2, we consider problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$ the solution of which gives an answer to all three classical sensitivity analysis questions.

2. Scheduling theory has received a lot of attention among OR practitioners, management scientists, production and operations research workers and mathematicians since the early 1950s. The term "scheduling theory" has been introduced by Bellman in the article [30], where a first list of definitions and conditions has been given for deterministic scheduling as a branch of applied mathematics. The current state of scheduling theory was presented in the following books and book chapters [36, 37, 51, 81, 204, 269, 270, 299, 331, 337, 352, 353, 355]. As it was noted in the books [37, 81, 189, 269, 270, 331] and in many surveys (e.g., [24, 26, 89, 152, 142, 148, 162, 169, 235, 241, 243, 264, 273, 283, 310]), the utilization of deterministic scheduling theory in many production environments is restricted. MacCarthy and Liu [235] aimed the gap between scheduling theory and scheduling practice. They also discussed some research issues which attempt to make scheduling theory more useful in practice.

Next, we describe some trends in scheduling research which aim to make it more relevant to production scheduling and more applicable to real-world processes. For an uncertain scheduling environment *stochastic* models are introduced, where the processing times (and some other parameters) are assumed to be random variables with known probability distributions. For example, such stochastic models for a single machine with the minimization of mean flow time were considered by Chand et al. [64], by Li and Cao [226], and with the minimization of earliness-tardiness penalties by Cai and Tu [60] as well as by Robb and Rohleder [279]. Since it is possible for a company to estimate the times at which jobs are expected to arrive, Chand et al. [64] developed a decomposition approach such that a large problem can be solved by combining optimal solutions of several smaller problems. The model of Robb and Rohleder [279] consists of a probabilistic dynamic scheduling problem with non-regular performance measures. Using simulation, they explored the robustness of the heuristics with respect to uncertainty in the durations of the operations. Jain and Meeran [169] presented a concise overview of job shop scheduling techniques and the best computational results obtained. The deterministic job shop problem was investigated from a variety of perspectives subclass of this problem in which the objective is the minimization of makespan, by providing an overview of the history, the techniques used and the researchers involved.

Scheduling problems with *controllable processing times* have received an increasing attention during the last two decades (see book [129]). It is often assumed that the actual possible processing time of a job can be continuously *controlled* by a decision-maker, and it can be any number in the given interval. Recent results for scheduling problems with controllable processing times were presented, e.g., in [71, 72, 87, 173, 187, 296, 297].

Traditional scheduling procedures consider static and deterministic future conditions even though this may not be the case in actual scheduling problems. After a description, the preplanned schedule can become inapplicable to the new conditions. As Graves [144] stated, in practice, there is no pure scheduling problem but rather a *rescheduling problem.* Responding to such dynamic factors immediately as they occur is called *real-time scheduling.* An on-line simulation methodology was proposed by Davis and Jones [91] to analyze several scheduling rules in a stochastic job shop. The job shop rescheduling problem is considered as a particularly hard combinatorial optimization problem [263]. The production rescheduling problem deals with uncertainty caused by the exterior business environment and interior production conditions. Since it has practical applications, the rescheduling

problem was studied by many authors (see, e.g., [225, 263, 349]).

A *reactive scheduling approach* was developed by Smith et al. [300], which uses different knowledge sources and aims to make decisions faster with less emphasis on optimality. For knowledge-based systems, the most difficult operation is to decide which knowledge source has to be activated. A discussion of knowledge-based reactive scheduling systems can be found in Blazewicz et al. [36] as well as Szelke and Kerr [349]. Bean et al. [29] proposed a 'match-up' heuristic method for scheduling problems with disruptions. They showed that assuming enough idle time is present in the original schedule and disruptions are sufficiently spaced over time, the optimal rescheduling strategy is to match-up with the preschedule at some time in the future. The objective in [6] was to create a new schedule that is consistent with the order production planning decisions like material flow, tooling and purchasing by utilizing the time critical decision-making concept. When a machine breakdown forces a modified flow shop out of the prescribed state, the proposed strategy reschedules a part of the initial schedule to match-up with the preschedule at some point.

Fuzzy scheduling techniques proposed in the literature either fuzzify directly the existing scheduling rules, or solve mathematical programming problems to determine optimal schedules [101, 133, 143, 167, 196, 262, 284]. The optimality of a fuzzy logic alternative to the usual treatment of uncertainties in a scheduling system using probability theory was examined by Ozelkan and Duckstein [262]. The purpose of that article was to investigate necessary optimality conditions of fuzzy counterparts of classical dispatching rules, such as the shortest processing time (SPT) and the earliest due date (EDD). Dumitru and Lubau [101] proposed fuzzy mathematical models to solve the job shop problem. Grabot and Geneste [143] used a fuzzy rule-based approach to find a compromise between different job shop dispatching rules. Kuroda and Wang [196] also analyzed fuzzy job shop problems using a branch-and-bound algorithm to obtain results for lateness related criteria. A mathematical programming approach to a single machine problem with *fuzzy precedence relation* was given in [167]. Job shop scheduling with both fuzzy processing times and fuzzy due dates was considered in [284]. Sakawa and Kubota [284] formulated a multi-objective fuzzy job shop problem as a three-objective one which not only maximizes the minimum agreement index but also maximizes the average agreement index and minimizes the maximum fuzzy completion time. Slowinski and Hapke [299] collected the main works in this field.

Decision-makers often consider multiple objectives when making schedul-

ing decisions. However, very little research has been done in multiple machine environments with *multiple objectives* [15, 70, 75, 74, 73, 149, 153, 198, 224, 358]. Allahverdi and Mittenthal [15] considered a two-machine flow shop scheduling problem, where machines suffer random breakdowns and processing times are constant, with respect to both the makespan and the maximum lateness objective functions. Kyparisis and Koulamas [198] and Gupta et al. [153] studied the two-machine open shop problem with a hierarchical objective: Minimize the total completion time subject to minimum makespan $\mathcal{O}2//\sum \mathcal{C}_i|\mathcal{C}_{max}$. Moreover, Gupta et al. [151], Gupta et al. [149] and T'kindt et al. [358] presented heuristic and exact algortihms for the two-machine flow shop problem with a hierarchical objective: $\mathcal{F}2//\sum \mathcal{C}_i|\mathcal{C}_{max}$. Cheng and Shakhlevich [75] considered a special class of flow shop problems, known as the *proportionate flow shop*. In such a shop, each job flows through the machines in the same order and has equal processing times on the machines. It was assumed that all operations of a job may be compressed by the same amount which will incur an additional cost. The objective was to minimize the makespan of the schedule together with a compression cost function which is non-decreasing with respect to the amount of compression. A *bi-criterion approach* to solve the single machine scheduling problem in which the job release dates can be compressed while incurring additional costs, was considered in [74]. Stein and Wein [344] gave a proof that, for any instance of a rather general class of scheduling problems, there exists a schedule with a makespan at most twice that of the optimal value and of a total weighted completion time at most twice that of the optimal value. In [245], the stability of an optimal schedule for the two-machine flow shop problem with the makespan criterion $\mathcal{F}2//\mathcal{C}_{max}$ was considered. Results on multi-criteria scheduling obtained before 2002 were presented in the book [357].

The papers above addressed problems of practical importance in planning, scheduling, and control. It is important to produce schedules that are both stable (robust) and adaptable to system disturbances. More important, it offers unique properties that lead to more effective planning and control methods for systems under uncertainty. An important step in the process of designing a railway station track layout is the verification of the robustness of the layout with respect to the timetables it is based on [259]. Odijk [259] developed an algorithm to randomly perturb a given timetable such that the perturbation is feasible and has the same structure as the given timetable. He studied the problem of generating solutions uniformly at random for a given set of integer variables and a given set of binary relations

stating minimal and maximal differences between the variables.

In general, studying a scheduling problem with *uncertain processing times* and its *sensitivity analysis* is of importance. Reasons for that can be illustrated by giving references to practical applications. In many cases the data used are imprecise due to an uncertainty with respect to the exact parameter values or due to errors in the measurement. In industrial applications of mathematical programming models, there are almost always uncertain elements that are assumed away or suppressed in the formal description of the model. Wagner [363] gave reasons why imprecise parameter values are used in practice. The suggested approaches are kindred to familiar multi-variable statistical techniques, such as stepwise regression, that have required an empirical testing to establish their practical value. Wagner [363] provided practitioners with a quick start in performing a global sensitivity analysis.

Heuristic methods such as simulated annealing, genetic algorithms and tabu search take up a growing space in the literature on the job shop scheduling problem (see [35] for an overview). Blazewicz et al. [35] surveyed wide solution techniques available for solving certain types of job shops. Pezzella and Marelli [266] presented a computationally effective heuristic method based on a tabu search technique and on the shifting bottleneck procedure [2, 27] for solving the minimum makespan job shop problem $\mathcal{J}//\mathcal{C}_{max}$.

Multi-objective discrete decision-making models have been widely applied in design, management, economics, and other applied fields. One of the directions in investigating these problems was the analysis of the stability of solutions to perturbations of the initial parameters. Various aspects of the stability of vector discrete optimization problems were studied in [58, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 229] by Emelichev et al. In these and many other close papers, the stability of a vector discrete optimization problem can be considered as a discrete analog of the property of an optimal mapping specifying the Pareto choice function being upper semi-continuous in the Hausdorff sense (i.e., there exists a neighborhood in the space of the initial parameters such that new Pareto optima are impossible to arise inside it). The Pareto set within this neighborhood can only become narrower. The concept of stability is treated as the existence of a neighborhood of the initial parameters in the space of the numerical input data such that for each perturbation, there exists at least one efficient solution to the initial problem that retains Pareto optimality. This type of stability was first investigated in [215] for a single-criterion traveling salesman problem, in [135] for a shortest path problem, in [136] for a bottleneck problem and in [134, 137, 138, 139] for Boolean trajectory problems.

Table 1.5: *Notations for the general shop scheduling problem*

Symbols	Description
Q	Set of operations: $Q = \{1, 2, \ldots, q\}$
Q_k	Set of operations which has to be processed by machine $M_k \in M$: $Q = \cup_{k=1}^{m} Q_k$ and $Q_k \cap Q_l = \emptyset$, if $k \neq l$
Q^{J_i}	Set of operations for processing job $J_i \in J$: $Q = \cup_{i=1}^{n} Q^{J_i}$, $Q^{J_i} \neq \emptyset$ and $Q^{J_i} \cap Q^{J_j} = \emptyset$, if $i \neq j$
$w(i)+1$	First operation of job J_i, $1 \leq i \leq n$, where $w(i) = \sum_{k=0}^{i-1} n_k$ and $n_0 = 0$
$w(i)+n_i$	Last operation of job J_i, $1 \leq i \leq n$
s_i	Starting time of operation $i \in Q$
c_i	Completion time of operation $i \in Q$
p_i	Processing time of operation $i \in Q$
a_i	Lower bound for the processing time of operation i
b_i	Upper bound for the processing time of operation i
$c_i(s)$	Earliest completion time of operation $i \in Q$ in the semiactive schedule s
$s \in S$	Semiactive schedule $s = (c_1(s), c_2(s), \ldots, c_q(s))$
$\tilde{H}$	Set of paths in digraph $(Q, A, \emptyset)$
$\tilde{H}_s$	Set of paths in digraph $G_s \in \Lambda(G)$
H	Set of dominant paths in digraph $(Q, A, \emptyset)$
H_s	Set of dominant paths in digraph $G_s \in \Lambda(G)$
$H_k(p)$	Set of critical dominant paths in digraph $G_k \in \Lambda(G)$ with respect to vector p : $H_k(p) \subseteq H_k$
l_k^p	Critical weight of digraph $G_k \in \Lambda(G)$ with the vector p of processing times: $l_k^p = \max_{\mu \in H_s} l^p(\mu)$
$\tilde{H}_k^i$	Set of paths in digraph G_k ending in vertex $r \in Q^{J_i}$, r denotes the last operation of job J_i: $r = w(i) + n_i$
H_k^i	Subset of all dominant paths in set $\tilde{H}_k^i$
H_{sk}	Subset of paths of set H_s, which are not dominated by paths from set H_k
$H_{sk}(T)$	$H_{sk}(T) = \{\mu \in H_s :$ There is no path $\nu \in H_k$ which dominates path μ in polytope $T\}$

Chapter 2

General Shop with Interval Processing Times

It usually takes me more than three
weeks to prepare a good impromptu speech
Mark Twain

In [269], it was noted that one "source of uncertainty is processing times, which, typically, are not known in advance. Thus, a good model of a scheduling problem would need to address these forms of uncertainty." In the stochastic settings of scheduling problems considered in the second part of [269], the random processing time of an operation is assumed to take a known probability distribution when dealing with uncertain scheduling environments mainly for single-stage systems. This chapter deals with general shop and job shop scheduling problems with the objective to minimize the makespan or mean flow time provided that the numerical input data are uncertain. The scheduling environments that we consider are so uncertain that all information available about the processing time of an operation is its upper and lower bounds. To be more specific, we consider problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$ and problem $\mathcal{J}/a_i \leq p_i \leq b_i/\Phi$. Such problems may arise in many practical situations since, even if no specific bounds for an uncertain processing time p_i are known in advance, one can set $a_i = 0$ and b_i equal to the planning horizon. In spite of practical interest, such a type of scheduling problems was considered in a limited OR literature so far.

Most algorithms developed in this chapter are time-consuming. So, we assume that an optimization may be performed under less time pressure and with less time cost. As such a practical example, one can consider a production shop, where the schedule for the next day can be run overnight.

Usually, time constraints during the night are less severe than those arising during the following day's production process. On the following day, one may need a quick evaluation of the effect of processing time changes on the optimal schedule. Consequently, one may be willing to use a less efficient algorithm overnight, if it allows one to answer stability questions more quickly during the following day. As in Chapter 1, we use a mixed graph model for representing the input data, the scheduling process and the final solution.

Our 'strategy' is to separate the 'structural' input data from the 'numerical' input data as much as possible. The precedence and capacity constraints (i.e., the structural input data) are given by the mixed graph G, which completely defines the set of semiactive schedules. The set of optimal semiactive schedules is defined by the weighted mixed graph $G(p)$ which presents both the structural and numerical input data. In Section 2.1, we demonstrate some preliminary ideas of our approach using the results from Chapter 1 for solving an example of a job shop problem $\mathcal{J}3/n = 2, a_i \leq p_i \leq b_i/C_{max}$. In Section 2.1, we define a G-solution of problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$ as a set $\Lambda^*(G)$ of dominant digraphs with the following property: For any fixed vector p of the processing times with the components p_i in the intervals $[a_i, b_i],\ i \in Q$, there exists at least one optimal digraph in set $\Lambda^*(G)$. Section 2.2 deals with the mathematical background for later presentations. In Section 2.3, we present the main formulas and an algorithm for solving problem $\mathcal{G}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ and show how to restrict the number of digraphs which have to be considered for calculating the stability radius. In Sections 2.1 – 2.3, we present an approach to deal with problem $\mathcal{G}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ based on an improved stability analysis of an optimal schedule and demonstrate this approach on an example of the job shop problem $\mathcal{J}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$. In the course of this chapter, an *optimal* schedule (digraph), a *better* and a *best* schedule (digraph) are considered with respect to criterion $\mathcal{C}_{max}$ (Sections 2.1 – 2.6) or criterion $\sum \mathcal{C}_i$ (Sections 2.4 – 2.6). All necessary notions from Chapter 1 are generalized for problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$ for a more effective use.

2.1. Minimal Dominant Set

We consider a general shop scheduling problem as described in Section 1.1. In a *deterministic* setting of the scheduling problem, the processing times p_i are assumed to be known exactly for all operations $i \in Q$, and as it was mentioned in Chapter 1, a schedule is defined by the starting times s_i or completion times c_i of all operations $i \in Q$. Unfortunately, in many practical

cases, the operation processing times are unknown before scheduling or only known with some error. In such an uncertain environment, it is not possible to determine a priori the starting times or completion times of all operations.

Let the input data of a general shop scheduling problem be presented by a mixed graph $G = (Q, A, E)$ introduced in Section 1.1. If the processing times of all operations Q are known exactly before scheduling, we associate a non-negative real weight p_i (operation duration) with each vertex $i \in Q$ in the mixed graph $G = (Q, A, E)$ to obtain the weighted mixed graph denoted by $G(p) = (Q(p), A, E)$. As in Chapter 1, set $\Lambda(G) = \{G_1, G_2, \ldots, G_\lambda\}$ denotes the set of all feasible (i.e., circuit-free) digraphs $G_s = (Q, A \cup E_s, \emptyset) \in \Lambda(G)$ generated from the mixed graph $G = (Q, A, E)$ via substituting its edges E by the arcs E_s. Given a vector $p = (p_1, p_2, \ldots, p_q)$ of the processing times, the weighted digraph $G_s(p) = (Q(p), A \cup E_s, \emptyset)$ corresponding to a feasible digraph $G_s = (Q, A \cup E_s, \emptyset) \in \Lambda(G)$ uniquely defines the *earliest completion time* $c_i(s)$ of each operation $i \in Q$ along with the makespan value $\max\{c_i(s) : i \in Q\}$ of schedule s. The running time of calculating $c_1(s)$, $c_2(s), \ldots, c_q(s)$ for the given weighted digraph $G_s(p)$ may be restricted by $O(q^2)$, where q is the number of operations: $q = |Q|$. The maximal weight of a path in the weighted digraph $G_s(p)$ (called a *critical weight*) defines the makespan, $\max\{c_i(s) : i \in Q\}$, of schedule s. A path in $G_s(p)$ with a critical weight is called a *critical path.* As already mentioned in Section 1.1, given a fixed vector $p = (p_1, p_2, \ldots, p_q)$ of the processing times, in order to construct an optimal schedule for the general shop problem $\mathcal{G}//\mathcal{C}_{max}$, one may first enumerate (explicitly or implicitly) all *feasible* digraphs $G_1(p)$, $G_2(p), \ldots, G_\lambda(p)$ and then select an *optimal* digraph, i.e., one with a minimal value of the critical weight among all λ feasible digraphs.

It is worthwhile to note that the *feasibility* of a schedule s (and the *feasibility* of a weighted digraph $G_s(p)$) is independent of the vector $p = (p_1, p_2, \ldots, p_q)$ of the processing times, while the *optimality* of a weighted digraph $G_s(p)$ depends on the vector p. In other words, the set $\Lambda(G) = \{G_1, G_2, \ldots, G_\lambda\}$ of feasible digraphs is completely defined by the mixed graph $G = (Q, A, E)$ (without weights p) while the information on the vector p of the processing times is needed to determine whether a schedule $k \in S$ is optimal or not, i.e., the optimality of a schedule is defined by the weighted mixed graph $G(p) = (Q(p), A, E)$. As in Chapter 1, $S = \{1, 2, \ldots, \lambda\}$ denotes the set of all semiactive schedules (see Definition 1.1 on page 22).

If the vector p of processing times is not known accurately before applying a scheduling procedure (e.g., the processing times may *vary* in a realization of a schedule), different realizations may result in different critical paths in

the weighted digraph $G_s(p)$. For practical problems, the cardinality λ of the set $\Lambda(G)$ may be huge (the obvious upper bound $\lambda \leq 2^{|E|}$ could be reached). However, as we shall show, we often only need to consider some subset B of the set $\Lambda(G)$: $B \subseteq \Lambda(G)$. Since $p_i \geq 0$ for all $i \in Q$, we obtain the equality $\max_{i \in Q} c_i(s) = \max_{\mu \in H_s} l^p(\mu)$, and from equality (1.16) (see page 32) it follows that digraph $G_s(p)$ has the minimal critical weight within the set $B \subseteq \Lambda(G)$ if and only if

$$\max_{\mu \in H_s} l^p(\mu) = \min_{G_k \in B} \max_{\nu \in H_k} l^p(\nu). \tag{2.1}$$

For the case $B = \Lambda(G)$, equality (2.1) provides an *optimality* criterion of a schedule $s \in S$ (if the vector p of processing times is fixed).

In this chapter, we allow the duration p_i of an operation $i \in Q$ to assume any value in the fixed closed interval $[a_i, b_i]$, $0 \leq a_i \leq b_i$, defined by inequalities (4) (see Condition 5 on page 15). Such a general shop problem with *uncertain* processing times is denoted by $\mathcal{G}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$. As it was already mentioned, problem $\mathcal{G}//\mathcal{C}_{max}$ (with *fixed* processing times) is a special case of the general shop problem $\mathcal{G}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ with *uncertain* processing times (namely, when $a_i = b_i$ for each operation $i \in Q$). In what follows, we call problem $\mathcal{G}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ uncertain in contrast to problem $\mathcal{G}//\mathcal{C}_{max}$ called deterministic.

One can interpret p_i in the problem $\mathcal{G}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ as a random variable x_i with the following cumulative probability distribution:

$$F_i(t) = P(x_i < t) = \begin{cases} 0, & \text{if } t < a_i, \\ 1, & \text{if } t = b_i. \end{cases}$$

The density function $f_i(t)$ of such a cumulative probability distribution is uncertain in the closed interval $[a_i, b_i]$ for operation $i \in Q$:

$$f_i(t) = \frac{dF_i(t)}{dt} = \begin{cases} 0, & \text{if } t < a_i, \\ ? & \text{if } a_i \leq t \leq b_i, \\ 0, & \text{if } t > b_i. \end{cases}$$

In the framework of stochastic scheduling ([269], pp. 167 – 252), each random variable x_i associated with the processing time of operation $i \in Q$ (and perhaps similar random variables associated with release times, due dates, setup times, etc.) is assumed to have a *known* probability distribution. For example, the stochastic version of problem $\mathcal{G}//\mathcal{C}_{max}$ with exponential continuous time distributions with rates α_i, $i \in Q$, is denoted by $\mathcal{G}/p_i \sim exp(\alpha_i)/E\mathcal{C}_{max}$, where the density function of an exponentially distributed random variable x_i is $f_i(t) = \alpha_i e^{-\alpha t}$, the corresponding probability

distribution is $F_i(t) = P(x_i < t) = 1 - e^{-\alpha t}$, and

$$E_i(x_i) = \int_0^\infty t f_i(t)dt = \int_0^\infty t dF_i(t) = \frac{1}{\alpha_i}$$

is the expected (mean) value of the random processing time x_i. The objective of problem $\mathcal{G}/p_i \sim exp(\alpha_i)/E\mathcal{C}_{max}$ is to minimize the expected value of the makespan $E\mathcal{C}_{max}$ for a schedule which may be constructed using an appropriate *scheduling policy*.

The approach we present in this chapter for solving the uncertain problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$ is based on an improved stability analysis of an optimal digraph (see Sections 1.2 – 1.5). As follows from Chapter 1, an optimal digraph $G_s \in \Lambda(G)$ provides a solution of the deterministic problem $\mathcal{G}//\mathcal{C}_{max}$. In other words, an optimal digraph defines a set of m sequences of the operations Q_k processed by machine $M_k, k = 1, 2, \ldots, m$, with a minimal value of the makespan among all feasible schedules when the vector $p = (p_1, p_2, \ldots, p_q)$ of the processing times is fixed.

Since the vector $p \in T$ of processing times is unknown in the general case of the uncertain problem $\mathcal{G}/a_i \le p_i \le b_i/\Phi$ when there exist jobs $J_i \in J$ with $a_i < b_i$, the completion time of such jobs $J_i \in J$ cannot be calculated before scheduling. Therefore, mathematically, the uncertain problem $\mathcal{G}/a_i \le p_i \le b_i/\Phi$ is not correct. In the OR literature, different approaches for correcting the optimization problem like $\mathcal{G}/a_i \le p_i \le b_i/\Phi$ have been used. In this chapter, we propose to use a G-solution to the uncertain problem $\mathcal{G}/a_i \le p_i \le b_i/\Phi$ defined as a *dominant set of digraphs (schedules)*.

Recall that T is the polytope in the space R^q_+ (with the maximum metric) defined by inequalities (4) on page 15: $T = \{x = (x_1, x_2, \ldots, x_q) : \ a_i \le x_i \le b_i, \ i \in Q\}$. The set $\Lambda^*(G) \subseteq \Lambda(G)$ of feasible digraphs will be called a G-solution to problem $\mathcal{G}/a_i \le p_i \le b_i/\Phi$ if this set contains at least one optimal digraph for each fixed vector $x \in T$ of the processing times. Note again that instead of considering a set of digraphs $\Lambda^*(G)$, we can consider directly a set of schedules $S^* \subseteq S = \{1, 2, \ldots, \lambda\}$ which can be induced by the set $\Lambda^*(G)$: $S^* = \{k \ : \ G_k \in \Lambda^*(G)\}$.

Obviously, the whole set $\Lambda(G)$ may be considered as a G-solution to problem $\mathcal{G}/a_i \le p_i \le b_i/\Phi$ for any given polytope $T \subseteq R^q_+$, i.e., for each pair of vectors $a = (a_1, a_2, \ldots, a_q) \in R^q_+$ and $b = (b_1, b_2, \ldots, b_q) \in R^q_+$ with $a_i \le b_i, \ i = 1, 2, \ldots, q$. However, such a G-solution $\Lambda(G)$ is usually redundant: Polytope T may contain no point p, where some digraph from set $\Lambda(G)$ is optimal. Moreover, the construction of the whole set $\Lambda(G)$ is only possible for a small size of the scheduling problem since the cardinality λ of the set

$\Lambda(G)$ could be equal to $2^{|E|}$. Note also that during the realization of a schedule, a decision-maker may have difficulties while dealing with such a vast set of possible candidates for a single realization of a schedule. Therefore, it is practically important to look for a 'minimal G-solution' $\Lambda^*(G) \subseteq \Lambda(G)$ for the uncertain problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$. A set $\Lambda^*(G)$ is a minimal (with respect to inclusion) G-solution if any proper subset of set $\Lambda^*(G)$ is not a G-solution. Note that a minimal G-solution $\Lambda^*(G)$ may be not unique, e.g., since there may exist two or more optimal digraphs for some vector $p \in T$ of the processing times. We combine these arguments as follows.

Definition 2.1 *A set of digraphs $\Lambda^*(G) \subseteq \Lambda(G)$ is called a G-solution of problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$ if for each fixed vector $p \in T$ of the processing times the set $\Lambda^*(G)$ contains at least one optimal digraph. If any proper subset of the set $\Lambda^*(G)$ is no longer a G-solution of problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$, it is called a minimal G-solution denoted as $\Lambda^T(G)$.*

Table 2.1 summarizes different settings of a general shop scheduling problem with the criterion $\mathcal{C}_{max}$ in accordance with the information which is available for a vector p of the processing times. These problems are classified on the basis of the mixed graph approach. Row 1 in Table 2.1 refers to the *mass* general shop scheduling problem, where the only information requirement on the processing times p is that vector p belongs to the space R^q_+. Thus, the individual uncertain problem $\mathcal{G}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ turns out into a mass general shop problem if b_i is assumed to be ∞ and $a_i = 0$ for each operation $i \in Q$.

Table 2.1: *Scheduling with different requirements on the numerical data*

	Scheduling problem	Input data	Semiactive schedules	Solution (G-solution)
1	Mass general shop problem	$G(p) = (Q(p), A, E)$; $0 \leq p_i < \infty,\ i \in Q$	$\Lambda(G)$	$\Lambda(G)$
2	Individual problem $\mathcal{G}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$	$G(p) = (Q(p), A, E)$; $a_i \leq p_i \leq b_i,\ i \in Q$	$\Lambda(G)$	$\Lambda^*(G) \subseteq \Lambda(G)$
3	Individual problem $\mathcal{G}/p_i \sim F_i(t)/E\mathcal{C}_{max}$	$G(x) = (Q(x), A, E)$; $F_i(t) = P(x_i < t)$	$\Lambda(G)$	$\{G_s\} \subseteq \Lambda^*(G) \subseteq \Lambda(G)$
4	Individual problem $\mathcal{G}//\mathcal{C}_{max}$	$G(p) = (Q(p), A, E)$; $a_i = p_i = b_i,\ i \in Q$	$\Lambda(G)$	$G_s \in \Lambda(G)$

For any digraph $G_s \in \Lambda(G)$, it is possible to construct a problem $\mathcal{G}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ with appropriate values a_i and b_i such that digraph G_s will be

optimal in some realization of vector p, $a_i \le p_i \le b_i,\ i \in Q$, *of the processing times.* The above statement is true. Indeed, for each critical path $\mu \in H_s$ we can set p_i equal to a sufficiently small real $\epsilon \ge 0$ for each operation $i \in Q^0 = [\mu] \setminus \cup_{k \ne s} \cup_{\nu \in H_k(p)} [\nu]$, where $\mu \in H_s$ is a critical path in digraph G_s. For such a setting of the processing times, equality (2.1) is satisfied with $B = \Lambda(G)$. If $Q^0 = Q$, we get a trivial individual problem $\mathcal{G}//\mathcal{C}_{max}$ with $p_i = \epsilon = 0,\ i \in Q$, where any digraph G_s from set $\Lambda(G)$ is optimal.

In this chapter, we consider the uncertain problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$ which is rather general (see row 2 in Table 2.1). In one extreme case when $a_i = 0$ and $b_i = \infty$ for each operation $i \in Q$, the uncertain problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$ coincides with the whole mass problem presented in row 1. In the other extreme case when $a_i = b_i$ for each $i \in Q$, the uncertain problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$ is reduced to the deterministic problem $\mathcal{G}//\mathcal{C}_{max}$ (row 4 in Table 2.1), which is one of the basic problems studied in deterministic scheduling theory. The more information about vector p is available before applying a scheduling procedure, the 'better' G-solution may be obtained: The cardinality of a minimal G-solution $\Lambda^T(G)$ is smaller if the polytope T is defined by smaller closed intervals $[a_i, b_i]$. For example, a minimal G-solution (see Definition 2.1) is reduced to a single optimal digraph $G_s \in \Lambda(G)$ in the case of the deterministic problem $\mathcal{G}//\mathcal{C}_{max}$ (row 4 in Table 2.1).

Row 3 refers to the individual stochastic problem $\mathcal{G}/p_i \sim F_i(t)/E\mathcal{C}_{max}$, one of the basic problems studied in stochastic scheduling theory, where each operation $i \in Q$ is assumed to be a random variable with a probability distribution $F_i(t)$ known before applying a scheduling procedure. For problem $\mathcal{G}/p_i \sim F_i(t)/E\mathcal{C}_{max}$, the optimal solution may be a single digraph G_s when one adopts a static scheduling policy ([269], page 178) or a subset of feasible digraphs $\Lambda^*(G)$ when one adopts a dynamic scheduling policy ([269], page 179). When a static scheduling policy is adopted, a decision-maker can use an optimal schedule $s \in S$ which minimizes the expected makespan $E\mathcal{C}_{max}$ and thus, schedule s remains unchanged during the entire process. In the case of a dynamic scheduling policy, an initial schedule s may be constantly revised during the solution process based on the updated information available [269]. We note that a minimal G-solution $\Lambda^T(G)$ for an uncertain problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$, may be calculated exactly before the realization of the process, while for a stochastic problem $\mathcal{G}/p_i \sim F_i(t)/E\mathcal{C}_{max}$ the solution may vary and may even coincide with the whole set $\Lambda(G)$ for some of the probability distributions $F_i(t)$. It is worth to note that for all four formulations presented in Table 2.1, the set of feasible solutions (set of semiactive schedules) remains the same and therefore, the properties of the

feasible digraphs $\Lambda(G)$ are of particular importance for all problem settings presented in Table 2.1. Our approach for solving problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$ is based on the stability property of an optimal digraph which guarantees that a feasible digraph remains optimal after feasible variations of the processing times. To facilitate the presentation of the ideas of our approach for solving an uncertain problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$, we demonstrate it using a small example of a job shop problem with uncertain numerical data.

Example 2.1 *We consider the job shop problem* $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\mathcal{C}_{max}$ *with two jobs* $J = \{J_1, J_2\}$ *and three machines* $M = \{M_1, M_2, M_3\}$. *Job* J_1 *(job* J_2*, respectively) consists of the following set of ordered operations* $\{O_{1,1}, O_{1,2}, O_{1,3}\} = Q^{J_1}$ *(operations* $\{O_{2,1}, O_{2,2}, O_{2,3}\} = Q^{J_2}$*). The assignment of the operations* $\{O_{1,1}, O_{1,2}, O_{1,3}, O_{2,1}, O_{2,2}, O_{2,3}\}$ *to the set of machines* M *is as follows:* $Q_1^J = \{O_{1,1}, O_{2,2}\}$, $Q_2^J = \{O_{1,2}, O_{2,1}, O_{2,3}\}$, $Q_3^J = \{O_{1,3}\}$.

For criterion $\mathcal{C}_{max}$*, it is useful to consider also a dummy operation* O_{ij} *provided that its processing time is equal to zero:* $p_{ij} = 0$. *To accommodate dummy operations in the framework of the mixed graph* $G = (Q^J, A^J, E^J)$*, we assume that each dummy operation 'has to be processed' by a special dummy machine with a zero processing time, and we assume that the number of dummy machines is equal to the number of dummy operations. As a result, each dummy operation turns out to be an isolated vertex in the subgraph* $(Q^J, \emptyset, E^J)$ *of the mixed graph* G. *In Example 2.1, we consider the operations* $O_{0,1}$ *and* $O_{0,2}$ *as dummy operations, and the machines* M_4 *and* M_5 *as dummy machines. Let operation* $O_{0,1}$ *(operation* $O_{0,2}$*) denote the beginning (the end) of a schedule, and so operation* $O_{0,1}$ *precedes all other operations (all other operations precede operation* $O_{0,2}$*). We assume that* $Q_4^J = \{O_{0,1}\}$ *and* $Q_5^J = \{O_{0,2}\}$. *The input data of Example 2.1 is represented by the weighted mixed graph* $G(p) = (Q^J(p), A^J, E^J)$ *given in Figure 2.1, where each processing time* p_{ij} *is presented near the vertex* $O_{ij} \in Q^J$*, and the vector* p *of processing times is defined as follows:* $p = (p_{1,1}, p_{1,2}, p_{1,3}, p_{2,1}, p_{2,2}, p_{2,3}) = (75, 50, 40, 60, 55, 30)$. *(Vector* $p \in R_+^6$ *does not include the processing times of the dummy operations* $O_{0,1}$ *and* $O_{0,2}$*.) For this small example, we can explicitly enumerate all feasible digraphs* $\Lambda(G) = \{G_1, G_2, \ldots, G_5\}$ *with their signatures.*

$$
\begin{aligned}
E_1^J &= \{(O_{1,1}, O_{2,2}), (O_{2,1}, O_{1,2}), (O_{1,2}, O_{2,3})\},\\
E_2^J &= \{(O_{1,1}, O_{2,2}), (O_{2,1}, O_{1,2}), (O_{2,3}, O_{1,2})\},\\
E_3^J &= \{(O_{1,1}, O_{2,2}), (O_{1,2}, O_{2,1}), (O_{1,2}, O_{2,3})\},\\
E_4^J &= \{(O_{2,2}, O_{1,1}), (O_{2,1}, O_{1,2}), (O_{1,2}, O_{2,3})\},\\
E_5^J &= \{(O_{2,2}, O_{1,1}), (O_{2,1}, O_{1,2}), (O_{2,3}, O_{1,2})\}.
\end{aligned}
$$

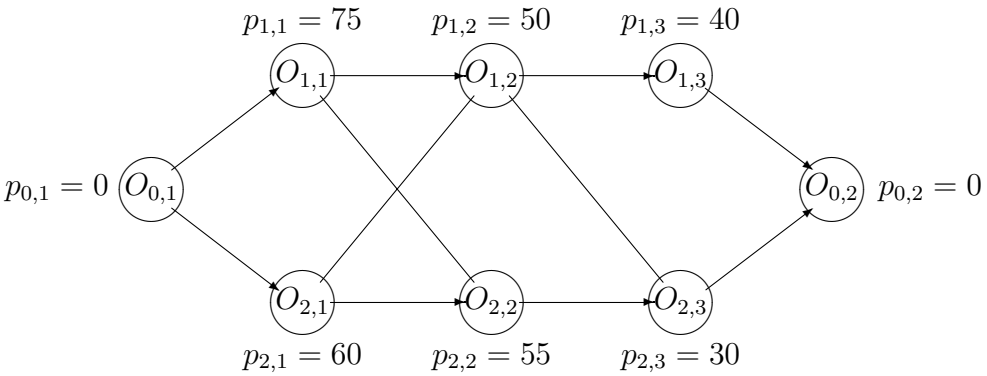


Figure 2.1: *Weighted mixed graph* $G(p) = (Q^J(p), A^J, E^J)$

We calculate the makespans for all feasible digraphs from set $\Lambda(G)$*:*

$$\begin{aligned} l_1^p &= \max\{c_{in_i}(1) : J_i \in J\} = 165, \\ l_2^p &= \max\{c_{in_i}(2) : J_i \in J\} = 250, \\ l_3^p &= \max\{c_{in_i}(3) : J_i \in J\} = 270, \\ l_4^p &= \max\{c_{in_i}(4) : J_i \in J\} = 280, \\ l_5^p &= \max\{c_{in_i}(5) : J_i \in J\} = 280, \end{aligned}$$

and select an optimal weighted digraph $G_1(p) = (Q^J(p), A^J \cup E_1^J, \emptyset)$ *with the signature* $E_1^J = \{(O_{1,1}, O_{2,2}), (O_{2,1}, O_{1,2}), (O_{1,2}, O_{2,3})\}$*. The digraph* $G_1(p)$ *has a minimal critical weight equal to 165 (see Figure 2.2). Using formu-*

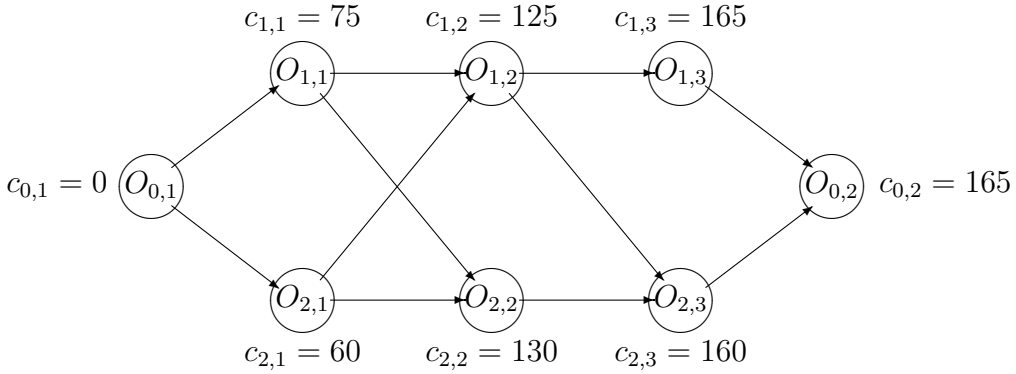


Figure 2.2: *Optimal digraph* $G_1 = (Q^J, A^J \cup E_1^J, \emptyset)$ *with the completion times* c_{ij} *presented near the vertices* $O_{ij} \in Q^J$

las (1.31) and (1.32) given on page 43, we can calculate the stability radius of the optimal digraph $G_1(p)$*. To this end, we compare digraph* $G_1(p)$ *with each digraph* $G_k \in \Lambda(G) \setminus \{G_1\}$*.*

In Table 2.2, we present the calculations with respect to formulas (1.31) and (1.32) in detail. Each path $\mu \in H_{1k} \subseteq H_1$ *in* $G_1(p)$ *presented in column 3 is compared with each path* $\nu \in H_k$ *presented in column 4 provided that*

$l^p(\nu) \geq l_1^p = 165$. The cardinalities of the sets $H_{1k}, k = 2, 3, \ldots, \lambda$, are given in column 2. (Since $H_{1,2} = \emptyset$ and $H_{1,4} = \emptyset$, digraphs G_2 and G_4 are not involved in the computations.) The non-decreasing sequence (1.30) of the processing times $(p^{\nu\mu}_{(0)}, p^{\nu\mu}_{(1)}, \ldots, p^{\nu\mu}_{(\omega_{\nu\mu})})$ defined on page 42, is given in column 5. In column 6, we present the calculations according to the fraction in the formula (1.31) consecutively for each $\beta = 0, 1, \ldots, \omega_{\nu\mu}$. In columns 7, 8 and 9, respectively, we extract the maximum for $\beta = 0, 1, \ldots, w_{\nu\mu}$, the maximum for $\nu \in H_k, l^p(\nu) \geq l_1^p$, and the minimum for $\mu \in H_{1k}$ from the values obtained in column 6. As a result, column 9 presents the values of r_{k1} for the digraphs G_k. The last step is to adapt the formula (1.32) from Theorem 1.4 (see page 43). The minimum value $\hat{r}_{ks}$ is given in column 9.

Due to formula (1.31), we calculate the stability radius $\widehat{\varrho}_1(p) = \min\{30, 30\} = 30$. Thus, digraph G_1 remains optimal for any vector $x = (x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, x_{2,2}, x_{2,3})$ of the processing times if its distance from the given vector $p = (75, 50, 40, 60, 55, 30)$ is no more than 30 : $p_{ij} - 30 \leq x_{ij} \leq p_{ij} + 30$. Therefore, while solving this instance of an uncertain problem $\mathcal{J}3/n = 2, a_i \leq p_i \leq b_i/\mathcal{C}_{max}$, digraph $G_1 = (Q^J, A^J \cup E_1^J, \emptyset)$ remains optimal if all possible variations of the processing times $x = (x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, x_{2,2}, x_{2,3})$ belong to the stability ball $O_{\widehat{\varrho}_1(p)}(p) = O_{30}(p)$, i.e., if the following inequality holds:

$$\max_{O_{ij} \in Q^J} \{x_{ij} - a_{ij},\ b_{ij} - x_{ij}\} \leq 30. \tag{2.2}$$

In such a case, polytope T defined by inequalities (4) in the vector space R_+^6 has to be completely contained in the stability ball $O_{30}(p)$ of the optimal digraph $G_1 : T \subseteq O_{30}(p)$. In other words, digraph G_1 provides a G-solution to the uncertain problem $\mathcal{J}3/n = 2, a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ as long as inequality (2.2) is satisfied: $\Lambda^*(G) = \{G_1\}$. It is clear that this G-solution $\Lambda^*(G)$ is also a minimal G-solution to the uncertain problem $\mathcal{J}3/n = 2, a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ due to Definition 2.1. Thus, in the case when inequality (2.2) holds, a decision-maker needs to use only digraph G_1 from the set $\Lambda(G) = \{G_1, G_2, G_3, G_4, G_5\}$ as an optimal one for any feasible realization of the processing times. In such a case, a G-solution of problem $\mathcal{J}3/n = 2, a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ turns out to be a solution (in the ordinary sense) for any deterministic problem $\mathcal{J}3/n = 2/\mathcal{C}_{max}$ with the fixed vector p of the processing times from the polytope T. The minimal G-solution to the uncertain problem $\mathcal{J}3/n = 2, a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ consists of a single digraph: $\{G_1\} = \Lambda^*(G) \subseteq \Lambda^T(G)$.

Let polytope T for the instance $\mathcal{J}3/n = 2, a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ under consideration be given such that inequality (2.2) does not hold. Then the optimality

Table 2.2: *Calculation of the stability radius $\widehat{\varrho}_1(p)$ for problem $\mathcal{J}3/n=2/\mathcal{C}_{max}$*

G_k	$\lvert H_{1k}\rvert$	$\mu\in H_{1k}$ $l^p(\mu)$	$\nu\in H_k$ $l^p(\nu)\geq l_1^p$	$p_{(\beta)}^{\nu\mu}$ $0\leq\beta\leq w_{\nu\mu}$	$\frac{l^p(\nu)-l^p(\mu)-\sum_{\alpha=0}^{\beta}p_{(\alpha)}^{\nu\mu}}{\lvert[\mu]\cup[\nu]\rvert-\lvert[\mu]\cap[\nu]\rvert-\beta}$	$\max\limits_{\beta}$	$\max\limits_{\nu}$	$\min\limits_{\mu}$
1	2	3	4	5	6	7	8	9
G_2	0							
G_3	1	$(O_{2,1},O_{1,2},O_{1,3})$ $l^p(\mu)=150$	$(O_{1,1},O_{1,2},O_{1,3})$ $l^p(\nu)=165=l_1^p$	$p_{(0)}^{\nu\mu}=0$ $p_{(1)}^{\nu\mu}=75$	$\frac{165-150-0}{2-0}=7.5$ $\frac{165-150-75}{2-1}<0$	7.5	30	30
			$(O_{1,1},O_{1,2},O_{2,1},O_{2,2},O_{2,3})$ $l^p(\nu)=270$ >165	$p_{(0)}^{\nu\mu}=0$ $p_{(1)}^{\nu\mu}=30$ $p_{(2)}^{\nu\mu}=55$ $p_{(3)}^{\nu\mu}=75$	$\frac{270-150-0}{4-0}=30$ $\frac{270-150-30}{4-1}=30$ $\frac{270-150-(30+55)}{4-2}=17.5$ $\frac{270-150-(30+55+75)}{4-3}<0$	30		
G_4	0							
G_5	2	$(O_{1,1},O_{1,2},O_{2,3})$ $l^p(\mu)=155$	$(O_{2,1},O_{2,2},O_{1,1},O_{1,2},O_{1,3})$ $l^p(\nu)=280$ >165	$p_{(0)}^{\nu\mu}=0$ $p_{(1)}^{\nu\mu}=40$ $p_{(2)}^{\nu\mu}=50$ $p_{(3)}^{\nu\mu}=60$	$\frac{280-155-0}{4-0}=31.25$ $\frac{280-155-40}{4-1}=28\frac{1}{3}$ $\frac{280-155-(40+55)}{4-2}=15$ $\frac{280-155-(40+55+60)}{4-3}<0$	31.25	31.25	30
			$(O_{2,1},O_{2,2},O_{2,3},O_{1,2},O_{1,3})$ $l^p(\nu)=235$ >165	$p_{(0)}^{\nu\mu}=0$ $p_{(1)}^{\nu\mu}=40$ $p_{(2)}^{\nu\mu}=55$ $p_{(3)}^{\nu\mu}=60$	$\frac{235-155-0}{4-0}=20$ $\frac{235-155-40}{4-1}=13\frac{1}{3}$ $\frac{235-155-(40+55)}{4-2}<0$ $\frac{235-155-(40+55+60)}{4-3}<0$	20		
		$(O_{1,1},O_{2,2},O_{2,3})$ $l^p(\mu)=160$	$(O_{2,1},O_{2,2},O_{1,1},O_{1,2},O_{1,3})$ $l^p(\nu)=280$ >165	$p_{(0)}^{\nu\mu}=0$ $p_{(1)}^{\nu\mu}=40$ $p_{(2)}^{\nu\mu}=50$ $p_{(3)}^{\nu\mu}=60$	$\frac{280-160-0}{4-0}=30$ $\frac{280-160-40}{4-1}=26\frac{2}{3}$ $\frac{280-160-(40+50)}{4-2}=15$ $\frac{280-160-(40+50+60)}{4-3}<0$	30	30	
			$(O_{2,1},O_{2,2},O_{2,3},O_{1,2},O_{1,3})$ $l^p(\nu)=235$ >165	$p_{(0)}^{\nu\mu}=0$ $p_{(1)}^{\nu\mu}=40$ $p_{(2)}^{\nu\mu}=50$ $p_{(3)}^{\nu\mu}=60$	$\frac{235-160-0}{4-0}=18.75$ $\frac{235-160-40}{4-1}=11\frac{2}{3}$ $\frac{235-160-(40+50)}{4-2}=7.5$ $\frac{235-160-(40+50+60)}{4-3}<0$	18.75		

of digraph G_1 is not guaranteed within the given polytope T: There exists another feasible digraph $G_k\in\Lambda(G)$, $k\neq 1$, (we call it a competitive digraph for G_1) with a critical weight being smaller than the critical weight of digraph G_1 in some realization of the processing times $x\in R_+^6$. If such a 'superiority' of the competitive digraph G_k occurs when the processing times are defined by the vector $p^*=(p_{1,1}^*,p_{1,2}^*,\ldots,p_{2,3}^*)\in T$ (i.e., digraph G_k instead of G_1 is optimal for the new vector p^* of the processing times), we can calculate the

stability radius $\widehat{\varrho}_k(p^*)$ of digraph G_k for this new vector p^*.

In the case when the stability radius $\widehat{\varrho}_k(p^*)$ is strictly positive, we can consider the union $O_{30}(p) \cup O_{\widehat{\varrho}_k(p^*)}(p^*)$ of the two balls (instead of one ball $O_{30}(p)$ as before). If inclusion $T \subseteq O_{30}(p) \cup O_{\widehat{\varrho}_k(p^*)}(p^*)$ holds, then a G-solution for the uncertain problem $\mathcal{J}2/n = 3, a_i \le p_i \le b_i/\mathcal{C}_{max}$ is already obtained: $\Lambda^*(G) = \{G_1, G_k\}$. In such a case, a decision-maker needs to use either digraph G_1 or digraph G_k for the realization of an optimal schedule.

Otherwise (if inclusion $T \subseteq O_{30}(p) \cup O_{\widehat{\varrho}_k(p^*)}(p^*)$ does not hold), we have to calculate the stability radius $\widehat{\varrho}_r(p^{**})$ of a competitive digraph G_r of digraph G_k for such a vector p^{**} of the processing times, for which digraph G_r is optimal.

Continuing in this manner, we may cover the given polytope T by the union of the stability balls of some feasible digraphs from set $\Lambda(G)$. As a result, for any vector of the processing times from the polytope T (i.e., whenever inequalities (4) hold), we may obtain at least one optimal schedule.

For Example 2.1 with the vector $p = (75, 50, 40, 60, 55, 30)$, the competitive digraphs for the optimal digraph G_1 are the digraphs $G_3 = (Q^J, A^J \cup E_3^J, \emptyset)$ and $G_5 = (Q^J, A^J \cup E_5^J, \emptyset)$, where $E_3^J = \{(O_{1,1}, O_{2,2}), (O_{1,2}, O_{2,1}), (O_{1,2}, O_{2,3})\}$ and $E_5^J = \{(O_{2,2}, O_{1,1}), (O_{2,1}, O_{1,2}), (O_{2,3}, O_{1,2})\}$. The digraph G_3 along with the completion times of the operations are presented in Figure 2.3.

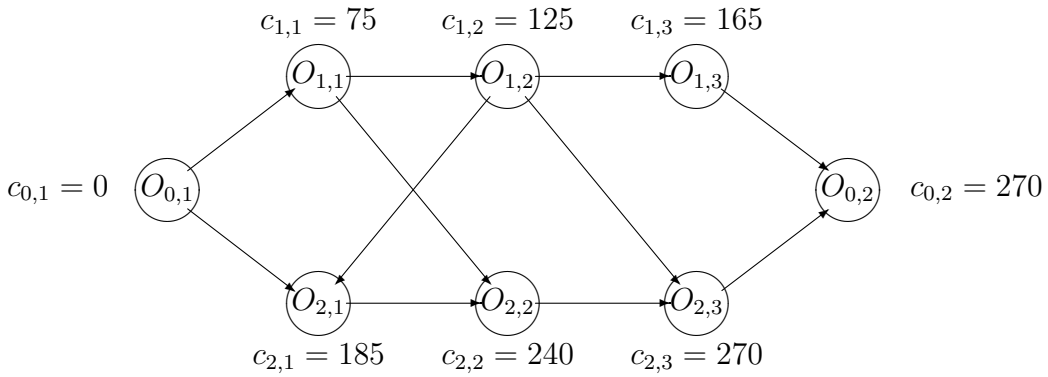


Figure 2.3: *Competitive digraph $G_3 = (Q^J, A^J \cup E_3^J, \emptyset)$ for $G_1 = (Q^J, A^J \cup E_1^J, \emptyset)$ which is optimal for vector $p = (75, 50, 40, 60, 55, 30)$ of the processing times*

As the calculation of the stability radius shows, at the boundary of the ball $O_{30}(p)$ (namely, at the point $p^* = (p^*_{1,1}, p^*_{1,2}, p^*_{1,3}, p^*_{2,1}, p^*_{2,2}, p^*_{2,3}) = (45, 80, 70, 90, 25, 0) \in R^6_+$) both digraphs G_1 and G_3 are optimal. Note that vector p^* has already been determined during our calculation of the stability radius on the basis of formulas (1.31) and (1.32). Specifically, vector p^* has been obtained from vector p by decreasing the processing times

of the operations $O_{1,1}, O_{2,2}$, $O_{2,3}$ *by the value* $\widehat{\varrho}_1(p)$: $p^*_{1,1} = 75 - 30 = 45$, $p^*_{2,2} = 55 - 30 = 25$, $p^*_{2,3} = 30 - 30 = 0$, *and by increasing the processing times of the operations* $O_{1,2}, O_{1,3}, O_{2,1}$ *by the same value* $\widehat{\varrho}_1(p)$: $p^*_{1,2} = 50 + 30 = 80$, $p^*_{1,3} = 40 + 30 = 70$, $p^*_{2,1} = 60 + 30 = 90$. *In other words, to obtain vector* p^*, *we can use the following formula:*

$$p^*_i = \begin{cases} p_i + r, & \text{if } i \in [\mu], \\ \max\{0, p_i - r\}, & \text{if } i \in [\nu] \setminus [\mu], \\ p_i, & \text{if } i \notin [\mu] \cup [\nu], \mu \in H_{sk}, \nu \in H_k, \end{cases} \tag{2.3}$$

where $[\mu] = \{O_{1,2}, O_{1,3}, O_{2,1}\}$, $\mu \in H_{1,3}$, $[\nu] = \{O_{1,1}, O_{1,2}, O_{2,1}, O_{2,2}, O_{2,3}\}$, $\nu \in H_3$, *and* $r = \widehat{\varrho}_1(p) = 30$. *Due to such changes in the processing times, the critical weight of digraph* G_1 *is increased from* 165 *to* 240, *while the critical weight of digraph* G_3 *is decreased from* 270 *to* 240.

The existence of two or more optimal digraphs is a necessary condition (but not a sufficient one) for the stability radius to be equal to zero (see Theorem 1.1 on page 33). Nevertheless, the 'nonstability' of an optimal digraph may happen at the boundary of a stability region (the stability region of the digraph G_s *is the whole set of vectors* $p \in R^q_+$ *with the schedule* s *being optimal), where at least two optimal digraphs exist. Such a situation occurs for Example 2.1 under consideration, namely:* $\widehat{\varrho}_1(p^*) = \widehat{\varrho}_3(p^*) = 0$. *Indeed, according to Theorem 1.1, there exists a path* $\mu^* \in H_1(p)$, $[\mu^*] = \{O_{1,2}, O_{1,3}, O_{2,1}\}$, *such that there does not exist any path* $\nu \in H_3(p)$ *with* $[\mu^*] \subseteq [\nu]$. *On the other hand, there exists a path* $\nu^* \in H_3(p)$, $[\nu^*] = \{O_{1,1}, O_{1,2}, O_{2,1}, O_{2,2}, O_{2,3}\}$, *such that there does not exist any path* $\mu \in H_1(p)$ *with* $[\nu^*] \subseteq [\mu]$. *Note that for point* p^*, *the only competitive digraph for digraph* G_3 *is digraph* G_1 *(and vice versa), where the stability radius of* G_1 *for the original point* $p \in R^q_+$ *has been already calculated.*

Considering the competitive digraph G_5 *instead of the competitive digraph* G_3 *gives also zero stability radii for both digraphs* G_1 *and* G_5 *with the corresponding vector* $p' = (105, 20, 10, 30, 85, 60)$ *of the processing times, constructed due to (1.27) with* $r = \widehat{\varrho}_1(p) = 30$ *for the paths* $[\mu'] = \{O_{1,1}, O_{2,2}, O_{2,3}\}$, $\mu' \in H_{1,5}$, *and* $[\nu'] = \{O_{1,1}, O_{1,2}, O_{1,3}, O_{2,1}, O_{2,2}\}$, $\nu' \in H_5$.

Thus, we can conclude that using the notion of a stability radius given in Definition 1.2 is not sufficient for solving Example 2.1 when inequality (2.2) does not hold for the given set T *of feasible vectors of the processing times.*

From the above discussion it follows that another type of stability radius is required for solving a problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$. While $\widehat{\varrho}_s(p)$ denotes the largest radius of a ball $O_{\widehat{\varrho}}(p)$ within which digraph G_s is 'the best' for the

whole set $\Lambda(G)$ (see Definition 1.2 on page 28), we need to determine the largest ball within which digraph G_s is 'the best' for a subset B of the set $\Lambda(G)$ of feasible digraphs. In particular, for Example 2.1 we need to calculate the largest radius of the ball within which digraph G_3 has the minimal critical weight among the digraphs $\Lambda(G)$ except digraph G_1. Indeed, digraph G_1 is optimal within the ball $O_{\widehat{\varrho}_1(p)}(p)$ and so digraph G_1 has already to be contained in the set of candidates for an optimal realization: $G_1 \in \Lambda^*(G)$. Thus, in Example 2.1, we need to consider the set $B = \Lambda(G)\backslash\{G_1\}$ instead of the whole set $\Lambda(G)$ of feasible digraphs while calculating the stability radius of digraph G_3. It is clear that considering set B instead of set $\Lambda(G)$ may increase such a stability radius in an appropriate way. In Section 2.2, we propose another definition (more general than Definition 1.2) of the stability radius which is just suitable for solving problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$. Note that the given bounds a_i and b_i for the possible variations of the processing time $x_i,\ i \in Q$, may also enlarge the stability ball of the optimal digraph G_s. E.g., this is true for Example 2.1 since inequality (2.2) becomes only a sufficient condition for the optimality of digraph G_1 (but not a necessary one). We provide both necessary and sufficient conditions for a zero (and for an infinitely large) stability radius. In Section 2.3, the formulas (1.31) and (1.32) proven for the case of calculating the stability radius with $0 \le p_i < \infty,\ i \in Q$, are generalized to the case when the variations of the processing times are given by inequalities (4) and some feasible digraphs have to be excluded from the comparisons with 'the best' one.

2.2. Relative Stability Radius

In Chapter 1, the stability radius $\widehat{\varrho}_s(p)$ of an optimal digraph G_s has been investigated which denotes the largest quantity of independent variations of the processing times p_i of operations $i \in Q$ within the interval $[0, \infty)$ such that digraph G_s remains 'the best' (i.e., the weighted digraph $G_s(p)$ has the minimal critical weight) among all feasible digraphs $\Lambda(G)$ (see Definition 1.2 on page 28). From Example 2.1 it follows that for solving problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$, we need a more general notion of the stability radius since the processing time of operation $i \in Q$ falls within the given closed interval $[a_i, b_i],\ 0 \le a_i \le b_i$, and competitive digraphs have to belong to a subset B of set $\Lambda(G)$. The following generalization of the stability radius $\widehat{\varrho}_s(p)$ (we call it *relative* stability radius) is defined by considering the closed interval $[a_i, b_i]$ instead of $[0, \infty)$ and considering set $B \subseteq \Lambda(G)$ instead of the whole set $\Lambda(G)$. In Definition 2.2, l_s^p denotes the critical weight of the weighted

digraph $G_s(p)$, $p \in T$, see equality (1.18) on page 33.

Definition 2.2 *Let digraph $G_s \in B \subseteq \Lambda(G)$ have the minimal critical weight $l_s^{p'}$ for each vector $p' \in O_\varrho(p) \cap T$ among all digraphs from the given set B: $l_s^{p'} = \min\{l_k^{p'} : G_k \in B\}$. The maximal value of the radius ϱ of such a ball $O_\varrho(p)$ is denoted by $\hat{\varrho}_s^B(p \in T)$ and is called the relative stability radius of digraph G_s with respect to polytope T.*

Note that relativity of $\hat{\varrho}_s^B(p \in T)$ is defined not only by the polytope T of feasible vectors, but also by the set B of feasible digraphs. From Definition 1.2 and Definition 2.2, it follows: $\hat{\varrho}_s(p) = \hat{\varrho}_s^{\Lambda(G)}(p \in R_+^q)$. Thus, the relative stability radius is equal to the *maximal error* of the given processing times p_i $(a_i \le p_i \le b_i,\ i \in Q)$ within which the 'superiority' of digraph G_s is still preserved over the given subset B of feasible digraphs.

The following two extreme cases of the relative stability radius are of particular importance for solving problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$. On the one hand, if for any positive real number $\epsilon > 0$ which may be as small as desired, there exist a vector $p' \in O_\epsilon(p) \cap T$ and a digraph $G_k \in B$ such that $l_s^{p'} > l_k^{p'}$, we obtain a zero relative stability radius: $\hat{\varrho}_s^B(p \in T) = 0$. On the other hand, if $l_s^{p'} \le l_k^{p'}$ for any vector $p' \in T$ and for any digraph $G_k \in B$, we obtain an infinitely large relative stability radius: $\hat{\varrho}_s^B(p \in T) = \infty$. Note that even in the case of finite upper bonds $(b_i < \infty,\ i \in Q)$, i.e., when the maximal error of the processing time p_i for each operation $i \in Q$ is restricted by

$$\epsilon_{max} = \max\{\{p_i - a_i,\ b_i - p_i\} : i \in Q\}, \tag{2.4}$$

the value of $\hat{\varrho}_s^B(p \in T)$ may be infinitely large as it follows from Definition 2.2. The deterministic problem $\mathcal{G}//\mathcal{C}_{max}$ with the vector p of processing times and the optimal digraph G_s provides such a trivial example with an infinitely large relative stability radius $\hat{\varrho}_s^B(p \in T)$. Indeed, if $a_i = p_i = b_i$ for each operation $i \in Q$, then polytope T degenerates into a single point: $T = \{p\}$ and therefore, from inclusion $p' \in O_\varrho(p) \cap T$, it follows that vector p' mentioned in Definition 2.2 is definitely equal to vector p, for which digraph G_s is optimal.

To characterize the extreme values of $\hat{\varrho}_s^B(p \in T)$, we define the following binary relation which generalizes the dominance relation used in Chapter 1.

Definition 2.3 *Path ν dominates path μ in set T if and only if for any vector $x = (x_1, x_2, \ldots, x_q) \in T$ the following inequality holds:*

$$l^x(\mu) \le l^x(\nu). \tag{2.5}$$

Note that the binary relation introduced in Definition 2.3 is an extension of the dominance relation introduced in Definition 1.3 (see page 30) in the sense that path ν dominates path μ in any set $T \subseteq R_+^q$ (i.e., due to Definition 2.3) if path ν dominates path μ (due to Definition 1.3). Indeed, if $[\mu] \subset [\nu]$, then inequality $l^x(\mu) \leq l^x(\nu)$ holds for any vector $x \in R_+^q$. Note also that both dominance relations coincide at least when $a_i = 0$ and $b_i = \infty$ for each operation $i \in Q$ (it is easy to see that inclusion $[\mu] \subset [\nu]$ holds if and only if inequality (2.5) holds for $a_i = 0$ and $b_i = \infty,\ i \in Q$). Moreover, in this case equality $l^x(\mu) = l^x(\nu)$ is achieved only if equalities $x_i = a_i = 0$ hold for any operation $i \in [\nu] \setminus [\mu]$.

Thus, we conclude that the dominance relation introduced in Definition 1.3 is a special case of the dominance relation defined by inequality (2.5) when T coincides with the whole vector space R_+^q (i.e., $a_i = 0$ and $b_i = \infty$ for each operation $i \in Q$). Hence, the phrase "path ν dominates path μ" is identical to the phrase "path ν dominates path μ in R_+^q".

The following lemma gives a simple criterion for the dominance relation defined by inequality (2.5) in Definition 2.3.

Lemma 2.1 *Path ν dominates path μ in set T if and only if the following inequality holds:*

$$\sum_{i \in [\mu] \setminus [\nu]} b_i \leq \sum_{j \in [\nu] \setminus [\mu]} a_j. \tag{2.6}$$

Proof. By subtracting all common variables from the left-hand side and the right-hand side of inequality (2.5) and taking into account that $a_i \leq b_i$ for each operation $i \in Q$, we obtain that inequality (2.5) is equivalent to the following ones:

$$\sum_{i \in [\mu] \setminus [\nu]} x_i \leq \sum_{j \in [\nu] \setminus [\mu]} x_j \ \text{ for any } x_i \ \text{ with } \ a_i \leq x_i \leq b_i,\ i \in [\nu] \cup [\mu]. \tag{2.7}$$

Vector $x \in T$ satisfies inequalities (2.7) if and only if inequality (2.6) holds since we have:

$$\sum_{i \in [\mu] \setminus [\nu]} a_i \leq \sum_{i \in [\mu] \setminus [\nu]} b_i \leq \sum_{j \in [\nu] \setminus [\mu]} a_j \leq \sum_{j \in [\nu] \setminus [\mu]} b_j.$$

◇

On the basis of the above path domination, we can introduce the following dominance relation of path sets.

Definition 2.4 *The set of paths H_k dominates the set of paths H_s in T if and only if for any path $\mu \in H_s$, there exists a path $\nu \in H_k$, which dominates path μ in set T.*

The following statement gives a simple sufficient condition when a domination of sets of paths does not hold.

Lemma 2.2 *The set of paths H_k does not dominate the set of paths H_s in T if there exists a path $\mu \in H_s$ such that system*

$$\begin{cases} \sum_{i\in[\nu]\setminus[\mu]} a_i < \sum_{j\in[\mu]\setminus[\nu]} b_j, \\ a_i \le x_i \le b_i,\ i \in Q, \end{cases} \tag{2.8}$$

has a solution for any path $\nu \in H_k$.

Proof. From Definition 2.4, it follows that the set of paths H_k does not dominate the set of paths H_s in T if there exists a path $\mu^* \in H_s$ such that there is no path $\nu \in H_k$ which dominates path μ^* in set T. This means that inequality (2.5) is violated for path $\mu^* \in H_s$ for some vector $x^0 \in T$, i.e., system

$$\begin{cases} l^x(\nu) < l^x(\mu), \\ a_i \le x_i \le b_i,\ i \in Q, \end{cases} \tag{2.9}$$

has a solution for any path $\nu \in H_k$. Furthermore, system (2.9) is consistent if and only if it has the following solution:

$$x_i = x_i^0 = \begin{cases} a_i, & \text{if } i \in [\mu^*] \setminus [\nu], \\ b_i, & \text{if } i \in [\nu] \setminus [\mu^*]. \end{cases} \tag{2.10}$$

It is easy to see that vector x according to (2.10) is a solution of system (2.9) if and only if condition (2.6) does not hold for any vertex $i \in [\nu] \cup [\mu^*]$. In other words, vector $x^0 = (x_1^0, x_2^0, \ldots, x_q^0) \in T$ and path $\mu^* \in T$ provide a solution of the equivalent system (2.8).

◇

Obviously, if $H_k = H_k(p)$, we have $H_k(p') \subseteq H_k = H_k(p)$ for any vector $p' \in R_+^q$ of the processing times. The following lemma shows that the set of critical paths is not expanded for small variations of the processing times.

Lemma 2.3 *If $H_k \ne H_k(p)$, inclusion $H_k(p') \subseteq H_k(p)$ holds for any vector $p' \in O_\epsilon(p) \cap R_+^q$ with the real number $\epsilon_k > \epsilon > 0$ defined as follows:*

$$\epsilon_k = \frac{1}{q}\,[l_k^p - \max\{l^p(\nu) : \nu \in H_k \setminus H_k(p)\}]. \tag{2.11}$$

Proof. Since $H_k \setminus H_k(p) \ne \emptyset$, we can consider any path $\nu^\star \in H_k$ with $l^p(\nu^\star) = \max\{l^p(\nu) : \nu \in H_k \setminus H_k(p)\}$. From (2.11) it follows that $l_k^p - l^p(\nu^\star) = q \cdot \epsilon_k$, and therefore, to make the difference $l_k^p - l^p(\nu^\star)$ equal to zero, we need a vector p' with a distance from vector p greater than or

equal to ϵ_k : $d(p,p') \geq \epsilon_k$. However, due to the condition of Lemma 2.3, we have $d(p,p') \leq \epsilon < \epsilon_k$. Consequently, $\nu^\star \notin H_k(p')$. Since for any path $\nu \in H_k \setminus H_k(p)$ with $l^p(\nu) < l^p(\nu^\star)$ the difference $l^p_k - l^p(\nu)$ is still greater than the product $q \cdot \epsilon_k$, such a path ν cannot belong to set $H_k(p')$.

$\diamond$

Next, we present a generalization of Theorem 1.1 (see page 33) proven for a zero stability radius to the case of a zero relative stability radius.

Theorem 2.1 *Let digraph G_s have the minimal critical weight l^p_s, $p \in T$, within the given subset $B \subseteq \Lambda(G)$ of feasible digraphs. Then equality $\widehat{\varrho}^B_s(p \in T) = 0$ holds if and only if there exists a digraph $G_k \in B$ such that $l^p_s = l^p_k$, $k \neq s$, and the set of paths $H_k(p)$ does not dominate the set of paths $H_s(p)$ in T.*

Proof. *Sufficiency (if).* Let the conditions of Theorem 2.1 hold: There exists a digraph $G_k \in B$ such that $l^p_s = l^p_k$, $k \neq s$, and $H_k(p)$ does not dominate set $H_s(p)$ in T. We show that $\widehat{\varrho}^B_s(p \in T) < \epsilon$ for any given real number $\epsilon > 0$ which may be as small as desired.

Since set $H_k(p)$ does not dominate set $H_s(p)$ in T, there exists a path $\mu^* \in H_s(p)$ such that no path $\nu \in H_k(p)$ dominates path μ^* in set T, i.e., system (2.9) has a solution for any path $\nu \in H_k(p)$. First, we make the following remark.

Remark 2.1 From the consistency of system (2.9), it follows that for the considered problem $\mathcal{G}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$, the trivial case with $a_i = b_i$ for each $i \in Q$ does not hold. Indeed, in this case the first inequality in (2.9) is transformed into inequality $l^p(\nu) < l^p(\mu^*)$ which is certainly wrong: $l^p(\nu) = l^p_k = l^p_s = l^p(\mu^*)$.

We construct a vector $p' = (p'_1, p'_2, \ldots, p'_q)$ with the following components:

$$p'_i = \begin{cases} p_i + \epsilon', & \text{if } i \in [\mu^*],\ p_i \neq b_i, \\ p_i - \epsilon', & \text{if } i \in \{\cup_{\nu \in H_k(p)}[\nu]\} \setminus [\mu^*],\ p_i \neq a_i, \\ p_i, & \text{otherwise,} \end{cases} \tag{2.12}$$

where ϵ' is chosen as a strictly positive real number less than both value ϵ and value $\epsilon_{min} = \max\{0,\ \min\{\min\{p_i - a_i :\ p_i > a_i, i \in Q\},\ \min\{b_i - p_i :\ b_i > p_i, i \in Q\}\}\}$. We can also choose ϵ' less than $\epsilon_k > 0$ defined in (2.11). More precisely, if $H_k \neq H_k(p)$, then $\epsilon_k > 0$, and we can choose ϵ' such that $0 < \epsilon' < \min\{\epsilon,\ \epsilon_k,\ \epsilon_{min}\}$. Otherwise, if $H_k = H_k(p)$, we choose ϵ' such

that $0 < \epsilon' < \min\{\epsilon,\ \epsilon_{min}\}$. Such choices are possible since in both cases, inequality $\epsilon_{min} > 0$ holds due to Remark 2.1.

The following arguments are the same for both cases of the choice of ϵ' except the 'last step' since $H_k \setminus H_k(p) = \emptyset$ in the latter case.

Since system (2.9) has a solution for each path $\nu \in H_k$, the first inequality in (2.9), $l^x(\nu) < l^x(\mu^*)$, has a solution for $x \in T$ which implies that inclusion $[\mu^*] \subset [\nu]$ does not hold for any path $\nu \in H_k(p)$. Therefore, from the equalities $l^p(\nu) = l^p_k = l^p_s = l^p(\mu^*)$ and (2.12), we can conclude that vector p' is a solution to system (2.9) for each path $\nu \in H_k(p)$. In other words, vector p' is a solution to the following system of inequalities:

$$\begin{cases} l^x(\nu) < l^x(\mu^*),\ \nu \in H_k(p), \\ a_i \le x_i \le b_i,\ i \in Q. \end{cases}$$

Thus, we obtain inequality $l^{p'}(\nu) < l^{p'}(\mu^*)$ for each path $\nu \in H_k(p)$, and

$$\max\{l^{p'}(\nu) : \nu \in H_k(p)\} < l^{p'}(\mu^*). \tag{2.13}$$

Since $p' \in O_{\epsilon'}(p) \cap R^q_+$ with $0 < \epsilon' < \epsilon_k$, due to Lemma 2.3, we obtain $H_k(p') \subseteq H_k(p)$ and, as a result, for each path $\tau \in H_k \setminus H_k(p)$ inequality

$$l^{p'}(\tau) < l^{p'}_k = \max\{l^{p'}(\nu) : \nu \in H_k(p)\} \tag{2.14}$$

holds. From inequalities (2.13) and (2.14), it follows that $l^{p'}_k < l^{p'}_s$. Taking into account that $d(p', p) = \epsilon' < \epsilon$, we conclude that $\widehat{\varrho}^B_s(p \in T) < \epsilon$.

Necessity (only if). We prove necessity by contradiction. Suppose that $\widehat{\varrho}^B_s(p \in T) = 0$ but the condition of Theorem 2.1 does not hold. The following two cases *(i)* and *(ii)* of violating the condition of Theorem 2.1 may hold.

(i) There does not exist a digraph $G_k \in B$ such that $l^p_s = l^p_k,\ k \ne s$.

In the trivial case when $B = \{G_s\}$, we have $\widehat{\varrho}^B_s(p \in T) = \infty$ due to Definition 2.2. Let $B \setminus \{G_s\} \ne \emptyset$. Then we calculate the following real number:

$$\epsilon^* = \frac{1}{q} \min\{l^p_t - l^p_s\ :\ G_t \in B,\ t \ne s\} \tag{2.15}$$

which is strictly positive since inequality $l^p_s < l^p_t$ holds for each digraph $G_t \in B,\ t \ne s$. Arguing in a similar way as in the proof of Lemma 2.3, we can show that the difference $l^p_t - l^p_s$ cannot become negative when vector p is replaced by an arbitrary vector $p^0 \in O_{\epsilon^*}(p) \cap R^q_+$. Next, we show that the difference $l^p_t - l^p_s$ cannot become negative when vector p is replaced by an arbitrary vector $p^0 \in O_{\epsilon^*}(p) \cap T \subseteq R^q_+$ with $0 < \epsilon^* < \epsilon_k$.

Since $H_k \backslash H_k(p) \neq \emptyset$, we can consider any path $\nu^\star \in H_k$ with $l^p(\nu^\star) = \max\{l^p(\nu) : \nu \in H_k \backslash H_k(p)\}$. From (2.15) it follows that $l_k^{p^0} - l_s^{p^0} \geq q \cdot \epsilon^*$, and therefore, to make the difference $l_k^{p^0} - l_s^{p^0}$ equal to zero, we need a vector p' with a distance from vector p greater than or equal to ϵ_k : $d(p^0, p') \geq \epsilon_k$. However, due to the conditions of Lemma 2.3, we have $d(p, p') \leq \epsilon^* < \epsilon_k$. Consequently, $\nu^\star \notin H_k(p')$. Since for any digraph $G_t \in B$, with $l^{p^0}(\nu) < l^{p^0}(\nu^\star)$ the difference $l_k^{p^0} - l_s^{p^0}$ is still greater than the product $q \cdot \epsilon^*$, such a path ν cannot belong to the set $H_k(p')$. So we conclude that digraph G_s remains 'the best' (perhaps one of the 'best') within the set B for any vector p^0 of the processing times. Due to Definition 2.2, we have $\hat{\varrho}_s^B(p \in T) \geq \epsilon^* > 0$ which contradicts the above assumption of $\hat{\varrho}_s^B(p \in T) = 0$.

(ii) There exists a digraph $G_k \in B$ such that $l_s^p = l_k^p,\ k \neq s$, and for any such digraph G_k, the set of paths $H_k(p)$ dominates the set of paths $H_s(p)$ in T. In this case, we can take any ϵ that satisfies the following inequalities:

$$0 < \epsilon < \min\left\{\min\{\epsilon_k : l_k^p = l_s^p,\ G_k \in B\},\ \frac{1}{q}\min\{l_t^p - l_s^p : l_t^p > l_s^p,\ G_t \in B\}\right\}.$$

Due to inequality $\epsilon > \epsilon_s$, we get from Lemma 2.3 that equalities

$$l_s^{p^0} = \max_{\mu \in H_s(p^0)} l^{p^0}(\mu) = \max_{\mu \in H_s(p)} l^{p^0}(\mu) \tag{2.16}$$

hold for any vector $p^0 \in O_\epsilon(p) \cap R_+^q$. The statement that for any digraph $G_k \in B, k \neq s$, with $l_s^p = l_k^p$ the set of paths $H_k(p)$ dominates the set of paths $H_s(p)$ in T means that for any path $\mu \in H_s(p)$, there exists a path $\nu^* \in H_k(p)$ such that system

$$\begin{cases} l^x(\nu^*) < l^x(\mu), \\ a_i \leq x_i \leq b_i,\ i \in Q, \end{cases}$$

has no solution. Therefore, inequality

$$l^x(\mu) \leq l^x(\nu^*) \tag{2.17}$$

holds for any vector $x \in T$. Due to inequality (2.17) and taking into account that $\epsilon < \epsilon_k$ and $\epsilon < \epsilon_s$, we obtain the following inequality using Lemma 2.3:

$$\max_{\mu \in H_s(p)} l^{p^0}(\mu) \leq \max_{\nu \in H_k(p)} l^{p^0}(\nu). \tag{2.18}$$

Thus, due to (2.16) and (2.18), we obtain inequality

$$l_s^{p^0} \leq \max_{\nu \in H_k(p)} l^{p^0}(\nu) \tag{2.19}$$

for any digraph $G_k \in B,\ l_s^p = l_k^p,\ k \neq s$. Since $\epsilon < \frac{1}{q}\min\{l_t^p - l_s^p :\ l_t^p > l_s^p,\ G_t \in B\}$, inequality $l_t^p > l_s^p$ implies inequality $l_t^{p^0} > l_s^{p^0}$. Taking into account (2.19), we conclude that $l_s^{p^0} \leq l_k^{p^0}$ for any digraph $G_k \in B$ and for any vector $p^0 \in T$ with $d(p, p^0) \leq \epsilon$. Consequently, $\hat{\varrho}_s^B(p \in T) \geq \epsilon > 0$, which contradicts the assumption of $\hat{\varrho}_s^B(p \in T) = 0$.

$\diamond$

Theorem 2.1 directly implies the following statement.

Corollary 2.1 *If $G_s \in B$ is the unique optimal digraph for the vector $p \in T$ of processing times, then $\hat{\varrho}_s^B(p \in T) > 0$.*

From Theorem 2.1, we obtain the following lower bound for $\hat{\varrho}_s^B(p \in T)$.

Corollary 2.2 *If $G_s \in B$ and $l_s^p = \min\{l_k^p :\ G_k \in B\}$, then $\hat{\varrho}_s^B(p \in T) \geq \epsilon^*$ with ϵ^* calculated according to (2.15).*

PROOF. If there exists a digraph $G_k \in B$ such that $l_s^p = l_k^p,\ k \neq s$, we obtain $\hat{\varrho}_s^B(p \in T) \geq \epsilon^* = 0$ due to Definition 2.2. Otherwise, inequality $\hat{\varrho}_s^B(p \in T) \geq \epsilon^*$ follows from the proof of necessity in Theorem 2.1 (case *(i)*).

$\diamond$

Theorem 2.1 identifies a digraph $G_s \in \Lambda(G)$ whose 'superiority' within the set B is *unstable*: Even a very small change in the processing times can make another digraph from the set B to be 'better' than G_s.

The following theorem identifies a digraph G_s whose 'superiority' within the set B in the polytope T is 'absolute': Any changes of the processing times within the polytope T cannot make another digraph from the set B to be 'better' than digraph G_s.

Theorem 2.2 *For digraph $G_s \in B$, $\hat{\varrho}_s^B(p \in T) = \infty$ if and only if for any digraph $G_t \in B,\ t \neq s$, the set of paths H_t dominates the set of paths $H_s \setminus H$ in T.*

PROOF. *Sufficiency.* Let ϱ be a positive number (as large as desired). We take any vector $p \in O_\varrho(p) \cap T \subseteq R_+^q$ and consider a path $\mu \in H_s$ such that $l_s^p = l^p(\mu)$.

(j) If $\mu \in H$, then $l_s^p = l^p(\mu) \leq l_t^p$ for any digraph $G_t \in \Lambda(G)$.

(jj) If $\mu \in H_s \setminus H$, then due to the condition of Theorem 2.2, it follows that for any digraph $G_t \in B, t \neq s$, there exists a path $\nu^* \in H_t$ such that inequality $l^x(\mu) \leq l^x(\nu^*)$ holds for any vector $x \in T$ (and for vector p as well). Therefore, we obtain $l_s^p = l^p(\mu) < l^p(\nu^*) \leq l_t^p$.

Thus, in both cases *(j)* and *(jj)*, we obtain $l_s^p = \min\{l_t^p : G_t \in B\}$.

Necessity. We prove necessity by contradiction. Let us suppose that $\hat{\varrho}_s^B(p \in T) = \infty$, but there exists a digraph $G_t \in B, t \neq s$, such that the set of paths H_t does not dominate the set of paths $H_s \setminus H$ in T. Thus, there exists a path $\mu^0 \in H_s \setminus H$ such that for any path $\nu \in H_t$, the system of inequalities

$$\begin{cases} l^x(\nu) < l^x(\mu^0), \\ a_i \leq x_i \leq b_i, \ i \in Q, \end{cases} \tag{2.20}$$

has a solution. Therefore, due to Lemma 2.2, inequality

$$\sum_{i \in [\nu] \setminus [\mu^0]} a_i < \sum_{j \in [\mu^0] \setminus [\nu]} b_j \tag{2.21}$$

holds. We consider the vector $p^* = (p_1^*, p_2^*, \ldots, p_q^*) \in T$ with

$$p_i^* = \begin{cases} a_i, & \text{if } i \in \{\cup_{[\nu] \in H_t} [\nu]\} \ \setminus \ [\mu^0], \\ b_i, & \text{if } i \in [\mu^0], \\ p_i & \text{otherwise.} \end{cases}$$

Adding to the left-hand side and to the right-hand side of (2.21) the value $\sum_{j \in [\nu] \cap [\mu^0]} b_j$, we obtain that inequality

$$\sum_{i \in [\nu] \setminus [\mu^0]} a_i + \sum_{j \in [\nu] \cap [\mu^0]} b_j < \sum_{j \in [\mu^0]} b_j$$

holds. Thus, we can conclude that vector p^* is a solution to the system of linear inequalities obtained by joining systems (2.20) for all paths $\nu \in H_t$:

$$\begin{cases} l^{p^*}(\nu) < l^{p^*}(\mu^0), \ \nu \in H_t, \\ a_i \leq x_i \leq b_i, \ i \in Q. \end{cases}$$

Therefore, $l_t^{p^*} < l^{p^*}(\mu^0) \leq l_s^{p^*}$, and hence, we get a contradiction to the above assumption: $\hat{\varrho}_s^B(p \in T) < d(p^*, p) \leq \epsilon_{max} < \infty$.

$\diamond$

The following upper bound for $\hat{\varrho}_s^B(p \in T)$ immediately follows from the proof of necessity in Theorem 2.2.

Corollary 2.3 *If $\hat{\varrho}_s^B(p \in T) < \infty$, then $\hat{\varrho}_s^B(p \in T) \leq \epsilon_{max}$, where value ϵ_{max} is calculated according to (2.4).*

In the following section, we use Theorem 2.2 as a stopping rule in the algorithm developed for solving problem $\mathcal{G}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ since the optimality of digraph $G_s \in B$ with $\hat{\varrho}_s^B(p \in T) = \infty$ does not depend on the vector $p \in T$ of processing times.

2.3. Algorithms for Problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$

From Sections 2.1 and 2.2, it follows that a G-solution for the uncertain problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$ may be obtained on the basis of a repeated calculation of the relative stability radii $\widehat{\varrho}_s^B(p \in T)$. Thus, we need formulas for calculating $\widehat{\varrho}_s^B(p \in T)$. In Section 1.3, formulas (1.31) and (1.32) were proven for calculating the stability radius $\widehat{\varrho}_s(p) = \widehat{\varrho}_s^{\Lambda(G)}(p \in R_+^q)$ (see Theorem 1.4 on page 43). Theorem 2.3, which follows, generalizes these formulas for any given subset $B \subseteq \Lambda(G)$ of feasible digraphs and for any given polytope $T \subseteq R_+^q$ of feasible vectors of the processing times. To present the new formulas, we need the following notations. Let μ and ν be paths in the digraphs from set $\Lambda(G)$. Then $[\mu]+[\nu]$ denotes the *symmetric difference* $[\mu] \cup [\nu] \setminus [\mu] \cap [\nu]$ of the sets $[\mu]$ and $[\nu]$. We calculate the following values:

$$\Delta^i(\mu,\nu) = \begin{cases} b_i - p_i, & \text{if } i \in [\mu] \setminus [\nu], \\ p_i - a_i, & \text{if } i \in [\nu] \setminus [\mu]. \end{cases} \tag{2.22}$$

Moreover, let $\Delta_0^i(\mu,\nu)$ be equal to zero. We order the set of values $\Delta^i(\mu,\nu)$ for all operations i from the symmetric difference $[\mu]+[\nu]$ in the following way:

$$\Delta_1^{i_1}(\mu,\nu) \le \Delta_2^{i_2}(\mu,\nu) \le \ldots \le \Delta_{|[\mu]+[\nu]|}^{i_{|[\mu]+[\nu]|}}(\mu,\nu), \tag{2.23}$$

where the subscript $j \in \{1,2,\ldots,[\mu]+[\nu]\}$ indicates the location of $\Delta^i(\mu,\nu)$ in the above sequence (2.23), and the superscript i_j denotes operation $i_j \in [\mu]+[\nu]$ for which the value $\Delta_j^{i_j}(\mu,\nu)$ was calculated. For simplicity, we shall substitute the superscript i_j by one index i (which means operation $i = i_j$). We hope that it will not cause a misunderstanding.

For any two feasible digraphs G_s and G_k, we introduce the following set of paths: $H_{sk}(T) = \{\mu \in H_s :$ *There is no path* $\nu \in H_k$ *which dominates path* μ *in* $T\}$.

Theorem 2.3 *If digraph G_s has the minimal critical weight l_s^p, $p \in T$, in the given set $B \subseteq \Lambda(G)$ of feasible digraphs, then*

$$\widehat{\varrho}_s^B(p \in T) = \min_{G_k \in B} \widehat{r}_{ks}^B, \tag{2.24}$$

$$\widehat{r}_{ks}^B = \min_{\mu \in H_{sk}(T)} \max_{\nu \in H_k,\ l^p(\nu) \ge l_s^p} \max_{\beta=0,1,\ldots,|[\mu]+[\nu]|-1} \frac{l^p(\nu) - l^p(\mu) - \sum_{\alpha=0}^{\beta} \Delta_\alpha^i(\mu,\nu)}{|[\mu]+[\nu]| - \beta}. \tag{2.25}$$

Proof. From Definition 2.2 it follows:

$$\widehat{\varrho}_s^B(p \in T) = \inf\{d(p,x) :\ x \in T,\ l_s^x > \min\{l_k^x :\ G_k \in B\}\}.$$

Therefore, to find the relative stability radius $\hat{\varrho}_s^B(p \in T)$, it is sufficient to construct a vector $x \in T$ which satisfies the following three conditions.

(1) There exists a digraph $G_k(p) \in B,\ k \neq s$, such that $l_s^x = l_k^x$, i.e.,

$$\max_{\mu \in H_s} l^x(\mu) = \max_{\nu \in H_k} l^x(\nu). \tag{2.26}$$

(2) For any given real number $\epsilon > 0$, which may be as small as desired, there exists a vector $p^\epsilon \in T$ such that $d(x, p^\epsilon) = \epsilon$ and $l_s^{p^\epsilon} > l_k^{p^\epsilon}$, i.e., inequality

$$\max_{\mu \in H_s} l^{p^\epsilon}(\mu) > \max_{\nu \in H_k} l^{p^\epsilon}(\nu) \tag{2.27}$$

is satisfied for at least one digraph $G_k(p) \in B$.

(3) The distance $d(p, x)$ achieves the minimal value among the distances between the vector p and the other vectors in the given polytope T which satisfy both above conditions (1) and (2).

After having constructed such a vector $x \in T$, one can define the relative stability radius of digraph G_s as follows: $\hat{\varrho}_s^B(p \in T) = d(p, x)$. Indeed, the critical path of digraph G_s becomes larger than that of digraph G_k for any vector $p^\epsilon \in T$ with positive real ϵ, which may be as small as desired (see condition (2)). Therefore, digraph G_s has no longer the minimal critical weight among all other feasible digraphs while in the ball $O_{d(p,x)}(p \in T)$, digraph G_s has the minimal critical weight (see condition (3)). Digraph G_k satisfying conditions (1) and (2) is called a *competitive* digraph for digraph G_s. To satisfy conditions (1), (2) and (3) (except the inclusion $x \in T$), we first search for a vector $x = p(r) = (p_1(r), p_2(r), \ldots, p_q(r)) \in R^q$ with the components $p_i(r) \in \{p_i, p_i + r, p_i - r\}$ on the basis of a direct comparison of the paths from set H_s and the paths from the sets H_k, where $G_k \in B$.

Let the value $l^p(\nu)$ be greater than the weight of a critical path in the optimal digraph G_s. To satisfy equality (2.26), the weight of a path $\nu \in H_k$ must be smaller than or equal to the weight of at least one path $\mu \in H_s$, and there must exist a path $\nu \in H_k$ with a weight equal to the weight of a critical path of G_s. Thus, if we have calculated

$$r_\nu = \min_{\mu \in H_s} \frac{l^p(\nu) - l^p(\mu)}{|[\mu] + [\nu]|}, \tag{2.28}$$

we obtain equality

$$\max_{\mu \in H_s} l^{p(r)}(\mu) = l^{p(r)}(\nu) \tag{2.29}$$

for vector $p(r) = p(r_\nu)$ with the components

$$p_i(r) = p_i(r_\nu) = \begin{cases} p_i + r_\nu, & \text{if } i \in [\mu], \\ p_i - r_\nu, & \text{if } i \in [\nu] \backslash [\mu], \\ p_i, & \text{if } i \notin [\mu] + [\nu]. \end{cases} \tag{2.30}$$

We can give the following remark.

Remark 2.2 Due to (2.28), the vector $p(r)$ calculated in (2.30) is the closest one to the given vector p among all vectors x for which equality (2.29) holds with $p(r) = x$. Indeed, to make the difference $l^x(\nu) - \max_{\mu \in H_s} l^x(\mu)$ equal to zero, one needs a q-dimensional vector x with a distance from vector p greater than or equal to r_ν : $d(p, x) \ge r_\nu$.

To reach equality (2.26) for digraph G_k, we have to repeat calculation (2.28) for each path $\nu \in H_k$ with $l^p(\nu) \ge l^p_s$. Instead of vector $p(r_\nu)$, we consider vector $p(r) = p(r_{G_k})$ calculated according to formula (2.30), where

$$r_{G_k} = \min_{\mu \in H_s} \max_{\nu \in H_k;\ l^p(\nu) \ge l^p_s} \frac{l^p(\nu) - l^p(\mu)}{|[\mu] + [\nu]|}. \tag{2.31}$$

Next, we consider inequality (2.27). Since the vectors of the processing times have to belong to polytope $T \subseteq R^q_+$, this inequality may not be valid for a vector $p^\epsilon \in T$ if path ν dominates path μ in set T. Thus, we can restrict our consideration to the subset $H_{sk}(T)$ of the set H_s of all paths, which are not dominated in T by paths from set H_k and for which there does not exist a path $\nu \in H_k$ such that $[\nu] = [\mu]$. Hence, we can replace H_s in equality (2.31) by $H_{sk}(T)$. To obtain the desired vector $x \in R^q$, we have to use equality (2.31) for each digraph $G_k \in \Lambda(G), k \ne s$. Let r denote the minimum of such a value r_{G_k}: $r = r_{G_{k^*}} = \min\{r_{G_k} : G_k,\ k \ne s\}$ and let $\nu^* \in H_{k^*}$ and $\mu^* \in H_{sk^*}$ be paths at which value $r_{G_{k^*}}$ has been reached:

$$r_{G_{k^*}} = r_{\nu^*} = \frac{l^p(\nu^*) - l^p(\mu^*)}{|[\mu] + [\nu]|}.$$

Taking into account (2.31), we note that, if $r_{\nu^*} \le \Delta = \min\{b_i - p_i :\ i \in [\nu^*] \setminus [\mu^*]\}$ and $r_{\nu^*} \le \Delta' = \min\{p_i - a_i :\ i \in [\nu^*] \setminus [\mu^*]\}$, then vector $p(r)$ belongs to polytope T. Due to Remark 2.2 given after formula (2.30), we have obtained the following lower bound for the stability radius:

$$\hat{\varrho}^B_s(p \in T) \ge r \ = \ \min_{G_k \in B} \ \min_{\mu \in H_{sk}(T)} \ \max_{\nu \in H_k;\ l^p(\nu) \ge l^p_s} \ \frac{l^p(\nu) - l^p(\mu)}{|[\mu] + [\nu]|}. \tag{2.32}$$

The bound (2.32) is tight: If $\hat{\varrho}^B_s(p \in T) \le \min\{\Delta^i(\mu^*, \nu^*) \ :\ i \in [\mu^*] \cup [\nu^*]\}$, then $\hat{\varrho}^B_s(p \in T) = r$. We have $\hat{\varrho}^B_s(p \in T) = r$ in (2.32) if $\hat{\varrho}^B_s(p \in T) \le \epsilon_{min}$. To obtain the exact value of the relative stability radius $\hat{\varrho}^B_s(p \in T)$ in the general case, we can use vector $x = p^*(r) = (p^*_1(r), p^*_2(r), ..., p^*_q(r))$ with the components

$$p^*_i(r) = \begin{cases} p_i + \min\{r, b_i - p_i\}, & \text{if } i \in [\mu], \\ p_i - \min\{r, p_i - a_i\}, & \text{if } i \in [\nu] \setminus [\mu], \\ p_i, & \text{if } i \notin [\mu] + [\nu], \end{cases} \tag{2.33}$$

instead of the vector $p(r)$ defined in (2.30). As it follows from Remark 2.2, such a vector $p^*(r) \in T$ is the closest one to the vector p among all vectors $x \in T$ which satisfy both conditions (1) and (2).

For calculating the maximal value r for the vector $p^*(r)$, we can consider each operation i from the set $[\mu] \cup [\nu]$ one by one in a non-decreasing order (2.23) of the values $\Delta^i(\mu, \nu)$ defined in (2.22). As a result, formula (2.32) will be transformed into the formulas given in Theorem 2.3.

$\diamond$

We can give the following remark.

Remark 2.3 The formulas (2.24) and (2.25) defined in Theorem 2.3 turn into $\widehat{\varrho}_s^B(p \in T) = \infty$ if $H_{sk}(T) = \emptyset$ for each digraph $G_k \in B$.

Example 2.1 (continued). *Returning to Example 2.1 given in Section 2.1, let us consider problem $\mathcal{J}3/n = 2, a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ whose input data are given by the weighted mixed graph $G(p)$ in Figure 2.1 together with vector $a = (a_{1,1}, a_{1,2}, \ldots, a_{2,3})$ and vector $b = (b_{1,1}, b_{1,2}, \ldots, b_{2,3})$ of lower and upper bounds for the possible variations of the processing times p, where $a = (35, 40, 20, 50, 45, 20)$ and $b = (100, 90, 110, 80, 80, 40)$. The numerical input data for this instance of problem $\mathcal{J}3/n = 2, a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ are given in Table 2.3. Since the mixed graph G is the same for the above de-*

Table 2.3: *Numerical data for problem $\mathcal{J}3/n=2, a_i \leq p_i \leq b_i/\mathcal{C}_{max}$*

i	1	1	1	2	2	2
j	1	2	3	1	2	3
a_{ij}	35	40	20	50	45	20
b_{ij}	100	90	110	80	80	40

terministic problem $\mathcal{J}3/n = 2/\mathcal{C}_{max}$ considered in Section 2.1 and for the new uncertain problem $\mathcal{J}3/n = 2, a_i \leq p_i \leq b_i/\mathcal{C}_{max}$, we have the same set $\Lambda(G)$ of feasible digraphs (see Table 2.1 on page 86). Moreover, if we start the calculation with the same initial vector $p = (75, 50, 40, 60, 55, 30)$ of the processing times, we obtain the same optimal digraph G_1, presented in Figure 2.2 with the dummy operations $O_{0,1}$ and $O_{0,2}$. Using Theorem 2.3, we can calculate the relative stability radius of this digraph: $\widehat{\varrho}_1^{\Lambda(G)}(p \in T) = 60$, where polytope $T \in R_+^6$ is defined by the above vectors a and b (see Table 2.3). Note that, due to these bounds a_{ij} and b_{ij} for the possible variations of the processing times p_{ij}, $O_{ij} \in Q^J = \{O_{1,1}, O_{1,2}, \ldots, O_{2,3}\}$, the

stability radius of the digraph G_1 increased from 30 to 60 (remind that in Section 2.1, we calculated $\widehat{\varrho}_1^{\Lambda(G)}(p \in R_+^6) = \varrho_1(p) = 30$). One can observe the calculation of $\widehat{\varrho}_1^{\Lambda(G)}(p \in T)$ in Table 2.4. The set $H_{1k}(T)$ is empty for each digraph $G_k, k \in \{2,4,5\}$. Note that $H_{sk}(T) \subseteq H_{sk}$, therefore we have $H_{1,2}(T) = \emptyset$ and $H_{1,4}(T) = \emptyset$. Moreover, for both paths $\mu_1 = (O_{1,1}, O_{1,2}, O_{2,3}) \in H_{1,5}$ and $\mu_2 = (O_{1,1}, O_{2,2}, O_{2,3}) \in H_{1,5}$, there exists a path $\nu_2 = (O_{2,1}, O_{2,2}, O_{1,1}, O_{1,2}, O_{1,3}) \in H_5$ which dominates both paths μ_1 and μ_2 in T, i.e., inequality (2.6) holds. Table 2.4 has an analogous design as Table 2.2 from Section 2.1 with the exception of column 5, which contains the values $\Delta_\beta^{ij}(\mu,\nu)$, $\beta \in \{0,1,\ldots,|[\mu]+[\nu]|-1\}$, defined by formula (2.22) on page 103 in non-decreasing order defined by sequence (2.23). Let us consider path $\mu = (O_{2,1}, O_{1,2}, O_{1,3}) \in H_{1,3}(T)$ and path $\nu_1 = (O_{1,1}, O_{1,2}, O_{1,3}) \in H_3$. For each vertex from set $[\mu]+[\nu_1]$ (set $[\mu]+[\nu_1]$ denotes the symmetric difference of sets $[\mu]$ and $[\nu_1]$), $|[\mu]+[\nu_1]| = 2$, we calculate the values $\Delta^{1,1}(\mu,\nu_1) = p_{1,1} - a_{1,1} = 75 - 35 = 40$, $\Delta^{2,1}(\mu,\nu_1) = b_{2,1} - p_{2,1} = 80 - 60 = 20$. By a comparison of path μ with path $\nu_2 = (O_{1,1}, O_{1,2}, O_{2,1}, O_{2,2}, O_{2,3}) \in H_3$, we find the values $\Delta^{1,1}(\mu,\nu_2) = p_{1,1} - a_{1,1} = 75 - 35 = 40$, $\Delta^{1,3}(\mu,\nu_2) = b_{1,3} - p_{1,3} = 110 - 40 = 70$, $\Delta^{2,2}(\mu,\nu_2) = p_{2,2} - a_{2,2} = 55 - 45 = 10$, $\Delta^{2,3}(\mu,\nu_2) = p_{2,3} - a_{2,3} = 30 - 20 = 10$. The sequential calculations of the fraction from the formula (2.25) are represented in column 6 in Table 2.4. Column 9 (see Table 2.2) is redundant for this small example. So, one of the two competitive digraphs, namely digraph G_3 (see Figure 2.3 on page 92), remains also a competitive digraph of G_1 for problem $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\mathcal{C}_{max}$. However, the new vector of the processing times $p^* = p^{(2)}$ has to be calculated due to formula (2.33) with $r = \widehat{\varrho}_1^{\Lambda(G)}(p \in T) = 60$, $\mu = (O_{2,1}, O_{1,2}, O_{1,3}) \in H_1$ and $\nu = (O_{1,1}, O_{1,2}, O_{2,1}, O_{2,2}, O_{2,3}) \in H_{1,3}(T) \subseteq H_3$. Thus, vector p^* is as follows: $p^* = p^{(2)} = (35, 90, 100, 80, 45, 20)$. Next, we follow the scheme proposed on pages 90–94 for obtaining a G-solution of problem $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\mathcal{C}_{max}$. We calculate $\widehat{\varrho}_3^{\Lambda(G)\setminus\{G_1\}}(p^{(2)} \in T) = 32.5$ on the basis of Theorem 2.3 and obtain the competitive digraph G_2 of digraph G_3. For digraph G_2, the minimum in (2.24) is reached on the set $B = \Lambda(G) \setminus \{G_1\}$, and thus digraph G_2 becomes optimal at least for one point $p^{(3)}$ of the stability sphere (the boundary of the stability ball $O_{32.5}(p^{(2)})$). Then we calculate the stability radius $\widehat{\varrho}_2^{\Lambda(G)\setminus\{G_1,G_3\}}(p^{(3)} \in T) = 27.5$ for the new optimal digraph G_2 and for the new set $B := B \setminus \{G_3\} = \Lambda(G) \setminus \{G_1, G_3\}$. Then on the basis of Theorem 2.2 (or Theorem 2.3), we obtain $\widehat{\varrho}_4^{\Lambda(G)\setminus\{G_1,G_2,G_3\}}(p^{(4)} \in T) = \infty$.

Thus, solving problem $\mathcal{J}3/n = 2, a_i \le p_i \le b_i/\mathcal{C}_{max}$ takes four iterations presented in Table 2.5. We obtain that the set of digraphs $\{G_1, G_2, G_3, G_4\}$

Table 2.4: *Calculation of the relative stability radius* $\widehat{\varrho}_1^{\Lambda(G)}(p \in T)$ *for problem* $\mathcal{J}3/n = 2, a_i \le p_i \le b_i/\mathcal{C}_{max}$

G_k	$\lvert H_{1k}(T)\rvert$	$\mu \in H_{1k}(T)$ $l^p(\mu)$	$\nu \in H_k$ $l^p(\nu) \ge l_1^p$	$\Delta_\beta^{ij}(\mu,\nu)$ $0 \le \beta \le \lvert[\mu]+[\nu]\rvert - 1$	$\frac{l^p(\nu) - l^p(\mu) - \sum_{\alpha=0}^{\beta} \Delta_\alpha^{ij}(\mu,\nu)}{\lvert[\mu]+[\nu]\rvert - \beta}$	$\max\limits_{\beta}$	$\max\limits_{\nu}$
1	2	3	4	5	6	7	8
G_2	0						
G_3	1	$(O_{2,1}, O_{1,2}, O_{1,3})$	$(O_{1,1}, O_{1,2}, O_{1,3})$	$\Delta_0^{ij}(\mu,\nu_1) = 0$ $\Delta_1^{2,1}(\mu,\nu_1) = 20$	$\frac{165-150-0}{2-0} = 7.5$ $\frac{165-150-20}{2-1} < 0$	7.5	60
		$l^p(\mu) = 150$	$(O_{1,1}, O_{1,2}, O_{2,1}, O_{2,2}, O_{2,3})$ $l^p(\nu_2) = 270 > 165$	$\Delta_0^{ij}(\mu,\nu_2) = 0$ $\Delta_1^{2,2}(\mu,\nu_2) = 10$ $\Delta_2^{2,3}(\mu,\nu_2) = 10$ $\Delta_3^{1,1}(\mu,\nu_2) = 40$	$\frac{270-150-0}{4-0} = 30$ $\frac{270-150-10}{4-1} = 36\frac{2}{3}$ $\frac{270-150-(10+10)}{4-2} = 50$ $\frac{270-150-(10+10+40)}{4-3} = 60$	60	
G_4	0						
G_5	0						

is a G-solution to problem $\mathcal{J}3/n = 2, a_i \le p_i \le b_i/\mathcal{C}_{max}$ *under consideration:* $\Lambda^*(G) = \{G_1, G_2, G_3, G_4\}$. *In other words, the given polytope* T *is covered by the union of the stability balls of four digraphs from set* $\Lambda^*(G)$.

Table 2.5: *Constructing a G-solution of problem* $\mathcal{J}3/n = 2, a_i \le p_i \le b_i/\mathcal{C}_{max}$ *by Algorithm* $SOL\_\mathcal{C}_{max}(1)$

i	Center $p^{(i)} \in T$ of the stability ball	Set B of feasible digraphs	Optimal digraph G_s	$\widehat{\varrho}_s^B(p^{(i)} \in T)$	Competitive digraph of G_s
1	$(75, 50, 40, 60, 55, 30)$	$\Lambda(G)$	G_1	60	G_3
2	$(35, 90, 100, 80, 45, 20)$	$\Lambda(G) \setminus \{G_1\}$	G_3	32.5	G_2
3	$(67.5,\ 90,\ 67.5,\ 80,\ 77.5,\ 40)$	$\Lambda(G) \setminus \{G_1, G_3\}$	G_2	27.5	G_4
4	$(40, 90, 95, 80, 80, 40)$	$\Lambda(G) \setminus \{G_1, G_2, G_3\}$	G_4	∞	–

The projections of these stability balls on the plane for the component $p_{1,3}$ *of vector* p *given at the axis of the x-coordinates and for the component* $p_{2,2}$ *of vector* p *given at the axis of the y-coordinates are drawn in Figure 2.4. The last stability ball has an infinite radius* $\widehat{\varrho}_4(p^{(4)} \in T) = \infty$, *i.e., it coincides with the whole vector space* R^6: $O_\infty(p^{(4)}) = R^6$. *Thus, the stability ball* $O_\infty(p^{(4)})$ *covers the given polytope* T *and all other stability balls. (The stability ball* $O_\infty(p^{(4)})$ *is not shown in Figure 2.4.) The compet-*

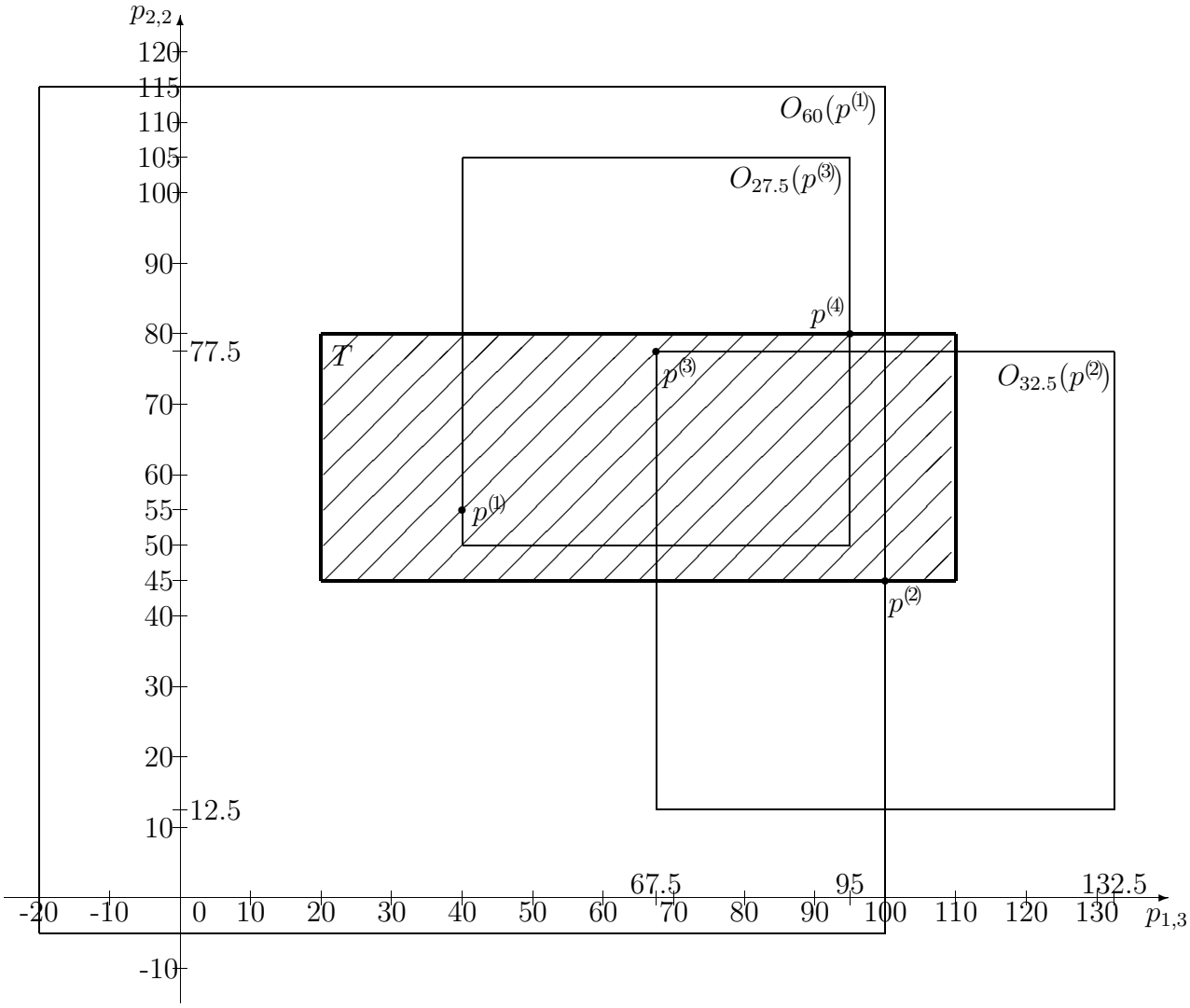


Figure 2.4: *Projections of the stability balls on the plane for problem* $\mathcal{J}3/n=2, a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ *constructed by Algorithm* $SOL\_\mathcal{C}_{max}(1)$

itive digraph, for which the minimum in (1.31) is reached, becomes optimal at least for one point of the stability sphere (the boundary of the stability ball $O_{\widehat{\varrho}_s(p)}(p)$*) and it becomes 'better' than digraph* $G_s(p)$*. For some suitable changes of the processing times* $p_i \pm (\widehat{\varrho}_s(p) + \epsilon)$ *(where* ϵ *is a positive real number, and it may be as small as desired), at least one of the four digraphs* $\{G_1, G_2, G_3, G_4\}$ *becomes optimal. Therefore, a decision-maker can use one of the schedules from set* $\Lambda^*(G) = \{G_1, G_2, G_3, G_4\}$ *for the possible realization of the processing times. As it will be shown at the end of this section, set* $\{G_1, G_2, G_3, G_4\}$ *is not a minimal G-solution of the considered instance of problem* $\mathcal{J}3/n=2, a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ *since at least digraph* G_2 *is redundant in this set.*

Problem $\mathcal{G}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ may be solved as follows. Let B denote a

subset of feasible digraphs which contains a G-solution $\Lambda^*(G)$ for problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$. On the basis of the algorithm, which follows, we can expand the set $\Lambda' \subseteq \Lambda^*(G)$ starting with $\Lambda' = \emptyset$ and finishing with $\Lambda' = \Lambda^*(G)$.

Algorithm $SOL\_\mathcal{C}_{max}(1)$

Input: Set $\Lambda(G)$ of feasible digraphs,
polytope T of feasible processing times.
Output: G-solution $\Lambda^*(G)$ of problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$.

Step 1: Find a set $B \subseteq \Lambda(G)$ such that $\Lambda^*(G) \subseteq B$.
Step 2: Set $\Lambda' = \emptyset$.
Step 3: Fix the vector p of processing times, $p \in T$.
Step 4: Find an optimal digraph $G_s(p) \in B$ for problem $\mathcal{G}//\mathcal{C}_{max}$ with the vector p of processing times.
Step 5: Calculate the relative stability radius $\hat{\varrho}_s^B(p \in T)$.
Step 6: **IF** $\hat{\varrho}_s^B(p \in T) < \infty$ and $B \setminus \{G_s\} \neq \emptyset$ **THEN**
BEGIN
Step 7: Select a digraph $G_k(p) \in B$ which is a competitive digraph for $G_s(p)$.
Step 8: Find a vector $p^* \in T$ of processing times closest to p such that $l_s^{p^*} = l_k^{p^*}$ and for any small $\epsilon > 0$, there exists a vector p^ϵ with $l_s^{p^\epsilon} > l_k^\epsilon$ and $d(p^*, p^\epsilon) \le \epsilon$.
Step 9: Set $\Lambda' := \Lambda' \cup \{G_s\}$.
Step 10: Set $B := B \setminus \{G_s\}$.
Step 11: Set $s = k$; $p = p^*$ **GOTO** *Step 5*
END
Step 12: **ELSE** $\Lambda^*(G) = \Lambda' \cup \{G_s\}$ **STOP**

Next, we concretize some steps of Algorithm $SOL\_\mathcal{C}_{max}(1)$. In Step 1, the determination of the set $B = \Lambda(G)$ of all feasible digraphs by an explicit enumeration is possible only for a small number of edges in the mixed graph G. In the computational experiments discussed in Section 1.5, such a direct enumeration of feasible digraphs $\Lambda(G)$ has been used for a small number $|E|$ of edges of the mixed graph $G = (Q, A, E)$, namely for $|E| \le 30$. These experiments have shown that a competitive digraph has a critical weight, which is close to that of an optimal digraph. Moreover, using the simple bound proven at the end of this section, one can considerably restrict the number of feasible digraphs, with which a comparison of an optimal digraph

G_s has to be done while calculating the relative stability radius $\hat{\varrho}_s^B(p \in T)$. For a large cardinality of the set E, one can use a branch-and-bound algorithm for the construction of the k best digraphs (see Section 1.5). As it was shown for the traveling salesman problem [228, 230] and for linear Boolean programming [365], the running time of such a branch-and-bound algorithm grows rather slowly with k.

In Step 3, we have to fix the processing times as any vector from set T. For example, we can use a 'historical' vector p of the processing times which helps to simplify Steps 3, 4 or 5 (as it was done in Example 2.1). If the input data of the problem are new, we can assume $p_i = \frac{1}{2}(b_i - a_i),\ i \in Q$.

Step 4 may be realized by an explicit enumeration or by an implicit enumeration (e.g., by a branch-and-bound method) of the feasible digraphs B. In Step 4, we can apply Theorem 2.1 to guarantee that the selected optimal digraph G_s is stable. If $\hat{\varrho}_s^B(p \in T) = 0$, we can take another optimal digraph (the latter exists due to Theorem 2.1) which is stable, or we can change the initial vector p of the processing times.

Steps 5, 7, and 8 may be done on the basis of Theorem 2.2 or Theorem 2.3. If $\hat{\varrho}_s^B(p \in T) = \infty$, Theorem 2.2 can be used as a 'stopping rule' of the algorithm. Otherwise, we are forced to use Theorem 2.3 which is more time-consuming. A competitive digraph and a new vector p^* of the processing times are calculated in Algorithm $SOL\_\mathcal{C}_{max}(1)$ in parallel with the calculation of the relative stability radius $\hat{\varrho}_s^B(p \in T)$. Note that a competitive digraph is not necessarily uniquely determined, and one can take any of them. Steps 5 and 7 are rather complicated. In Algorithm $SOL\_\mathcal{C}_{max}(1)$ we must anew construct a set $H_{sk}(T)$ in each iteration based on a direct comparison of the paths in a new optimal digraph G_s and in each other digraph G_k from set B, so it may be very time-consuming.

Next, we propose Algorithm $SOL\_\mathcal{C}_{max}(2)$, which is often more efficient. This algorithm focuses on one of the optimal digraphs G_1 and on one vector p of the processing times in set T. Let $\{\Gamma_i : i = 1, 2, \ldots, I\}$ be the set of competitive digraphs of digraph G_1 with respect to set B, where i is a counter of the current iteration and I is the number of iterations.

Algorithm $SOL\_\mathcal{C}_{max}(2)$

Input: Set $\Lambda(G)$ of feasible digraphs,
polytope T of feasible processing times.
Output: G-solution $\Lambda^*(G)$ of problem $\mathcal{G}/a_i \le p_i \le b_i/\mathcal{C}_{max}$.

Step 1: Find a set $B \subseteq \Lambda(G)$ such that $\Lambda^*(G) \subseteq B$.

Step 2: Set $\Lambda' = \emptyset$; $i = 1$ and $\Gamma_i = \emptyset$.
Step 3: Fix the vector p of processing times, $p \in T$.
Step 4: Find an optimal digraph $G_1(p) := G_s(p) \in B$ for problem $\mathcal{G}//\mathcal{C}_{max}$ with the vector p of processing times.
Step 5: Calculate $\hat{\varrho}_1^B(p \in T)$.
Step 6: **IF** $\hat{\varrho}_1^B(p \in T) < \infty$ **THEN**
BEGIN
Step 7: Select the set of competitive digraphs Γ_i of digraph $G_1(p)$ with respect to set B.
Step 8: Set $\Lambda' := \Lambda' \cup \Gamma_i$.
Step 9: Set $B := B \setminus \Gamma_i$ and $i := i + 1$. **GOTO** *Step 5*
END
Step 10: **ELSE** $\Lambda^*(G) := \Lambda' \cup \{G_1\}$ **STOP**

Using Algorithm $SOL\_\mathcal{C}_{max}(2)$, we construct an increasing sequence of the relative stability radii $\hat{\varrho}_1 < \hat{\varrho}_2 < \ldots < \hat{\varrho}_I$ of the stability balls $O_{\hat{\varrho}_i}(p)$, $i \in \{1, 2, \ldots, I\}$, with the same center $p \in T$ and different sets of feasible digraphs $B = \Lambda(G) \setminus \cup_{j=1}^{i} \Gamma_j$. Moreover, we construct a sequence of 'nested sets' of the competitive digraphs Γ_1, $\Gamma_1 \cup \Gamma_2$, $\ldots$, $\cup_{i=1}^{I} \Gamma_i$ of digraph G_1, where the set $\{G_1\} \cup \{\cup_{i=1}^{I} \Gamma_i\}$ is a G-solution $\Lambda^*(G)$ to the scheduling problem on the mixed graph $(Q, A \cup E_1, \emptyset)$, and G_1 is one of the optimal digraphs in the set $\Lambda(G)$ for the vector $p \in T$ of processing times.

The most difficult part of Algorithm $SOL\_\mathcal{C}_{max}(2)$ is to find the stability radius $\hat{\varrho}_1^B(p \in T)$ (Step 5 and Step 6) and to find the sets of competitive digraphs (Step 7). However, one can use the following remark.

Remark 2.4 It is not necessary to perform Steps 1 - 11 since one can construct a G-solution $\Lambda^*(G)$ in one scan. Namely, from Remark 2.3, it follows that all digraphs $G_k, k \neq 1$, for which a set $H_{1k}(T) \neq \emptyset$ was constructed in Step 5 are united with the optimal digraph G_1 and compose a G-solution: $\Lambda^*(G) = \{G_1\} \cup \{\cup_{i=1}^{I} \Gamma_i\} = \{G_1\} \cup \{G_k : \ H_{1k}(T) \neq \emptyset\}$.

Thus, one can use the software developed for the problems discussed in Chapter 1 with the following modification: We add the loop of Steps 6 – 9. An increasing sequence of the relative stability radii of the stability balls with the same center $p \in T$ corresponds to an increasing sequence of the values $\hat{r}_{k1}^B$ calculated by (2.25) for the optimal digraph $G_1(p)$ in Step 5. A competitive digraph (or set of competitive digraphs Γ_i) of digraph $G_1(p)$ is constructed in one scan as well.

Example 2.1 (continued). *Solving the above problem takes only two itera-*

tions by Algorithm SOL_$\mathcal{C}_{max}(2)$ (see Table 2.6). Thus, the set of digraphs $\Lambda^*(G) = \{G_1, G_3\}$ is also a G-solution to problem $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\mathcal{C}_{max}$ under consideration. Using Algorithm SOL_$\mathcal{C}_{max}(2)$, one can construct two stability balls $O_{60}(p)$ and $O_\infty(p)$, which cover the polytope T (see Figure 2.5). (Again, the stability ball with an infinite radius cannot be shown.) So, we convince that Algorithm SOL_$\mathcal{C}_{max}(1)$ did not construct a minimal G-solution. Indeed, in the G-solution $\{G_1, G_2, G_3, G_4\}$ constructed by Algorithm SOL_$\mathcal{C}_{max}(1)$, both digraphs G_2 and G_4 are redundant. In the general case, we do not know whether Algorithm SOL_$\mathcal{C}_{max}(2)$ constructs a minimal G-solution as well. However, for Example 2.1, it is easy to see that there is no one-element G-solution for this instance. Hence, the G-solution $\Lambda^*(G) = \{G_1, G_3\}$ presented in Table 2.6 is minimal for Example 2.1 (with respect to the cardinality of set $\Lambda^*(G)$).

Remark 2.5 For both algorithms, fixing the initial vector p in Step 3 and the choice of an optimal digraph $G_s(p)$ in Step 4 (and also in Step 7 for Algorithm *SOL_$\mathcal{C}_{max}(1)$*) have a large influence on the further calculations and the resulting G-solution.

Next, we show how to restrict the number of digraphs G_k (the cardinality of set B) with which an optimal digraph G_s has to be compared in the process of the calculation of the relative stability radius $\widehat{\varrho}_s^B(p \in T)$.

Redundant Digraphs for Calculating $\widehat{\varrho}_s^B(p \in T)$

Due to formulas (2.24) on page 103, the calculation of the relative stability radius is reduced to a complicated calculation on the set of digraphs $B \subseteq \Lambda(G)$. The main objects for the calculation of $\widehat{\varrho}_s^B(p \in T)$ are the sets of paths in the digraphs $G_k \in B$. In the worst case, the calculation of $\widehat{\varrho}_s^B(p \in T)$ implies to have an optimal digraph G_s and to construct all digraphs from the subset B of the set $\{G_1, G_2, \ldots, G_\lambda\}$. In order to restrict the number of digraphs G_k with which a comparison of the optimal digraph G_s has to be

Table 2.6: *Constructing a G-solution to problem $\mathcal{J}3/n = 2, a_i \le p_i \le b_i/\mathcal{C}_{max}$ by Algorithm SOL_$\mathcal{C}_{max}(2)$*

i	Set B	$\widehat{\varrho}_1^B(p \in T)$	Set Γ_i of competitive digraphs of the optimal digraph G_1
1	$\Lambda(G)$	60	$\{G_3\}$
2	$\Lambda(G) \setminus \{G_3\}$	∞	$\emptyset$

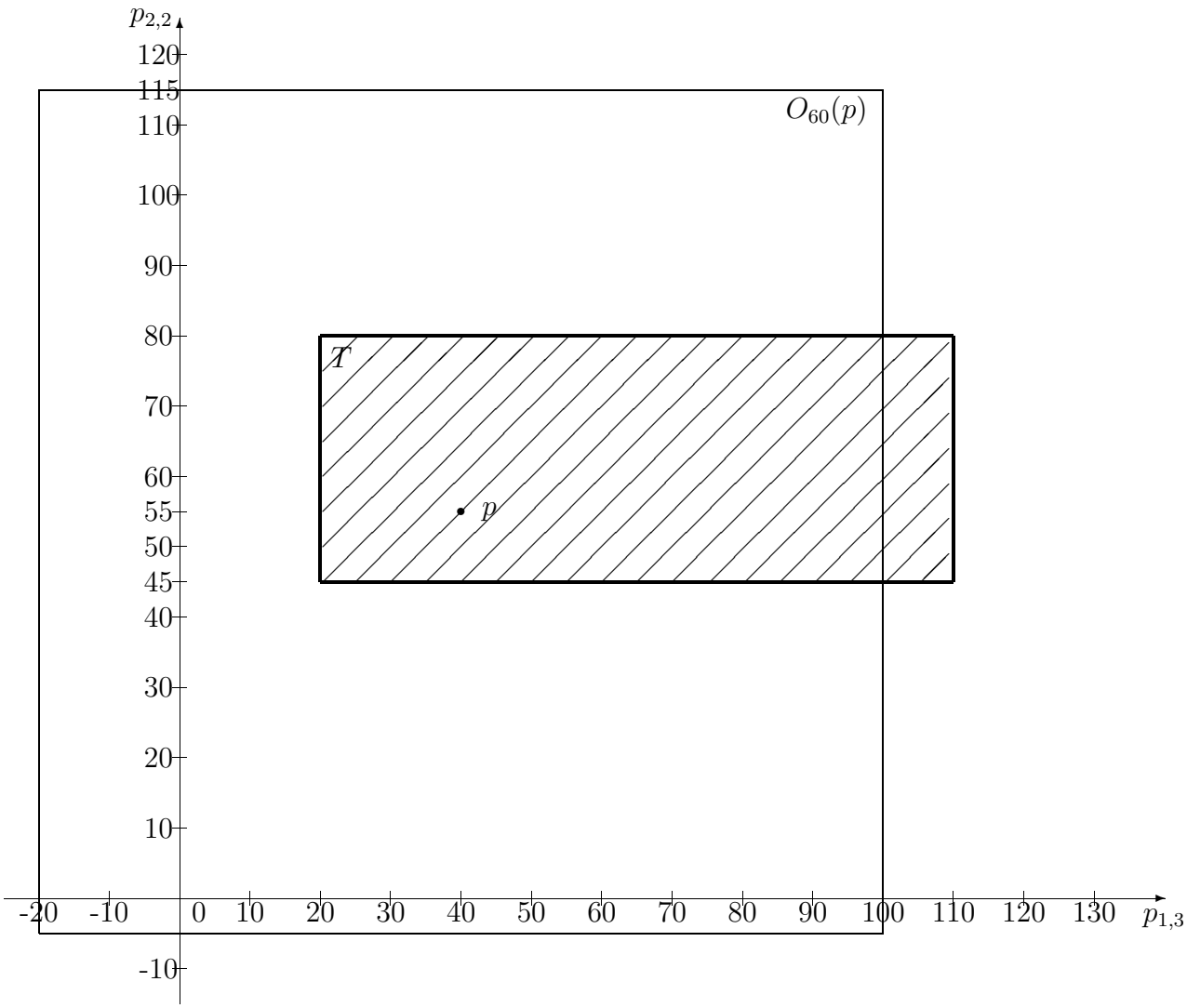


Figure 2.5: *Projections of the stability balls on the plane for problem* $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\mathcal{C}_{max}$ *constructed by Algorithm* $SOL\_\mathcal{C}_{max}(2)$

done during the calculation of $\hat{\varrho}_s^B(p \in T)$, one can use the upper bound for the relative stability radius $\hat{\varrho}_s^B(p \in T) \le \hat{r}_{ks}^B$, where $\hat{r}_{ks}^B$ is defined according to formula (2.25) on page 103.

Lemma 2.4 *If* $\hat{\varrho}_s^B(p \in T) < \infty$ *and there exists a digraph* $G_k \in B$ *such that*

$$\hat{r}_{ks}^B \le \frac{l_t^p - l_s^p}{q} \tag{2.34}$$

for a digraph $G_t \in B$*, then it is not necessary to consider digraph* G_t *during the calculation of the relative stability radius* $\hat{\varrho}_s^B(p \in T)$*.*

PROOF. To calculate the relative stability radius $\hat{\varrho}_s^B(p \in T)$, one can compare the optimal digraph G_s consecutively with each feasible digraph $G_i, i \ne s$, from set B. The value $\hat{r}_{ks}^B$ calculated according to (2.25) (see

page 103) shows that there exists a feasible digraph G_k, which becomes better than digraph G_s for some vector $p' \in T$ of the processing times if $d(p,p') = \hat{r}^B_{ks} + \epsilon$, where ϵ is a positive real number which may be as small as desired (see condition (2) introduced on page 104).

We show that, if the condition of Lemma 2.4 is satisfied, i.e., inequality (2.34) holds, then the value $\hat{r}^B_{ts}$ calculated for the digraph G_t does not improve the minimum in formula (2.24) (since inequalities $\hat{\varrho}^B_s(p \in T) \le \hat{r}^B_{ks} \le \hat{r}^B_{ts}$ hold). Compare digraph G_s, which is optimal for vector p of the processing times, with digraph G_t, $t \neq k$. From condition (1) on page 104 (condition (2), respectively), it follows that digraph G_t is a competitive digraph for G_s if the weight of each path $\nu \in H_t$ of digraph G_t becomes equal to the weight (smaller than the weight) of at least one path $\mu^* \in H_s$ of digraph G_s for some new vector $\hat{x} \in T$ (new vector $\hat{p}^\epsilon = \hat{x} \pm \epsilon \in T$, where the real number $\epsilon = d(\hat{x}, \hat{p}^\epsilon) > 0$ may be as small as desired). Hence, inequality

$$\max_{\mu^* \in H_s} l^{\hat{x}}(\mu^*) > \max_{\nu \in H_t} l^{\hat{x}}(\nu) \quad \Big(\max_{\mu^* \in H_s} l^{\hat{p}^\epsilon}(\mu^*) > \max_{\nu \in H_t} l^{\hat{p}^\epsilon}(\nu), \text{ respectively } \Big)$$

holds. This means that the critical weight of digraph G_t becomes smaller than that of digraph G_s for some feasible vector of the processing times. Such a 'superiority' of the competitive digraph G_t occurs for suitable changes of the processing times $\hat{p}^\epsilon_i = p_i \pm (\hat{r}^B_{ts} + \epsilon) = \hat{x}_i \pm \epsilon$, when the value $\hat{r}^B_{ts} = d(p, \hat{x})$ calculated in (2.25) reaches the minimum value in (2.24) (see condition 3). To this end, one must increase the weights of the vertices, which form a path $\mu^* \in H_s$, by the minimal value $\hat{r}^B_{ts}$ and decrease the weights of the vertices from set $[\nu^*] \setminus [\mu^*]$, $\nu^* \in H_t$, by the same value $\hat{r}^B_{ts}$ (according to formula (2.33)). Note that we must take such a path $\nu^* \in H_t$ for which the maximum in (2.25) is reached. So, for the competitive digraph G_t, the distance $d(p, \hat{x}) = \hat{r}^B_{ts}$ must achieve its minimal value in formula (2.24) among the distances between vector p and the other vectors in polytope T (i.e., the non-strict inequality $\hat{r}^B_{ts} \le \hat{r}^B_{ks}$ is also satisfied). Next, we show that, due to (2.34), value $\hat{r}^B_{ts}$ cannot be smaller than $\hat{r}^B_{ks}$ during the calculation of the relative stability radius $\hat{\varrho}^B_s(p \in T)$. Indeed:

$$\begin{aligned} \hat{r}^B_{ks} &\le \frac{l^p_t - l^p_s}{q} \le \frac{l^p_t - l^p(\mu^*)}{q} \le \frac{l^p_t - l^p(\mu^*)}{|[\nu^*] \setminus [\mu^*] + [\mu^*] \setminus [\nu^*]|} \\ &\le \max_{\nu \in H_t} \frac{l^p(\nu) - l^p(\mu^*)}{|[\mu^*] \cup [\nu]| - |[\mu^*] \cap [\nu]|} \le \min_{\mu \in H_s} \max_{\nu \in H_t} \frac{l^p(\nu) - l^p(\mu)}{|[\mu] + [\nu]|} \le \hat{r}^B_{ts}. \end{aligned}$$

Since $\hat{\varrho}^B_s(p \in T) \le \hat{r}^B_{ks} \le \hat{r}^B_{ts}$, the value $\hat{r}^B_{ts}$ cannot decrease the value $\hat{r}^B_{ks}$ in (2.24) and therefore, digraph G_t need not to be considered during the

calculation of the relative stability radius using formulas (2.24) and (2.25).

◇

Corollary 2.4 *Let set* $B = \{G_s = G_{i_1}, G_{i_2}, \ldots, G_{i_{|B|}}\}$ *be sorted in non-decreasing order of the objective function values:* $l^p_{i_1} \leq l^p_{i_2} \leq \ldots \leq l^p_{i_{|B|}}$. *If for the currently compared digraph* G_{i_k} *from set* $B \subseteq \Lambda(G)$ *inequality*

$$\hat{r}^B_{i_k s} \leq \frac{l^p_{i_t} - l^p_{i_1}}{q} \tag{2.35}$$

holds for digraph $G_{i_t} \in B$ *with* $l^p_{i_k} \leq l^p_{i_t}$, *then it is possible to exclude the digraphs* $G_{i_t}, G_{i_{t+1}}, \ldots, G_{i_{|B|}}$ *from further considerations during the calculation of the relative stability radius* $\hat{\varrho}^B_s(p \in T)$.

Proof. Since the digraphs in the set $B \subseteq \Lambda(G)$ are sorted in non-decreasing order of the objective function values and inequality (2.35) holds for digraph G_{i_t}, inequality $\hat{r}^B_{i_k s} \leq \frac{l^p_{i_j} - l^p_{i_1}}{q}$ holds for each digraph $G_{i_j}, j = t+1, t+2, \ldots, |B|$. Therefore, due to Lemma 2.4, these digraphs need not to be considered during the calculation of the relative stability radius $\hat{\varrho}^B_s(p \in T)$ (since we have the upper bound: $\hat{\varrho}^B_s(p \in T) \leq \hat{r}^B_{i_k s} \leq \frac{l^p_{i_j} - l^p_{i_1}}{q}$).

◇

Using Corollary 2.4, one can compare the optimal digraph $G_s = G_{i_1}$ consecutively with the digraphs $G_{i_2}, G_{i_3}, \ldots, G_{i_{|B|}}$ from set B in non-decreasing order of the objective function values: $l^p_{i_1} \leq l^p_{i_2} \leq \ldots \leq l^p_{i_{|B|}}$. If for the currently compared digraph $G_k = G_{i_r}$ inequality (2.34) holds, one can exclude the digraphs $G_{i_r}, G_{i_{r+1}}, \ldots, G_{i_{|B|}}$ from further considerations. Bound (2.34) is tight. Since $\hat{\varrho}_s(p) = \hat{\varrho}^{\Lambda(G)}_s(p \in R^q_+)$, Corollary 2.4 implies Corollary 2.5 which allows us to restrict the number of feasible digraphs while calculating the stability radius $\hat{\varrho}_s(p)$ (see Definition 1.2 on page 28).

Corollary 2.5 *Let set* $\Lambda(G) = \{G_s = G_{i_1}, G_{i_2}, \ldots, G_{i_\lambda}\}$ *be sorted in non-decreasing order of the objective function values:* $l^p_{i_1} \leq l^p_{i_2} \leq \ldots \leq l^p_{i_\lambda}$. *If for the currently compared digraph* G_{i_k} *from set* $\Lambda(G) = \{G_s = G_{i_1}, G_{i_2}, \ldots, G_{i_k}, \ldots, G_{i_t}, \ldots, G_{i_\lambda}\}$ *inequality*

$$\hat{r}^{\Lambda(G)}_{i_k s} \leq \frac{l^p_{i_t} - l^p_{i_1}}{q} \tag{2.36}$$

holds for digraph $G_{i_t} \in \Lambda(G)$ *with* $l^p_{i_k} \leq l^p_{i_t}$, *then it is possible to exclude digraphs* $G_{i_t}, G_{i_{t+1}}, \ldots, G_{i_\lambda}$ *from further considerations during the calculation of the stability radius* $\hat{\varrho}_s(p)$.

2.4. Dominance Relations

In this section, the job shop problem with the objective of minimizing the sum of job completion times under uncertain numerical input data is modeled in terms of a mixed graph. As far as practical scheduling is concerned, mean flow time is more important than the makespan criterion (see Section 1.7). Let us consider problem $\mathcal{J}/a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$: It is assumed that only the structural input data (i.e., precedence and capacity constraints) are fixed while for the operation processing times only their lower and upper bounds are known before scheduling and the probability distributions of the random processing times are unknown. The structural input data are defined by the technological routes of the jobs, e.g., for a flow or open shop fixing the structural input data simply means to fix the number of jobs and the number of machines. In this and the next sections, two variants of a branch-and-bound method are developed. The first one constructs a set of k schedules which are the best with respect to the mean flow time criterion (for a fixed vector of the processing times). The second variant constructs a set of *potentially optimal schedules* for all perturbations of the processing times within the given lower and upper bounds. To exclude redundant schedules, we use a stability analysis based on the pairwise comparison of schedules. Along with implicit enumerations based on a branch-and-bound method, we realize an explicit enumeration of all feasible schedules.

Let n jobs $J = \{J_1, J_2, \ldots, J_n\}$ have to be processed on m machines $M = \{M_1, M_2, \ldots, M_m\}$, the technological routes of the jobs being given (operation $O_{ij} \in Q_k^J$ has to be processed by machine $M_k \in M$). A machine can process at most one operation at a time (Condition 1 on page 11) and preemptions of an operation are forbidden (Condition 4 on page 12). Let us consider the job shop problem $\mathcal{J}/a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$ with fixed technological routes and uncertain (interval) processing times p_{ij}, $a_{ij} \leq p_{ij} \leq b_{ij}$, $J_i \in J$; $j = 1, 2, \ldots, n_i$ (Condition 5 on page 15). The sum of the job completion times (mean flow time) is the objective function $\Phi = \Phi(C_1, C_2, \ldots, C_n) = \sum_{i=1}^{n} C_i = \sum \mathcal{C}_i$, where $C_i = c_{in_i}$ is the completion time of job $J_i \in J$.

To present the structural input data for problem $\mathcal{J}/a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$, we use the mixed graph (Q^J, A^J, E^J) introduced on page 25. Such a mixed graph $G = (Q^J, A^J, E^J)$ defines the structural input data (precedence and capacity constraints) which are known before scheduling. A schedule is defined as a circuit-free digraph $G_s = (Q^J, A^J \cup E_s^J, \emptyset)$ generated from the mixed graph (Q^J, A^J, E^J) by replacing each edge $[O_{ij}, O_{uv}] \in E^J$ by one of the arcs (O_{ij}, O_{uv}) or (O_{uv}, O_{ij}). In the rest of this chapter, we use the

terms of an *optimal schedule (digraph)*, a *better* and a *best schedule (digraph)* with respect to the mean flow time criterion $\sum \mathcal{C}_i$. However, the makespan criterion $\mathcal{C}_{max}$ and a regular criterion Φ are considered as well.

Due to Definition 2.1 given on page 86, a *G-solution* $\Lambda^*(G)$ of problem $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$ is a set of digraphs containing at least one optimal digraph for each feasible vector $p = (p_{1,1}, p_{1,2}, \ldots, p_{nn_n}) \in T$ of the processing times, where $T = \{x = (x_{1,1}, x_{1,2}, \ldots, x_{nn_n}) : a_{ij} \le x_{ij} \le b_{ij}; i = 1, \ldots, n; j = 1, \ldots, n_i\}$ is the polytope of feasible vectors in the vector space R^q_+ with $q = |Q^J| = \sum_{i=1}^{n} n_i = \sum_{k=1}^{m} |Q^J_k|$. We shall look for a *minimal G-solution* $\Lambda^T(G)$ to problem $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$, i.e., for a minimal subset of the set $\Lambda(G)$ containing at least one optimal digraph for each fixed vector $p \in T$ of the processing times (see Definition 2.1 on page 86).

If the processing times p_{ij} of all operations $O_{ij} \in Q^J$ are fixed, one can calculate the value of the objective function for a digraph $G_s \in \Lambda(G)$ using the critical path method. As it follows from Section 1.2, to solve problem $\mathcal{J}//\Phi$ we must find a digraph G_s such that $\Phi^p_s = \min\{\Phi^p_k : \ k = 1, 2, \ldots, \lambda\}$ (see formula (1.8) on page 30), where

$$\Phi^p_k = \Phi(\max_{\nu \in H^1_k} l^p(\nu), \max_{\nu \in H^2_k} l^p(\nu), \ldots, \max_{\nu \in H^n_k} l^p(\nu))$$

is the value of the objective function of the job completion times for the digraph $G_k \in \Lambda(G)$ with fixed processing times $p \in R^q_+$, and $l^p(\mu)$ is the weight of path μ: $l^p(\mu) = \sum_{O_{ij} \in [\mu]} p_{ij}$. Remind that $\Phi^p_s = l^p_s$ for criterion $\mathcal{C}_{max}$ while $\Phi^p_s = L^p_s$ for criterion $\sum \mathcal{C}_i$.

As it has been proven in Chapter 1 (see Theorem 1.3 on page 36), there exists a problem $\mathcal{J}m/n = n^0/\mathcal{C}_{\max}$ with any given number of machines m and number of jobs $n = n^0$, for which the optimality of digraph $G_s \in \Lambda(G)$ does not depend on the numerical input data. In other words, relation $\widehat{\varrho}_s(p) = \infty$ holds, which means that schedule s minimizes the makespan for all non-negative processing times. However, such an optimal schedule cannot exist for criterion $\sum \mathcal{C}_i$: Each optimal digraph (for the mean flow time criterion) loses its optimality for some vectors $p \in R^q_+$ of the processing times, i.e., $\overline{\varrho}_s(p) < \infty$ (see Theorem 1.7 on page 49 and Remark 1.2). As it will be shown in the proof of Theorem 2.7 in the case of the relative stability radius $\overline{\varrho}^B_s(p \in T)$ (see Definition 2.6 below) when $T \subset R^q_+$ and $B \subset \Lambda(G)$, an unrestricted value of $\overline{\varrho}^B_s(p \in T)$ is still possible. For problem $\mathcal{J}/a_i \le p_i \le b_i/\Phi$, we introduce the following two transitive *dominance relations* which define partial orderings on the set of feasible digraphs $\Lambda(G)$.

Definition 2.5 *A feasible digraph G_s (strongly) dominates a feasible digraph G_k in the set $D \subseteq R_+^q$ if inequality $\Phi_s^p \leq \Phi_k^p$ (inequality $\Phi_s^p < \Phi_k^p$, respectively) holds for any vector $p \in D$ of the processing times. We denote the dominance relation by $G_s \preceq_D G_k$, and the strong dominance relation by $G_s \prec_D G_k$.*

Let equality $a_{ij} = b_{ij}$ hold for each operation $O_{ij} \in Q^J$. In other words, set T turns into one point: $T = \{a\}$, where $a = (a_{1,1}, a_{1,2}, \ldots, a_{nn_n})$. Such a problem $\mathcal{J}/a_i \leq p_i \leq b_i/\Phi$ becomes a deterministic problem $\mathcal{J}//\Phi$, and the dominance relation $\preceq_T$ defines a total ordering on the set of digraphs $\Lambda(G)$. Consequently, a minimal G-solution $\Lambda^T(G)$ consists of a single digraph: $\Lambda^T(G) = \{G_s\}$, where G_s is any optimal digraph for problem $\mathcal{J}//\Phi$ with the processing times p_{ij} being equal to $a_{ij} = b_{ij}$ for each operation $O_{ij} \in Q^J$. In other words, digraph G_s dominates all digraphs $G_k \in \Lambda(G)$ at point $a \in R_+^q$: $G_s \preceq_a G_k$. Moreover, if the strong dominance relation holds for each digraph $G_k \in \Lambda(G)$ at point a, i.e., if $G_s \prec_a G_k$, then digraph G_s is the unique optimal digraph for the processing times p_{ij} equal to $a_{ij} = b_{ij}$. As it was shown in the computational results (see Section 1.6 below), an optimal digraph for problem $\mathcal{J}//\sum \mathcal{C}_i$ was mainly uniquely determined. In such cases, if the dominance relation $G_s \preceq_a G_k$ is valid for each digraph $G_k \in \Lambda(G)$, then generally the strong dominance relation $G_s \prec_a G_k$ is valid for each digraph $G_k \in \Lambda(G)$ with $k \neq s$. (Note that this is not the case for the makespan criterion: For most job shop instances which have been randomly generated, makespan optimal digraphs were not uniquely determined.) In the general case of problem $\mathcal{J}/a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$, the operation processing times may vary between given lower and upper bounds and therefore, it is a priori unknown which path from set H_k^i will have the largest weight in a practical realization of schedule k corresponding to digraph G_k. Thus, we have to consider the whole set Ω_k^u of representatives of the family of sets $(H_k^i)_{J_i \in J}$ in a similar way to the approach considered for problem $\mathcal{J}//\sum \mathcal{C}_i$ (see Section 1.4).

Each of these sets Ω_k^u includes exactly one path from each set $H_k^i,\ J_i \in J$. Since $H_k^i \cap H_k^j = \emptyset$ for any pair of different jobs J_i and J_j, we have equality $|\Omega_k^u| = n$, and so there exist $\omega_k = \prod_{i=1}^n |H_k^i|$ different sets of representatives for digraph G_k, namely: $\Omega_k^1, \Omega_k^2, \ldots, \Omega_k^{\omega_k}$. Next, we show how to restrict the number of sets of representatives which have to be considered while solving problem $\mathcal{J}/a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$. Similarly to the notion of a *critical path* which is particularly important for criterion C_{max}, we use the notion of a *critical set* $\Omega_k^{u^*}$ for criterion $\sum C_i$. The set $\Omega_k^{u^*},\ u^* \in \{1, 2, \ldots, \omega_k\}$, is a critical set in

$G_k \in \Lambda(G)$ if the objective function value L_k^p is reached on this set of paths, i.e., if equality $\sum_{\nu \in \Omega_k^{u^*}} l^p(\nu) = L_k^p$ holds.

For different vectors $p \in R_+^q$ of the processing times, different sets $\Omega_k^u,\ u \in \{1, 2, \ldots, \omega_k\}$, may be *critical*, however a path $\nu \in H_k^i,\ J_i \in J$, may belong to a critical set only if $l^p(\nu) = \max_{\mu \in H_k^i} l^p(\mu)$. Therefore, while solving problem $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$, it is sufficient to consider only paths from the set H_k^i which may have the largest weight for at least one vector $p \in T$ of the processing times. Moreover, if there are two or more paths in H_k^i which have the largest weight at the same vector $p \in T$, it is sufficient to consider only one of them. Thus, it is sufficient to consider only *dominant* paths which were defined in Section 2.2 (see Definition 2.5 on page 95).

Using Corollary 2.2, one can simplify digraph G_s while solving problem $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$ or problem $\mathcal{J}/a_i \le p_i \le b_i/\mathcal{C}_{max}$. First, we delete all transitive arcs, then we delete some arcs on the base of a domination of path sets (see Definition 2.4).

Let $H_s^i(T)$ denote the set of all dominant paths in H_s^i with respect to the polytope T. Since $H_s \subseteq \cup_{i=1}^n H_s^i$ for problem $\mathcal{J}/a_i \le p_i \le b_i/\mathcal{C}_{max}$, one can construct a set $H_s^i(T)$ as a subset of the set of all dominant paths H_s by selecting all paths ending in vertex O_{in_i} (if they exist). Let $G_s^T = (Q_s^T, E_s^T, \emptyset)$ be a *minimal subgraph* of digraph G_s such that, if $\mu \in \cup_{i=1}^n H_s^i(T)$, then digraph G_s^T contains path μ. To construct the digraph G_s^T, one can use the following straightforward modification of the *critical path method* [94].

Let path μ have the maximal weight among all paths in digraph G_s ending in vertex O_{ij} when the processing times are defined by the vector $p \in R_+^q$. As usual, the weight of path μ minus p_{ij} is called the *earliest starting time* of operation O_{ij}, and we denote it by $l_s^p(O_{ij})$:

$$l_s^p(O_{ij}) = \sum_{O_{uv} \in [\mu] \setminus \{O_{ij}\}} p_{uv}.$$

The following recursive relations are obvious:

$$l_s^a(O_{ij}) = \max\{l_s^a(O_{uv}) + a_{uv} : (O_{uv}, O_{ij}) \in A^J \cup E_s^J\},$$

$$l_s^b(O_{ij}) = \max\{l_s^b(O_{uv}) + b_{uv} : (O_{uv}, O_{ij}) \in A^J \cup E_s^J\}.$$

Starting with a vertex in digraph G_s which has a zero in-degree and following the critical path method, we define values $l_s^a(O_{ij})$ and $l_s^b(O_{ij})$ for each vertex $O_{ij} \in Q^J$. Then, using *backtracking*, we define the vertices Q_s^T and the arcs E_s^T of digraph G_s^T as follows. Initially, we set $Q_s^T = \{O_{in_i} : J_i \in J\}$ for problem $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$ (if $H_s^i(T) \ne \emptyset$, $H_s^i(T) \subseteq H_s$, for problem

$\mathcal{J}/a_i \le p_i \le b_i/\mathcal{C}_{max}$) and we set $E_s^T = \emptyset$. Then we add the vertex O_{uv} to set Q_s^T, and we add the arc (O_{uv}, O_{in_i}) to set E_s^T:

$$Q_s^T := Q_s^T \cup \{O_{uv}\},\ E_s^T := E_s^T \cup \{(O_{uv}, O_{in_i})\}$$

if and only if the following two conditions hold.

1) There is no arc $(O_{u_1v_1}, O_{in_i})$ such that $l_s^b(O_{u_1v_1}) < l_s^a(O_{uv})$.

2) Inequality $l_s^b(O_{uv}) + b_{in_i} \ge l_s^a(O_{in_i})$ holds.

Continuing in a similar way for each vertex which is already included into set Q_s^T, we construct the digraph G_s^T (see Lemma 2.1).

Thus, instead of digraphs $G_k,\ k = 1, 2, \ldots, \lambda$, one can consider digraphs G_k^T which contain all dominant paths $\cup_{i=1}^n H_k^i(T)$ and which are often essentially simpler than the corresponding digraphs G_k. The transformation of digraph G_k into digraph G_k^T by testing inequality (2.6)

$$\sum_{O_{ij} \in [\mu] \setminus [\nu]} b_{ij} \le \sum_{O_{uv} \in [\nu] \setminus [\mu]} a_{uv}$$

(see page 96) takes $O(q^2)$ elementary steps (q is the number of operations).

Let for criterion $\sum \mathcal{C}_i$ the superscripts of the sets $\Omega_k^1, \Omega_k^2, \ldots, \Omega_k^{\omega_k^T}, \ldots, \Omega_k^{\omega_k}$ be such that for a path μ, inclusion $\mu \in \cup_{i=1}^n H_k^i(T)$ holds if and only if

$$\mu \in \bigcup_{i=1}^{\omega_k^T} \Omega_k^i, \quad \omega_k^T = \prod_{i=1}^n |H_k^i(T)|.$$

Example 2.2 *To illustrate the above notions and definitions, we introduce a job shop problem* $\mathcal{J}3/n = 3, a_i \le p_i \le b_i/\sum \mathcal{C}_i$ *with* $Q_1^J = \{O_{1,1}, O_{1,3}, O_{3,2}\}$, $Q_2^J = \{O_{1,2}, O_{2,1}, O_{3,3}\}$, *and* $Q_3^J = \{O_{2,2}, O_{3,1}\}$. *The mixed graph* $G = (Q^J, A^J, E^J)$ *represented in Figure 2.6 defines the structural input data. The numerical input data are defined by polytope* $T \in R_+^8$ *via Table 2.7. For this small example, one can explicitly enumerate all feasible digraphs of set* $\Lambda(G)$ *(the cardinality of set* $\Lambda(G)$ *is equal to 22). Since not all digraphs may be optimal for the given segments* $[a_{ij}, b_{ij}]$ *of the feasible variations of the processing times* p_{ij}, *we construct a subset* B *of the set* $\Lambda(G)$ *of possible candidates of competitive (optimal) digraphs using the algorithms from Section 2.6 below. The cardinality of set* B *is equal to 12, while* $\lambda = 22$.

Before finding a minimal G-solution $\Lambda^T(G)$ *for this problem* $\mathcal{J}3/n = 3, a_i \le p_i \le b_i/\sum \mathcal{C}_i$, *we consider its deterministic version* $\mathcal{J}3/n = 3/\sum \mathcal{C}_i$ *by setting the vector of the processing times to be equal to* $p^0 = (p_{1,1}^0, p_{1,2}^0, \ldots, p_{3,3}^0) \in T$ *with* $p_{1,1}^0 = 70, p_{1,2}^0 = 30, p_{1,3}^0 = 60, p_{2,1}^0 = 20, p_{2,2}^0 = 60, p_{3,1}^0 = 70, p_{3,2}^0 = 40$, *and* $p_{3,3}^0 = 30$ *(this vector can be arbitrarily chosen*

Table 2.7: *Numerical data for problem* $\mathcal{J}3/n=3, a_i \le p_i \le b_i/\sum \mathcal{C}_i$

i	1	1	1	2	2	3	3	3
j	1	2	3	1	2	1	2	3
a_{ij}	60	20	45	10	50	60	30	30
b_{ij}	80	40	60	30	70	80	50	40

from polytope T*.) We number the digraphs* $G_1, G_2, \ldots, G_{12}$ *in accordance with non-decreasing values of function* $\sum \mathcal{C}_i$ *calculated for the vector* p^0 *of processing times:*

$$L_1^{p^0} = 440, L_2^{p^0} = 470, L_3^{p^0} = 500, L_4^{p^0} = 500, L_5^{p^0} = 520, L_6^{p^0} = 530,$$

$$L_7^{p^0} = 540, L_8^{p^0} = 550, L_9^{p^0} = 570, L_{10}^{p^0} = 610, L_{11}^{p^0} = 610, L_{12}^{p^0} = 620.$$

For vector $p^0 \in T$*, the digraph* $G_1 = (Q^J, A^J \cup E_1^J, \emptyset)$ *with the signature* $E_1^J = \{(O_{1,1}, O_{3,2}), (O_{3,2}, O_{1,3}), (O_{2,1}, O_{1,2}), (O_{1,2}, O_{3,3}), (O_{2,1}, O_{3,3}), (O_{3,1}, O_{2,2})\}$ *is the only optimal digraph. Therefore, for the initial problem* $\mathcal{J}3/n = 3, a_i \le p_i \le b_i/\sum \mathcal{C}_i$*, we have to include digraph* G_1 *in the desired minimal G-solution* $\Lambda^T(G)$*. Using the critical path method, we simplify the digraphs* G_1*,* $G_2, \ldots, G_{12}$*. Then we compare the sets of representatives* $\Omega_1^1, \Omega_1^2, \ldots, \Omega_1^{\omega_1^T}$ *for digraph* G_1 *with the sets of representatives* $\Omega_k^1, \Omega_k^2, \ldots, \Omega_k^{\omega_k^T}$ *for the other digraphs* G_k*,* $k \in \{2, 3, \ldots, 12\}$*. Due to a pairwise comparison of these sets, we find that only two digraphs may be better than digraph* G_1 *(provided that vector* p *belongs to set* T *defined in Table 2.7). These two digraphs are as follows: Digraph* $G_2 = (Q^J, A^J \cup E_2^J, \emptyset)$ *with the signature*

$$E_2^J = \{(O_{1,1}, O_{3,2}), (O_{1,3}, O_{3,2}), (O_{2,1}, O_{1,2}), (O_{1,2}, O_{3,3}), (O_{2,1}, O_{3,3}), (O_{2,2}, O_{3,1})\}$$

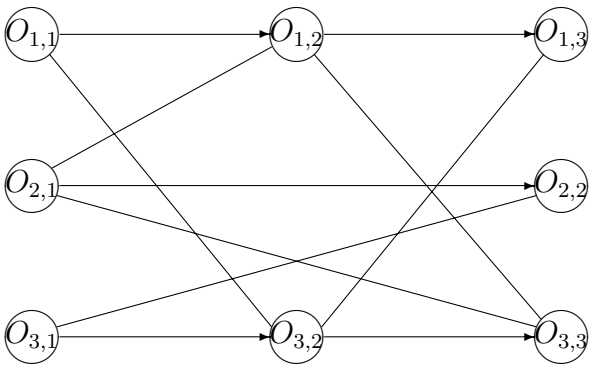


Figure 2.6: *Mixed graph* $G = (Q^J, A^J, E^J)$ *for problem* $\mathcal{J}3/n=3, a_i \le p_i \le b_i/\sum \mathcal{C}_i$

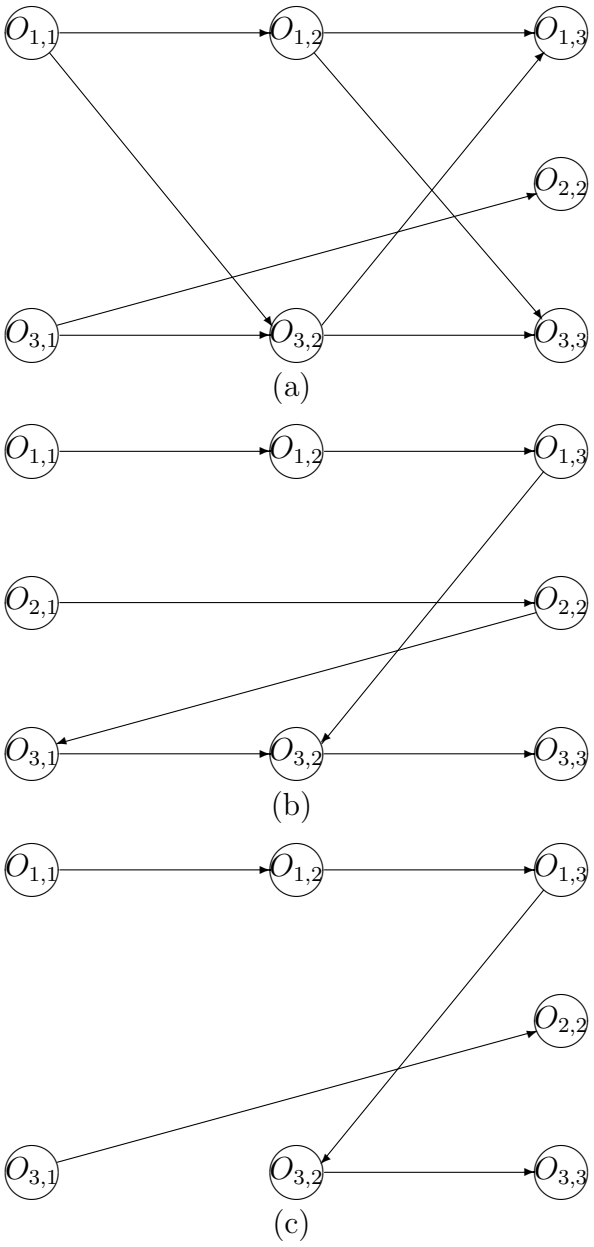


Figure 2.7: *Digraphs G_1^T, G_2^T and G_5^T which define a minimal G-solution $\Lambda^T(G)$ for Example 2.2*

and digraph $G_5 = (Q^J, A^J \cup E_5^J, \emptyset)$ with the signature

$$E_5^J = \{(O_{1,1},O_{3,2}), (O_{1,3},O_{3,2}), (O_{2,1},O_{1,2}), (O_{1,2},O_{3,3}), (O_{2,1}, O_{3,3}),(O_{3,1},O_{2,2})\}.$$

Moreover, digraph G_2 is the only optimal one for vector $p' = (60, 20, 60, 10, 60, 80, 40, 30) \in T$, and digraph G_5 is the only optimal one for vector $p'' = (60, 20, 45, 30, 70, 80, 50, 30) \in T$. Consequently, a minimal G-solution to problem $\mathcal{J}3/n=3, a_i \le p_i \le b_i/\sum \mathcal{C}_i$ consists of three digraphs, namely: $\Lambda^T(G) = \{G_1, G_2, G_5\}$. The corresponding digraphs G_1^T, G_2^T and

G_5^T are represented in Figure 2.7 a), b) and c), respectively. Note that while digraph G_1 has $\omega_1 = 4 \cdot 2 \cdot 5 = 40$ sets of representatives, digraph G_1^T has only $\omega_1^T = 3 \cdot 1 \cdot 3 = 9$ sets of representatives. For the digraphs G_2 and G_2^T, these numbers are $\omega_2 = 16$ and $\omega_2^T = 2$, and for the digraphs G_5 and G_5^T, these numbers are $\omega_5 = 28$ and $\omega_5^T = 1$.

The above explicit enumeration of the digraphs $\Lambda(G)$ is only possible for a small number of edges in the mixed graph G and for a practical use, one has to reduce the number of digraphs which have to be constructed. In particular, for the example under consideration, it is sufficient to construct only $k = 5$ digraphs, which are the best for the initial vector p^0 of the processing times. Further, in Section 2.6, such a calculation will be developed on the basis of a branch-and-bound method for constructing the k best digraphs. Moreover, the digraphs G_3 and G_4 in the set of the $k = 5$ best digraphs are also redundant. In Section 2.6, we present a branch-and-bound method for constructing all digraphs which are the only ones that may be optimal for feasible vectors of the processing times. We also show how to calculate the stability radius of an optimal digraph on the basis of an explicit enumeration of the digraphs $\Lambda(G)$. The calculation of the stability radius will be used in Sections 2.6 and 2.7 as the main procedure for finding a minimal G-solution of problem $\mathcal{J}/a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$.

2.5. Characterization of a G-solution

A characterization of a G-solution Λ of problem $\mathcal{J}/a_i \leq p_i \leq b_i/\Phi$ which is a proper subset of the set $\Lambda(G), \Lambda \subset \Lambda(G)$, may be obtained on the basis of the dominance relation $\preceq_D$ introduced in Section 2.4. Next, we prove necessary and sufficient conditions for a set of feasible digraphs to be a G-solution to problem $\mathcal{J}/a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$.

Theorem 2.4 *The set $\Lambda \subset \Lambda(G)$ is a G-solution to problem $\mathcal{J}/a_i \leq p_i \leq b_i/\Phi$ if and only if there exists a finite covering of polytope T by convex closed sets $D_j \subset R_+^q : T \subseteq \bigcup_{j=1}^d D_j,\ d \leq |\Lambda|$, such that for any digraph $G_k \in \Lambda(G)$ and for any set $D_j,\ j = 1, 2, \ldots, d$, there exists a digraph $G_s \in \Lambda$ for which dominance relation $G_s \preceq_{D_j} G_k$ holds.*

Proof. *Sufficiency.* For any fixed vector $p \in T$, one can find a set D_j, $1 \leq j \leq d$, such that $p \in D_j$. From Theorem 2.4, it follows that for any digraph $G_k \in \Lambda(G)$, there exists a digraph G_s such that dominance relation $G_s \preceq_{D_j} G_k$ holds. Hence, we have $\Phi_s^p \leq \Phi_k^p$ and so inequality $\min\{\Phi_s^p :$

$G_s \in \Lambda \subseteq \Lambda(G)\} \leq \Phi_k^p$ holds for each $k = 1, 2, \ldots, \lambda$. Consequently, for any vector $p \in T$ of the processing times, set Λ contains an optimal digraph.

Necessity. Let set $\Lambda \subseteq \Lambda(G)$ be a G-solution to problem $\mathcal{J}/a_i \leq p_i \leq b_i/\Phi$. We define a subset Λ' of set Λ such that each digraph $G_s \in \Lambda'$ is optimal for at least one vector $p \in T$ of the processing times.

For each digraph $G_s \in \Lambda$, there exists a *stability region*, i.e., the set of all vectors $p \in T \subseteq R_+^q$ for which digraph G_s is optimal. Let D_s be the intersection of the stability region of digraph G_s with the polytope T:

$$D_s = \{p \in R_+^q : \Phi_s^p \leq \Phi_k^p,\ k = 1, 2, \ldots, \lambda\} \cap T. \tag{2.37}$$

Since Λ' is a G-solution, we have $T \subseteq \bigcup_{j=1}^{|\Lambda'|} D_j \subset R_+^q$ and for each digraph $G_k \in \Lambda(G)$ and each set D_s, the dominance relation $G_s \preceq_{D_s} G_k$ holds. The inclusion $G_s \in \Lambda'$ implies $D_s \neq \emptyset$. From inequality (2.37), it follows that D_s is a closed set. Note that, if digraph G_s is optimal for vector p, it remains optimal for a feasible vector αp with any positive real number $\alpha > 0$. Consequently, the stability region is a convex set and so set D_s is convex as the intersection of convex sets.

$\diamond$

Theorem 2.4 implies the following claim characterizing a single-element G-solution to problem $\mathcal{J}/a_i \leq p_i \leq b_i/\Phi$, which is necessarily a minimal G-solution.

Corollary 2.6 *The equality $\Lambda^T(G) = \{G_s\}$ holds if and only if the dominance relation $G_s \preceq_T G_k$ holds for any digraph $G_k \in \Lambda(G)$.*

A minimal G-solution including more than one digraph is characterized on the basis of the strong dominance relation $\prec_D$ as follows.

Theorem 2.5 *Let set $\Lambda^*(G)$ be a G-solution to problem $\mathcal{J}/a_i \leq p_i \leq b_i/\Phi$ with $|\Lambda^*(G)| \geq 2$. This G-solution is minimal if and only if for each digraph $G_s \in \Lambda^*(G)$, there exists a vector $p^{(s)} \in T$ such that the strong dominance relation $G_s \prec_{p^{(s)}} G_k$ holds for each digraph $G_k \in \Lambda^*(G) \setminus \{G_s\}$.*

Proof. *Sufficiency.* If the condition of Theorem 2.5 holds, then for any digraph $G_s \in \Lambda^*(G)$, the set $\Lambda^*(G) \setminus \{G_s\}$ is no longer a G-solution to problem $\mathcal{J}/a_i \leq p_i \leq b_i/\Phi$. Indeed, for the above vector $p^{(s)} \in T$, inequality $\Phi_s^{p^{(s)}} < \Phi_k^{p^{(s)}}$ holds for each digraph $G_k \in \Lambda^*(G) \setminus \{G_s\}$. It follows that G_s is the unique optimal digraph to problem $\mathcal{J}//\Phi$ with the vector $p^{(s)}$ of processing times. Therefore, the G-solution $\Lambda^*(G)$ is minimal.

Necessity. Let $\Lambda^*(G)$ be a minimal G-solution but the condition of Theorem 2.5 does not hold, i.e., there exists a digraph $G_s \in \Lambda^*(G)$ such that for each vector $p^{(s)} \in T$, there exists a digraph $G_k \in \Lambda^*(G) \setminus \{G_s\}$ for which the strong dominance relation $G_s \prec_{p^{(s)}} G_k$ does not hold, i.e., $\Phi_s^p \geq \Phi_k^p$. It follows that the set $\Lambda^*(G) \setminus \{G_s\}$ is also a G-solution to problem $\mathcal{J}/a_i \leq p_i \leq b_i/\Phi$ (since the set $\Lambda^*(G)$ is supposed to be a G-solution). Thus, we get a contradiction to the assumption that the G-solution $\Lambda^*(G)$ is minimal.

$\diamond$

Section 2.6 deals with different algorithms for finding a G-solution and a minimal G-solution on the basis of an explicit or an implicit enumeration of feasible digraphs. All algorithms developed are based on the fact that a digraph $G_s \in \Lambda(G)$ being optimal for the fixed vector $p \in R_+^q$ of the processing times, generally remains optimal within some neighborhood of the point p in the space R_+^q: Digraph G_s dominates all digraphs in a neighborhood of p. We consider the closed ball $O_r(p) \subset R^q$ with the center $p \in T$ and the radius $r > 0$ as the neighborhood of the point $p \in T \subset R_+^q$ in the space R^q. Next, we rewrite some basic notions using the dominance relation $\preceq_D$.

The closed ball $O_r(p)$ is called a *stability ball* of digraph G_s if this digraph dominates all digraphs $G_k \in \Lambda(G)$ in $T^* = O_r(p) \cap T$, i.e., if $G_s \preceq_{T^*} G_k$ for each digraph $G_k \in \Lambda(G)$ (in this case, from Corollary 2.6, it follows that $\Lambda^{T^*}(G) = \{G_s\}$). The radius r of a stability ball may be interpreted as the *error* of the given processing times $p = (p_{1,1}, p_{1,2}, \ldots, p_{nn_n}) \in R_+^q$ such that for all processing times $x = (x_{1,1}, x_{1,2}, \ldots, x_{nn_n}) \in R_+^q$ with $p_{ij} - r \leq x_{ij} \leq p_{ij} + r$, digraph G_s remains the best. Similarly to Definition 2.2 of the relative stability radius for the makespan criterion, we give the definition of the relative stability radius for the mean flow time criterion.

Definition 2.6 *Assume that for each vector $p' \in O_\varrho(p) \cap T$, digraph $G_s \in B \subseteq \Lambda(G)$ with the vector p' of weights has the minimal critical sum of weights $L_s^{p'}$ among all digraphs of the set B. The maximal value of the radius ϱ of such a ball $O_\varrho(p)$ is denoted by $\overline{\varrho}_s^B(p \in T)$, and it is called the relative stability radius of digraph G_s with respect to polytope T.*

Remark 2.6 From Definition 2.5 and Definition 2.6, it follows that the *relative stability radius* $\overline{\varrho}_s^B(p \in T)$ of digraph $G_s \in B$ is equal to the maximal value of the radius ϱ of a ball $O_\varrho(p)$ such that for each digraph $G_k \in B \subseteq \Lambda(G)$, dominance relation $G_s \preceq_{T^*} G_k$ holds where $T^* = O_\varrho(p) \cap T$.

Similarly to Section 1.4, which deals with the stability radius $\overline{\varrho}_s(p)$ (see conditions (1.33) and (1.34) on page 43), to find the relative stability radius

$\overline{\varrho}_s^B(p \in T)$ for problem $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$, it is sufficient to construct a vector $\overline{x} = (\overline{x}_{1,1}, \overline{x}_{1,2}, \ldots, \overline{x}_{nn_n}) \in T \subseteq R_+^q$ which satisfies the following three conditions.

(a^*) There exists a digraph $G_k(p) \in B, k \neq s$, such that $L_s^{\overline{x}} = L_k^{\overline{x}}$, i.e.,

$$\sum_{i=1}^{n} \max_{\mu \in H_s^i} l^{\overline{x}}(\mu) = \sum_{i=1}^{n} \max_{\nu \in H_k^i} l^{\overline{x}}(\nu). \tag{2.38}$$

(b^*) For any given real $\epsilon > 0$, which may be as small as desired, there exists a vector $\overline{p}^{\epsilon} \in T$ such that $d(\overline{x}, \overline{p}^{\epsilon}) = \epsilon$ and $L_s^{\overline{p}^{\epsilon}} > L_k^{\overline{p}^{\epsilon}}$, i.e., inequality

$$\sum_{i=1}^{n} \max_{\mu \in H_s^i} l^{\overline{p}^{\epsilon}}(\mu) > \sum_{i=1}^{n} \max_{\nu \in H_k^i} l^{\overline{p}^{\epsilon}}(\nu) \tag{2.39}$$

is satisfied for at least one digraph $G_k(p) \in B$.

(c^*) The distance $d(p, \overline{x})$ achieves its minimal value among the distances between the vector p and the other vectors in the polytope T which satisfy both conditions (a^*) and (b^*).

Next, we describe the calculation of the relative stability radius $\overline{\varrho}_s^B(p \in T)$ using the above notation of the dominance relation. To this end, we prove Lemma 2.5 below about the dominance relation $\preceq_T$, and then we derive a formula for the calculation of the relative stability radius $\overline{\varrho}_s^B(p \in T)$ which is presented in Theorem 2.6.

If $\Lambda^T(G) = \{G_s\}$, then digraph G_s dominates all digraphs in the polytope T (see Corollary 2.6). In such a case, we assume that $\overline{\varrho}_s^{\Lambda(G)}(p \in T) = \infty$, since digraph G_s remains the best for all variable feasible vectors $x \in T$ of the processing times. Otherwise, there exists a digraph $G_k \in \Lambda(G)$ such that dominance relation $G_s \preceq_T G_k$ does not hold, and from Corollary 2.6 and Remark 2.6, it follows that the stability radius $\overline{\varrho}_s^{\Lambda(G)}(p \in T)$ has to be finite, i.e., there exists a vector $\overline{p}^{\epsilon} \in T$ such that inequality

$$L_s^{\overline{p}^{\epsilon}} > L_k^{\overline{p}^{\epsilon}} \tag{2.40}$$

holds. To calculate the stability radius $\overline{\varrho}_s^B(p \in T)$, $B \subseteq \Lambda(G)$, we will consider digraphs $G_k \in B$ such that dominance relation $G_s \preceq_T G_k$ does not hold, and for each of these digraphs G_k, we will look for the vector $\overline{p}^{\epsilon} \in T$ which is the closest to p among all vectors for which inequality (2.40) holds (see condition (c^*)). The following lemma allows us to restrict the set of digraphs $G_k \in B$ which have to be treated for any given regular criterion.

Lemma 2.5 *Digraph $G_s \in B$ dominates digraph $G_k \in B$ in the polytope T if (only if) inequality (2.41) holds (inequalities (2.42) hold, respectively):*

$$L_s^b \le L_k^a \tag{2.41}$$

$$(L_s^a \le L_k^a,\ L_s^b \le L_k^b). \tag{2.42}$$

PROOF. *Sufficiency.* Since the objective function is non-decreasing, it follows from inequality (2.41) that

$$L_s^x \le \sum_{i=1}^{n} l_s^b(O_{in_i}) = L_s^b \le L_k^a = \sum_{i=1}^{n} l_k^a(O_{in_i}) \le L_k^x$$

for any vector $x \in T$. Therefore, dominance relation $G_s \preceq_T G_k$ holds.

Necessity. Dominance relation $G_s \preceq_T G_k$ means that inequality $L_s^x \le L_k^x$ holds for any vector $x \in T$. In particular, inequality $L_s^x \le L_k^x$ holds for both vectors $a \in T$ and $b \in T$, i.e., inequalities (2.42) hold.

$\diamond$

To verify inequalities (2.41) and (2.42) takes $O(q^2)$ elementary steps, however, there is a 'gap' between the necessary and sufficient conditions of Lemma 2.5, if $L_s^a \ne L_s^b$. To overcome this gap for problem $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$, we are forced to compare the sets $\Omega_s^v,\ v = 1, 2, \ldots, \omega_s^T$, with the sets $\Omega_k^u,\ u = 1, 2, \ldots, \omega_k^T$, since we do not know a priori which set of paths will be critical for the factual processing times.

First, we will find a vector $\overline{x} = (\overline{x}_{1,1}, \overline{x}_{1,2}, \ldots, \overline{x}_{nn_n}) \in T$, which is the closest to vector $p \in T$ such that $L_s^{\overline{x}} = L_k^{\overline{x}}$ (see condition (a^*)). For the desired vector $\overline{x}$, the value $\sum_{\nu \in \Omega_k^u} l^{\overline{x}}(\nu)$ for each set $\Omega_k^u,\ u = 1, 2, \ldots, \omega_k^T$, has to be not greater than the value $\sum_{\mu \in \Omega_s^v} l^{\overline{x}}(\mu)$ for at least one set $\Omega_s^v,\ v = 1, 2, \ldots, \omega_s^T$. If the opposite inequality holds for the given vector $p \in T$, i.e., if $\sum_{\mu \in \Omega_s^v} l^p(\mu) < \sum_{\nu \in \Omega_k^u} l^p(\nu)$, we can calculate the value

$$r = \frac{\sum_{\nu \in \Omega_k^u} l^p(\nu) - \sum_{\mu \in \Omega_s^v} l^p(\mu)}{\sum_{O_{ij} \in Q} |n_{ij}(\Omega_k^u) - n_{ij}(\Omega_s^v)|} \tag{2.43}$$

(where $n_{ij}(\Omega_k^u)$ is the number of copies of operation O_{ij} in the multiset $\{[\nu] :\ \nu \in \Omega_k^u\}$) in order to obtain vector $\overline{x}$ with the equality

$$\sum_{\mu \in \Omega_s^v} l^{\overline{x}}(\mu) = \sum_{\nu \in \Omega_k^u} l^{\overline{x}}(\nu). \tag{2.44}$$

It is easy to convince that equality (2.44) holds for the vector $\overline{x}$ obtained from vector p by adding the value r calculated in (2.43) to all components p_{ij} with $n_{ij}(\Omega_k^u) < n_{ij}(\Omega_s^v)$, and by subtracting the same value r from all components p_{ij} with $n_{ij}(\Omega_k^u) > n_{ij}(\Omega_s^v)$. For the above vector $\overline{x}$, inclusion $\overline{x} \in T$ may not hold. To guarantee the latter inclusion, we have to look for a vector $\overline{x}$ in the form $\overline{x} = p(r) = (p_{1,1}(r), p_{1,2}(r), \ldots, p_{nn_n}(r))$, where

$$\overline{x}_{ij} = p_{ij}(r) = \begin{cases} p_{ij} + \min\{r, b_{ij} - p_{ij}\}, & \text{if } n_{ij}(\Omega_k^u) < n_{ij}(\Omega_s^v), \\ p_{ij} - \min\{r, p_{ij} - a_{ij}\}, & \text{if } n_{ij}(\Omega_k^u) > n_{ij}(\Omega_s^v), \\ p_{ij}, & \text{if } n_{ij}(\Omega_k^u) = n_{ij}(\Omega_s^v). \end{cases} \tag{2.45}$$

Let $r_{\Omega_k^u,\Omega_s^v}$ denote the minimal distance between the given vector $p \in T$ and the desired vector $\overline{x} = p(r) \in T$ for which equality (2.44) holds: $r_{\Omega_k^u,\Omega_s^v} = d(p, p(r))$. Next, we show how to calculate $r_{\Omega_k^u,\Omega_s^v}$. We define the value

$$\Delta^{ij}(\Omega_s^v, \Omega_k^u) = \begin{cases} b_{ij} - p_{ij}, & \text{if } n_{ij}(\Omega_k^u) < n_{ij}(\Omega_s^v), \\ p_{ij} - a_{ij}, & \text{if } n_{ij}(\Omega_k^u) > n_{ij}(\Omega_s^v), \end{cases} \tag{2.46}$$

for each operation $O_{ij} \in N(\Omega_k^u, \Omega_s^v) = \{\cup_{\mu \in \Omega_k^u \cup \Omega_s^v}[\mu] : \ n_{ij}(\Omega_k^u) \neq n_{ij}(\Omega_s^v)\}$, which we put in non-decreasing order:

$$\Delta_1^{ij}(\Omega_s^v, \Omega_k^u) \leq \Delta_2^{ij}(\Omega_s^v, \Omega_k^u) \leq \ldots \leq \Delta_{|N(\Omega_s^v,\Omega_k^u)|}^{ij}(\Omega_s^v, \Omega_k^u). \tag{2.47}$$

Note that each value $\Delta_\alpha^{ij}(\Omega_s^v, \Omega_k^u)$ is calculated according to (2.46) for all different operations O_{ij}, and the subscript $\alpha = 1, 2, \ldots, |N(\Omega_s^v, \Omega_k^u)|$ indicates the location of value (2.46) in the sequence (2.47). We define value $N_\alpha(\Delta) = |n_{ij}(\Omega_k^u) - n_{ij}(\Omega_s^v)|$ for each $\Delta_\alpha^{ij}(\Omega_s^v, \Omega_k^u), \alpha \in \{1, 2, \ldots, |N(\Omega_s^v, \Omega_k^u)|\}$, and assume $\Delta_0^{ij}(\Omega_s^v, \Omega_k^u) = 0$ and $N_0(\Delta) = 0$. From (2.45) and (2.47), it follows:

$$r_{\Omega_s^v,\Omega_k^u} = \max_{\beta=0,1,\ldots,|N(\Omega_s^v,\Omega_k^u)|-1} \frac{\sum_{\nu \in \Omega_k^u} l^p(\nu) - \sum_{\mu \in \Omega_s^v} l^p(\mu) - \sum_{\alpha=0}^{\beta} \Delta_\alpha^{ij}(\Omega_s^v, \Omega_k^u)\ N_\alpha(\Delta)}{\sum_{O_{ij} \in Q^J} |n_{ij}(\Omega_k^u) - n_{ij}(\Omega_s^v)| - \sum_{\alpha=0}^{\beta} N_\alpha(\Delta)}. \tag{2.48}$$

To ensure equality $L_s^{\overline{x}} = L_k^{\overline{x}}$ for digraph G_k and vector $\overline{x} = p(r) \in T$, we have to repeat the calculations (2.45) – (2.48) for each set $\Omega_k^u,\ u \in \{1, 2, \ldots, \omega_k^T\}$, with $\sum_{\nu \in \Omega_k^u} l^p(\nu) \geq L_s^p$. Then we have to take the maximum of $r_{\Omega_s^v,\Omega_k^u}$, for each set $\Omega_k^u,\ u \in \{1, 2, \ldots, \omega_k^T\}$, and to take the minimum of the maxima obtained:

$$\overline{r}_{ks}^B = \min_{v \in \{1,2,\ldots,\omega_s^T\}} \max\{r_{\Omega_s^v,\Omega_k^u} : u \in \{1, 2, \ldots, \omega_k^T\}, \sum_{\nu \in \Omega_k^u} l^p(\nu) \geq L_s^p\}. \tag{2.49}$$

Even if there exists a vector $\overline{x} \in T$ such that equality $L_s^{\overline{x}} = L_k^{\overline{x}}$ holds (see condition (a^*)), nevertheless, it may be that there exists no vector $\overline{p}^\epsilon \in T$ defined in condition (b^*) such that $L_s^{\overline{p}^\epsilon} > L_k^{\overline{p}^\epsilon}$. As follows from Definition 2.5, only inequality (2.39) guarantees that digraph G_s does not dominate digraph G_k in polytope T. Therefore, we have to look for a vector $\overline{p}^\epsilon \in T$ such that inequality (2.39) holds which may be rewritten in the equivalent form:

$$\max_{v \in \{1,2,\ldots,\omega_s^T\}} \sum_{\mu \in \Omega_s^v} l^{\overline{p}^\epsilon}(\mu) > \max_{u \in \{1,2,\ldots,\omega_k^T\}} \sum_{\nu \in \Omega_k^u} l^{\overline{p}^\epsilon}(\nu). \tag{2.50}$$

Remark 2.7 It is easy to see that there exists a vector $\overline{p}^\epsilon \in T$ such that

$$\sum_{\mu \in \Omega_s^v} l^{\overline{p}^\epsilon}(\mu) > \sum_{\nu \in \Omega_k^u} l^{\overline{p}^\epsilon}(\nu) \tag{2.51}$$

if and only if inequality (2.51) holds for vector $\overline{p}^{\epsilon} = p^* = (p^*_{1,1}, p^*_{1,2}, \ldots, p^*_{nn_n}) \in T$, where

$$p^*_{ij} = \begin{cases} b_{ij}, & \text{if } n_{ij}(\Omega^u_k) < n_{ij}(\Omega^v_s), \\ a_{ij}, & \text{if } n_{ij}(\Omega^u_k) > n_{ij}(\Omega^v_s), \\ p_{ij}, & \text{if } n_{ij}(\Omega^u_k) = n_{ij}(\Omega^v_s). \end{cases} \tag{2.52}$$

Indeed, the components of vector $p^* \in T$ with $n_{ij}(\Omega^u_k) < n_{ij}(\Omega^v_s)$ are as large as possible and the components with $n_{ij}(\Omega^u_k) > n_{ij}(\Omega^v_s)$ are as small as possible in the polytope T, while changing the components with $n_{ij}(\Omega^u_k) = n_{ij}(\Omega^v_s)$ does not influence the difference $\sum_{\nu\in\Omega^u_k} l^x(\nu) - \sum_{\mu\in\Omega^v_s} l^x(\mu)$. Thus, we can restrict the consideration of the sets Ω^v_s in inequality (2.50) to the following subset Ω^*_{sk} of set $\{\Omega^v_s : \ v = 1, 2, \ldots, \omega^T_s\}$, where Ω^*_{sk} is the set of all sets of representatives $\Omega^v_s,\ v \in \{1, 2, \ldots, \omega^T_s\}$, for which inequality

$$\sum_{\mu\in\Omega^v_s} l^{p^*}(\mu) > \sum_{\nu\in\Omega^u_k} l^{p^*}(\nu) \tag{2.53}$$

holds for each set of representatives $\Omega^u_k,\ u \in \{1, 2, \ldots, \omega^T_k\}$. Sufficiency in Lemma 2.5 may be generalized as follows.

Lemma 2.6 *Digraph $G_s \in B$ dominates digraph $G_k \in B$ in the polytope T if $\Omega^*_{sk} = \emptyset$.*

Proof. Equality $\Omega^*_{sk} = \emptyset$ implies that for each set of representatives $\Omega^v_s,\ v \in \{1, 2, \ldots, \omega^T_s\}$, inequality

$$\sum_{\mu\in\Omega^v_s} l^{p^*}(\mu) \le \sum_{\nu\in\Omega^u_k} l^{p^*}(\nu)$$

holds for each set of representatives $\Omega^u_k,\ u \in \{1, 2, \ldots, \omega^T_k\}$. Due to Remark 2.7, inequality

$$\sum_{\mu\in\Omega^v_s} l^{\overline{p}^{\epsilon}}(\mu) \le \sum_{\nu\in\Omega^u_k} l^{\overline{p}^{\epsilon}}(\nu)$$

holds for each vector $\overline{p}^{\epsilon} \in T$. Hence, $L^{\overline{p}^{\epsilon}}_s \le L^{\overline{p}^{\epsilon}}_k$ and so dominance relation $G_s \preceq_T G_k$ holds.

◇

Due to Lemma 2.6, we can rewrite equality (2.49) as follows:

$$\overline{r}^B_{ks} = \min_{\Omega^v_s\in\Omega^*_{sk}} \max\{r_{\Omega^v_s,\Omega^u_k} : u \in \{1, 2, \ldots, \omega^T_k\}, \sum_{\nu\in\Omega^u_k} l^p(\nu) \ge L^p_s\}. \tag{2.54}$$

To obtain the desired vector $\overline{p}^{\epsilon} \in T$, we have to calculate $\overline{r}^B_{ks}$ according to (2.54) for each digraph $G_k \in B$ which is not dominated by digraph G_s (if $G_s \not\preceq_T G_k$) and to take the minimum over all such digraphs G_k. We summarize the above arguments in the following claim.

Theorem 2.6 *If for digraph $G_s \in B \subseteq \Lambda(G)$, dominance relation $G_s \preceq_p G_k$ holds for each digraph $G_k \in B$ and fixed vector $p \in T$, then equalities*

$$\begin{aligned} \overline{\varrho}_s^B(p \in T) &= \min \left\{ \min_{\Omega_s^v \in \Omega_{sk}^*} \max_{\substack{u \in \{1,2,\ldots,\omega_k^T\}, \\ \sum_{\nu \in \Omega_k^u} l^p(\nu) \geq L_s^p}} r_{\Omega_s^v, \Omega_k^u} : G_s \not\preceq_T G_k \right\} \quad (2.55) \\ &= \min\{\overline{r}_{ks}^B : G_s \not\preceq_T G_k\} \end{aligned}$$

hold, where value $r_{\Omega_s^v, \Omega_k^u}$ is calculated according to (2.48).

The following corollary is used in the proof of Theorem 2.8 below.

Corollary 2.7 *The value $r_{\Omega_s^{v^0}, \Omega_k^u}$ calculated according to (2.48) for the set $\Omega_s^{v^0} \in \Omega_{sk}^* \setminus \Omega_s(p)$ is strongly positive.*

Proof. Due to formula (2.55), we have to repeat the calculation (2.48) for each set $\Omega_s^v \in \Omega_{sk}^*$ and each set $\Omega_k^{u^0}, u^0 \in \{1, 2, \ldots, \omega_k^T\}$, such that $\sum_{\nu \in \Omega_k^{u^0}} l^p(\nu) \geq L_s^p$. Since there exists a set $\Omega_s^{v^0} \in \Omega_{sk}^* \setminus \Omega_s(p)$, i.e., $\sum_{\mu \in \Omega_s^{v^0}} l^p(\mu) < L_s^p$, the following inequalities hold:

$$\begin{aligned} &\sum_{\nu \in \Omega_k^{u^0}} l^p(\nu) - \sum_{\mu \in \Omega_s^{v^0}} l^p(\mu) \min_{\substack{u^0 \in \{1,2,\ldots,\omega_k^T\}, \\ \sum_{\nu \in \Omega_k^{u^0}} l^p(\nu) \geq L_s^p}} \sum_{\nu \in \Omega_k^{u^0}} l^p(\nu) - \max_{\substack{v^0 \in \{1,2,\ldots,\omega_s^T\}, \\ \sum_{\nu \in \Omega_s^{v^0}} l^p(\mu) < L_s^p}} \sum_{\mu \in \Omega_s^{v^0}} l^p(\mu) \\ \geq\ & L_s^p - \max_{\substack{v^0 \in \{1,2,\ldots,\omega_s^T\}, \\ \sum_{\nu \in \Omega_s^{v^0}} l^p(\mu) < L_s^p}} \sum_{\mu \in \Omega_s^{v^0}} l^p(\mu) > 0. \end{aligned}$$

Therefore, due to the calculation of the value $r_{\Omega_s^{v^0}, \Omega_k^{u^0}}$, the numerator in (2.48) is strongly positive at least for $\beta = 0$. Since we have to take the maximum value among all values calculated for each $\beta = 0, 1, \ldots, |N(\Omega_s^{v^0}, \Omega_k^{u^0})| - 1$ (see formula (2.48)), we obtain $r_{\Omega_s^{v^0}, \Omega_k^{u^0}} > 0$.

◇

Next, we present necessary and sufficient conditions for an infinitely large relative stability radius $\overline{\varrho}_s^B(p \in T)$ for problem $\mathcal{J}/a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$ if $B \subset \Lambda(G)$ and $T \subset R_+^q$. Note that the deterministic problem $\mathcal{J}//\sum \mathcal{C}_i$ with $\lambda > 1$ cannot have an optimal digraph with an infinitely large stability radius $\overline{\varrho}_s(p)$ (see Remark 1.2). Recall that $\overline{\varrho}_s(p) = \overline{\varrho}_s^{\Lambda(G)}(p \in R_+^q)$.

Theorem 2.7 *For digraph $G_s \in B \subseteq \Lambda(G)$, we have $\overline{\varrho}_s^B(p \in T) = \infty$ if and only if equality $\Omega_{sk}^* = \emptyset$ holds for each digraph $G_k \in B$.*

PROOF. *Necessity.* Following the contradiction method, we suppose that $\overline{\varrho}_s^B(p \in T) = \infty$ but there exists a digraph $G_k \in B$ such that the set of representatives $\Omega_s^{v^0}, v^0 \in \{1, 2, \ldots, \omega_s^T\}$, belongs to set Ω_{sk}^*.

It follows that inequality

$$\sum_{\mu \in \Omega_s^{v^0}} l^{p^*}(\mu) > \sum_{\nu \in \Omega_k^u} l^{p^*}(\nu)$$

holds for the vector p^* calculated according to formula (2.52) for the set of representatives $\Omega_k^u, u \in \{1, 2, \ldots, \omega_k^T\}$. Thus, due to Remark 2.7, there exists a vector $p' \in T$ such that

$$\sum_{\mu \in \Omega_s^{v^0}} l^{p'}(\mu) > \sum_{\nu \in \Omega_k^u} l^{p'}(\nu).$$

Since the latter inequality holds for all sets Ω_k^u, $u \in \{1, 2, \ldots, \omega_k^T\}$, this inequality holds for a critical set $\Omega_k^{u^*} \in \Omega_k(p)$ as well. Therefore, we obtain

$$\sum_{\mu \in \Omega_s^{v^0}} l^{p'}(\mu) > \sum_{\nu \in \Omega_k^{u^*}} l^{p'}(\nu) = L_k^{p'}$$

and hence, digraph G_s cannot be optimal for the processing times given by vector $p' \in T$. We get a contradiction as follows:

$$\overline{\varrho}_s^B(p \in T) < d(p, p') \leq \max_{O_{ij} \in Q^J} \{b_{ij} - p_{ij}, p_{ij} - a_{ij}\} < \infty.$$

Sufficiency. Due to Lemma 2.6, equality $\Omega_{sk}^* = \emptyset$ (valid for each digraph $G_k \in B$) implies that digraph $G_s \in B$ dominates all digraphs $G_k \in B$ in polytope T. Hence, inequality $L_s^{p'} \leq L_k^{p'}$ holds for each vector $p' \in T$ and so $\overline{\varrho}_s^B(p \in T) = \infty$.

$\diamond$

From the above proof of necessity, we obtain an upper bound for the relative stability radius $\overline{\varrho}_s^B(p \in T)$ as follows.

Corollary 2.8 *If* $\overline{\varrho}_s^B(p \in T) < \infty$, *then*

$$\overline{\varrho}_s^B(p \in T) \leq \max\{\{b_{ij} - p_{ij}, p_{ij} - a_{ij}\} : \ O_{ij} \in Q^J\}.$$

Moreover, we can strengthen Corollary 2.6.

Corollary 2.9 *The following propositions are equivalent:*

1) $\Lambda^T(G) = \{G_s\}$,

2) $\overline{\varrho}_s^{\Lambda(G)}(p \in T) = \infty$,

3) $G_k \in \Lambda(G) \ \Rightarrow \ G_s \preceq_T G_k$,

4) $G_k \in \Lambda(G) \ \Rightarrow \ \Omega_{sk}^* = \emptyset$.

To present necessary and sufficient conditions for $\overline{\varrho}_s^B(p \in T) = 0$, we need the following auxiliary lemma. Let Ω_k denote the set $\{\Omega_k^u : \ u = 1, 2, \ldots, \omega_k\}$.

Lemma 2.7 *If $\Omega_k \neq \Omega_k(p)$, inclusion $\Omega_k(p') \subseteq \Omega_k(p)$ holds for any vector $p' \in O_\epsilon(p) \cap R_+^q$ with a real $\overline{\epsilon}_k > \epsilon > 0$ defined as follows:*

$$\overline{\epsilon}_k = \frac{1}{qn} \min \left\{ L_k^p - \sum_{\nu \in \Omega_k^u} l^p(\nu) : \ \Omega_k^u \in \Omega_k \backslash \Omega_k(p) \right\}. \tag{2.56}$$

The following claim is a generalization of Theorem 1.6.

Theorem 2.8 *Let G_s be a digraph with the minimal value L_s^p, $p \in T$, among the digraphs $B \subseteq \Lambda(G)$. The equality $\overline{\varrho}_s^B(p \in T) = 0$ holds if and only if*

1) there exists a digraph $G_k \in B$ such that $L_s^p = L_k^p$, $k \neq s$,

2) the set $\Omega_s^{v^*} \in \Omega_{sk}^* \cap \Omega_s(p)$ is such that for any set $\Omega_k^{u^*} \in \Omega_k(p)$, there exists an operation $O_{ij} \in Q^J$ for which condition (2.57) (or condition (2.58)) holds:

$$n_{ij}(\Omega_s^{v^*}) \geq n_{ij}(\Omega_k^{u^*}), \tag{2.57}$$

$$n_{ij}(\Omega_s^{v^*}) \leq n_{ij}(\Omega_k^{u^*}), \tag{2.58}$$

3) inequality (2.57) (inequality (2.58)) is satisfied as a strict one for at least one set $\Omega_k^{u^0} \in \Omega_k(p)$.

Proof. *Necessity.* We prove necessity by contradiction. Assume $\overline{\varrho}_s^B(p \in T) = 0$ but the conditions of the theorem are not satisfied. We consider four cases *i*), *ii*), *iii*) and *iv*) of violating these conditions.

i) Assume that there does not exist another optimal digraph $G_k \in B$ such that $L_s^p = L_k^p, k \neq s$. If $B \setminus \{G_s\} \neq \emptyset$, we can calculate the value

$$\overline{\epsilon}^* = \frac{1}{qn} \min_{t \neq s} (L_t^p - L_s^p), \tag{2.59}$$

which is strictly positive since $L_s^p < L_t^p$ for each $G_t \in B$, $t \neq s$. Using Lemma 2.7, one can verify that for any real ϵ, which satisfies the inequalities $0 < \epsilon < \overline{\epsilon}^*$, the difference in the right-hand side of equality (2.59) remains positive when vector p is replaced by any vector $p^0 \in O_\epsilon(p) \cap T$. Indeed, for any $v \in \{1, 2, \ldots, \omega_s^T\}$, the cardinality of set Ω_s^v may be at most equal to $q \cdot n$: $|\Omega_s^v| \leq q \cdot n$. Thus, the difference

$$L_t^p - L_s^p = L_t^p - \max_{v \in \{1,2,\ldots,\omega_s^T\}} \sum_{\nu \in \Omega_s^v} l^p(\nu)$$

may not be 'overcome' using a vector p^0 if $d(p, p^0) < \overline{\epsilon}^*$. Hence, we conclude that digraph G_s remains optimal for any vector $p^0 = (p_{1,1}^0, p_{1,2}^0, \ldots, p_{nn_n}^0) \in T$

of the processing times provided that $d(p,p^0) \le \epsilon < \overline{\epsilon}^*$. Therefore, we have $\overline{\varrho}_s^B(p \in T) \ge \overline{\epsilon}^* > \epsilon > 0$ which contradicts the assumption $\overline{\varrho}_s^B(p \in T) = 0$.

ii) Assume that there exists a digraph $G_k \in B$ such that $L_s^p = L_k^p, k \ne s$, and $\Omega_{sk}^* \cap \Omega_s(p) = \emptyset$. Note that $\Omega_{st}^* \ne \emptyset$ for all digraphs $G_t \in B, t \ne s$. Otherwise, we get $\overline{\varrho}_s^B(p \in T) = \infty$ due to Theorem 2.7.

Assume that there exists a set $\Omega_{st}^* \ne \emptyset$ for the digraphs G_s and G_t with $L_t^p > L_s^p$, i.e., there exists a set $\Omega_s^{v'} \in \Omega_{st}^*$. Similarly to the proof of Corollary 2.7, we can show that all values $r_{\Omega_s^{v'},\Omega_t^u}$ calculated for each digraph G_t with $L_t^p > L_s^p$ cannot be equal to zero. We obtain a strongly positive numerator in formula (2.48) at least for $\beta = 0$:

$$\sum_{\nu \in \Omega_t^u} l^p(\nu) - \sum_{\mu \in \Omega_s^{v'}} l^p(\mu) > 0.$$

Therefore, the maximum taken according to (2.48) is also strongly positive, i.e., $r_{\Omega_s^{v'},\Omega_t^u} > \epsilon > 0$, where we can choose any ϵ such that inequality

$$\epsilon < \min \left\{ \overline{\epsilon}_s,\, \overline{\epsilon}_k,\, \frac{1}{qn} \min_{\substack{G_t \in B,\\ L_t^p > L_s^p}} (L_t^p - L_s^p) \right\} \tag{2.60}$$

is satisfied. This means that only in the case of the calculation of the value $r_{\Omega_s^v,\Omega_k^u}$ for the optimal digraphs $G_k \in B, L_k^p = L_s^p$, with $\Omega_{sk}^* \ne \emptyset$, we can obtain $r_{\Omega_s^v,\Omega_k^u} = 0$.

Assume that there exists a set $\Omega_{sk}^* \ne \emptyset$ for the digraphs G_s and G_k with $L_k^p = L_s^p$, i.e., there exists a set $\Omega_s^{v''} \in \Omega_{sk}^*$. In this case, we can set

$$\epsilon' = \min \left\{ \overline{\epsilon}_s, \overline{\epsilon}_k, \frac{1}{qn} \min \left\{ L_s^p - \max \sum_{\mu \in \Omega_s^v} l^p(\mu) : \sum_{\nu \in \Omega_s^v} l^p(\mu) < L_s^p \right\} \right\}.$$

Taking into account our assumption that for each digraph $G_k \in B, L_s^p = L_k^p, k \ne s$, set $\Omega_{sk}^* \cap \Omega_s(p)$ is empty, it follows from the proof of Corollary 2.7 that $r_{\Omega_s^{v''},\Omega_k^u} > \epsilon' > 0$. Hence, for all digraphs $G_t, L_t^p \ge L_s^p$, inequality $\overline{r}_{ts}^B > \min\{\epsilon, \epsilon'\}$ holds, where the value $\overline{r}_{ts}^B$ is calculated due to formula (2.54) using the value $r_{\Omega_s^v,\Omega_t^u} > 0$. Therefore, the relative stability radius satisfies the inequalities $\overline{\varrho}_s^B(p \in T) > \min\{\epsilon, \epsilon'\} > 0$, which contradicts the assumption $\overline{\varrho}_s^B(p \in T) = 0$.

iii) Assume that there exists a digraph $G_k \in B$ such that $L_s^p = L_k^p, k \ne s$, and for any set $\Omega_s^{v^*} \in \Omega_{sk}^* \cap \Omega_s(p)$, there exists a set $\Omega_k^{u^*} \in \Omega_k(p)$ such that $n_{ij}(\Omega_s^{v^*}) = n_{ij}(\Omega_k^{u^*})$ for any operation $O_{ij} \in Q^J$.

In this case, we can take any ϵ that satisfies inequality (2.60). Due to $\epsilon < \overline{\epsilon}_s$, we get from Lemma 2.7 that equality

$$L_s^{p^0} = \max_{\Omega_s^{v^*} \in \Omega_s(p^0)} \sum_{\mu \in \Omega_s^{v^*}} l^{p^0}(\mu) = \max_{\Omega_s^{v^*} \in \Omega_s(p)} \sum_{\mu \in \Omega_s^{v^*}} l^{p^0}(\mu) \tag{2.61}$$

holds for any vector $p^0 \in O_\epsilon(p) \cap T$. On the other hand, since there exists a set $\Omega_k^{u^*} \in \Omega_k(p)$ such that $n_{ij}(\Omega_s^{v^*}) = n_{ij}(\Omega_k^{u^*})$, $O_{ij} \in Q$, for any set $\Omega_s^{v^*} \in \Omega_{sk}^* \cap \Omega_s(p)$ and for any digraph G_k, $L_s^p = L_k^p$, we obtain inequality

$$\max_{\Omega_s^{v^*} \in \Omega_s(p)} \sum_{\mu \in \Omega_s^{v^*}} l^{p^0}(\mu) \le \max_{\Omega_k^u \in \Omega_k(p)} \sum_{\nu \in \Omega_k^{u^*}} l^{p^0}(\nu),$$

because of $\epsilon < \overline{\epsilon}_s$ and $\epsilon < \overline{\epsilon}_k$. Therefore, due to (2.61), we have

$$L_s^{p^0} \le \max_{\Omega_k^u \in \Omega_k(p)} \sum_{\nu \in \Omega_k^{u^*}} l^{p^0}(\nu) \tag{2.62}$$

for any optimal digraph G_k, $k \neq s$. Since

$$\epsilon < \frac{1}{qn} \min_{L_t^p \neq L_s^p} \{L_t^p - L_s^p\},$$

the condition $L_t^p \neq L_s^p$ implies $L_t^{p^0} \neq L_s^{p^0}$. So taking into account (2.60) and the latter implication, we conclude that digraph G_s becomes an optimal digraph for any vector $p^0 \in T$, provided that $d(p, p^0) \le \epsilon$. Consequently, we have $\overline{\varrho}_s^B(p \in T) \ge \epsilon > 0$, which contradicts the assumption $\overline{\varrho}_s^B(p \in T) = 0$.

iv) Assume that conditions 1 and 2 of Theorem 2.8 hold. More exactly, there exists a digraph $G_k \in B$ such that $L_s^p = L_k^p, k \neq s$, and one of the two cases of condition 2 and one of the two cases of condition 3 hold. Assume that for any set $\Omega_s^{v^*} \in \Omega_{sk}^* \cap \Omega_s(p)$, there exists a set $\Omega_k^{u^*} \in \Omega_k(p)$ such that for any operation $O_{ij} \in Q^J$ with $n_{ij}(\Omega_s^{v^*}) > n_{ij}(\Omega_k^{u^*})$, there exists a set $\Omega_k^{u^0} \in \Omega_k(p)$ with $n_{ij}(\Omega_s^{v^*}) < n_{ij}(\Omega_k^{u^0})$.

Arguing in the same way as in case *iii*), we can show that $\overline{\varrho}_s^B(p \in T) \ge \epsilon > 0$, where ϵ is as in (2.60), since for any vector $p^0 \in O_\epsilon(p) \cap T$, the value $\sum_{\mu \in \Omega_s^{v^*}} l^{p^0}(\mu)$ is less than or equal to the value $\sum_{\nu \in \Omega_k^{u^*}} l^{p^0}(\nu)$ or $\sum_{\nu \in \Omega_k^{u^0}} l^{p^0}(\nu)$.

Sufficiency. We show that, if the conditions of Theorem 2.8 are satisfied, then $\overline{\varrho}_s^B(p \in T) < \epsilon$ for any given $\epsilon > 0$. First, we give the following remark.

Remark 2.8 In the case of $a_{ij} = b_{ij}$ for each operation $O_{ij} \in Q^J$, set $\Omega_{sk}^* \cap \Omega_s(p)$ is empty since vector p becomes equal to vector p^* constructed according to (2.52), and inequality (2.53) does not hold.

We construct a vector $p' = (p'_{1,1}, p'_{1,2}, \ldots, p'_{nn_n}) \in T$ with the components $p'_{ij} \in \{p_{ij}, p_{ij} + \epsilon', p_{ij} - \epsilon'\}$, where $\epsilon' = \min\{\epsilon, \overline{\epsilon}_k, \overline{\epsilon}_{min}\}$ with the value $\overline{\epsilon}_k > 0$ defined in (2.56), and $\overline{\epsilon}_{min} = \max\{0,\ \min\{\min\{p_{ij} - a_{ij} :\ p_{ij} > a_{ij},\ O_{ij} \in Q^J\}, \min\{b_{ij} - p_{ij} :\ b_{ij} > p_{ij},\ O_{ij} \in Q^J\}\}\}$ using the following rule. For each $\Omega_k^{u^*} \in \Omega_k(p)$, mentioned in Theorem 2.8, we set either $p'_{ij} = p_{ij} + \epsilon'$ if inequalities (2.57) hold, or $p'_{ij} = p_{ij} - \epsilon'$ if inequalities (2.58) hold.

More precisely, we can choose ϵ' as follows: If $\Omega_k \neq \Omega_k(p)$, then $\overline{\epsilon}_k > 0$, and we can choose ϵ' such that $0 < \epsilon' < \min\{\epsilon,\ \overline{\epsilon}_k,\ \overline{\epsilon}_{min}\}$. Otherwise, if $\Omega_k = \Omega_k(p)$, we choose ϵ' such that $0 < \epsilon' < \min\{\overline{\epsilon},\ \overline{\epsilon}_{min}\}$. Such choices are possible since in both cases, inequality $\overline{\epsilon}_{min} > 0$ holds due to Remark 2.8. Note that $\epsilon' > 0$ since $p_{ij} > 0, O_{ij} \in Q^J$. The following arguments are the same for both cases of the choice of ϵ'.

After changing at most $|\Omega_k(p)|$ components of vector p according to this rule, we obtain a vector p' of the processing times for which inequality

$$\sum_{\mu \in \Omega_s^{v^*}} l^{p'}(\mu) > \sum_{\nu \in \Omega_k^{u^*}} l^{p'}(\nu)$$

holds for each set $\Omega_k^{u^*} \in \Omega_k(p)$. Due to $\epsilon' \leq \overline{\epsilon}_{min}$, we have $p' \in T$. Since $\epsilon' \leq \overline{\epsilon}_k$, we have

$$\begin{aligned} L_k^{p'} &= \max_{u \in \{1,2,\ldots,\omega_k^T\}} \sum_{\nu \in \Omega_k^u} l^{p'}(\nu) = \max_{\Omega_k^u \in \Omega_k(p)} \sum_{\nu \in \Omega_k^u} l^{p'}(\nu) \\ &= \sum_{\nu \in \Omega_k^{u^*}} l^{p'}(\nu) < \sum_{\mu \in \Omega_s^{v^*}} l^{p'}(\mu) \leq L_s^{p'}. \end{aligned}$$

Thus, we conclude that digraph G_s is not optimal for vector $p' \in T$ with $d(p, p') = \epsilon'$ which implies $\overline{\varrho}_s^B(p \in T) < \epsilon' \leq \epsilon$.

◇

Theorem 2.8 directly implies the following assertion.

Corollary 2.10 *If $G_s \in B$ is the unique optimal digraph for vector $p \in T$, then $\overline{\varrho}_s^B(p \in T) > 0$.*

From Theorem 2.8, we obtain the following lower bound for the relative stability radius $\overline{\varrho}_s^B(p \in T)$.

Corollary 2.11 *If $G_s \in B$ is an optimal digraph, then $\overline{\varrho}_s^B(p \in T) \geq \overline{\epsilon}^*$, where $\overline{\epsilon}^*$ is calculated according to (2.59).*

Proof. If there exists a digraph $G_k \in B$ with $L_s^p = L_k^p,\ k \neq s$, relation $\overline{\varrho}_s^B(p \in T) \geq \overline{\epsilon}^* = 0$ holds due to Definition 2.6. Otherwise, inequality $\overline{\varrho}_s^B(p \in T) \geq \overline{\epsilon}^*$ follows from the above proof of necessity (case *i*)).

◇

Example 2.2 (continued). *Returning to Example 2.2 and using Theorem 2.6, we can calculate the relative stability radius of digraph $G_1 \in B \subseteq \Lambda(G), |B| = 12$, for vector $p = p^0 = (70, 30, 60, 20, 60, 70, 40, 30)$ of the*

processing times according to formula (2.55). After a pairwise comparison of the sets of representatives for digraph G_1^T with those for the digraphs $G_2^T, G_3^T, \ldots, G_{12}^T$, we obtain equality $\overline{\varrho}_1^B(p^0 \in T) = 3$, which means that digraph G_1 remains optimal at least for all vectors $p \in O_3(p^0) \cap T$ of the processing times. Due to the calculation of the relative stability radius, we show that only digraphs G_2 and G_5 may be better than digraph G_1 provided that vector p of the processing times belongs to polytope T, and for each digraph $G_k \in \Lambda(G)$ with $k \neq 2$ and $k \neq 5$, dominance relation $G_1 \preceq_T G_k$ holds. We also obtain the following equalities: $\overline{\varrho}_1^B(p^0 \in T) = \overline{r}_{2,1}^B = 3, \overline{\varrho}_1^{B\setminus\{G_2\}}(p^0 \in T) = \overline{r}_{5,1}^{B\setminus\{G_2\}} = 10$, where the values $\overline{r}_{k1}^B$ are calculated according to (2.54), and they give the minimum over all digraphs $G_k \in B$ which are not dominated by digraph G_1. It follows from Theorem 2.7 that $\overline{\varrho}_1^{B\setminus\{G_2,G_5\}}(p^0 \in T) = \infty$.

Due to Theorem 2.4, set $\Lambda^*(G) = \{G_1, G_2, G_5\}$ is a G-solution to problem $\mathcal{J}3/n = 3, a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$, since there exists a covering of polytope T by the sets $D_s = \{p \in R_+^8 : L_s^p \leq L_k^p, k = 1, 2 \ldots, \lambda\} \cap T$ with $s \in \{1, 2, 5\}$. More exactly, for any digraph $G_k \in \Lambda(G)$ and for any set D_s, $s \in \{1, 2, 5\}$, there exists a digraph $G_s \in \Lambda^*(G)$ for which dominance relation $G_s \preceq_{D_s} G_k$ holds (since dominance relation $G_1 \preceq_T G_k$ holds for each digraph $G_k \in \Lambda(G), k \neq 2, k \neq 5$, it follows that set $\{D_1, D_2, D_5\}$ is indeed a covering of polytope T). Moreover, since for each digraph $G_s \in \Lambda^*(G)$, there exists a point (see vectors p^0, $\overline{p}^\epsilon$ and $\overline{x}$, given in Section 2.4), for which this digraph is the unique optimal one, it follows from Theorem 2.5 that the G-solution $\Lambda^*(G) = \{G_1, G_2, G_5\}$ is minimal.

Note that from a practical point of view, it is more useful to consider a covering of polytope T by nested balls $O_3(p^0), O_{10}(p^0)$ and $O_{r^*}(p^0)$, where r^* may be any real number no less than $\max\{b_{ij} - p_{ij}^0, p_{ij}^0 - a_{ij} : \ i = 1, 2, \ldots, n; \ j = 1, 2, \ldots, n_i\}$. Indeed, due to the calculation of the stability radius $\overline{\varrho}_1^B(p^0 \in T)$, we know that for each vector $p \in O_3(p^0)$, digraph G_1 is optimal. Moreover, for each vector $p \in O_{10}(p^0)$, at least one digraph G_1 or G_2 is optimal since $\overline{\varrho}_1^{B\setminus\{G_2\}}(p^0 \in T) = 10$. Finally, for each vector $p \in O_{r^*}(p^0)$, at least one digraph G_1, G_2 or G_5 is optimal since $\overline{\varrho}_1^{B\setminus\{G_2,G_5\}}(p^0 \in T) = \infty$.

Remark 2.9 The solution of problem $\mathcal{J}3/n = 3, a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$ takes three iterations by the above algorithm (see Table 2.8). However, similarly to the calculation of the relative stability radius and the construction of a G-solution of the scheduling problem with the makespan criterion (see Remark 2.4), we can construct a G-solution $\Lambda^*(G)$ for the mean flow time criterion in one scan as follows. We union one of the optimal digraphs G_s

with all digraphs $G_k, k \neq s$, for which a non-empty set $\Omega^*_{sk} \neq \emptyset$ exists, i.e., for which dominance relation $G_s \preceq_T G_k$ does not hold, and the union of these digraphs composes such a G-solution $\Lambda^*(G)$. In other words, a G-solution of problem $\mathcal{J}/a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$ is the union of an optimal digraph and of all its competitive digraphs $\Lambda^*(G) = \{G_s\} \cup \{G_k : G_s \not\preceq_T G_k\} = \{G_s\} \cup \{\cup_{i=1}^{I} \Gamma_i\}$, where Γ_i is the set of competitive digraphs of digraph G_s with respect to set B in iteration $i = 1, 2, \ldots, I$.

Table 2.8: *G-solution of problem $\mathcal{J}3/n=3, a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$ with the initial vector $p^0 \in T$*

i	Set B	$\overline{\varrho}_1^B(p^0 \in T)$	Set Γ_i of competitive digraphs of digraph G_1
1	$B = \{G_1, G_2, \ldots, G_{12}\}$	3	$\{G_2\}$
2	$B \setminus \{G_2\}$	10	$\{G_5\}$
3	$B \setminus \{G_2, G_5\}$	∞	$\emptyset$

Next, we consider a small problem $\mathcal{J}3/n=2/\sum \mathcal{C}_i$ to illustrate the calculation of $\overline{\varrho}_1(p)$ by formulas (1.39) and (1.40). Then we calculate the relative stability radius for problem $\mathcal{J}3/n = 2, a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$. Without causing any confusion, we use the same notations for different examples, i.e., for Example 2.1 and Example 2.2.

Example 2.1 (continued). *Returning to Example 2.1 from Section 2.1, we consider the job shop problem with the mean flow time criterion $\mathcal{J}3/n = 2/\sum \mathcal{C}_i$, whose input data are given by the weighted mixed graph $G(p)$ with $p = (75, 50, 40, 60, 55, 30)$, presented in Figure 2.1. Obviously, the set of all feasible digraphs $\Lambda(G)$ is the same, but we number these digraphs in non-decreasing order of the objective function values: $L_1^p \leq L_2^p \leq \ldots \leq L_5^p$ (see Figure 2.8). Note that for criterion $\sum \mathcal{C}_i$, we do not need to use dummy operations. As we can see, digraph $G_1(p)$ is optimal for both criteria $\mathcal{C}_{max}$ and $\sum \mathcal{C}_i$. Next, we calculate the stability radius $\overline{\varrho}_1(p)$ for digraph $G_1(p)$.*

To this end, we construct an auxiliary Table 2.9, where for each feasible digraph $G_k, k \in \{1, 2, 3, 4, 5\}$, column 2 presents the sets Ω_k^u of representatives of the family of sets $(H_k^i)_{J_i \in J}$, column 3 presents the integer vector $n(\Omega_k^u) = (n_{1,1}(\Omega_k^u), n_{1,2}(\Omega_k^u), \ldots, n_{2,3}(\Omega_k^u))$, where the value $n_{ij}(\Omega_k^u)$ is equal to the number of vertices O_{ij} in the multiset $\{[\nu] : \nu \in \Omega_k^u\}$ (for simplicity, we use the notation n_{ij} instead of $n_{ij}(\Omega_k^u)$), and column 4 presents the value

$$\sum_{\nu \in \Omega_k^u} l^p(\nu) = \sum_{O_{ij} \in [\nu],\ \nu \in \Omega_k^u} p_{ij} \cdot n_{ij}(\Omega_k^u).$$

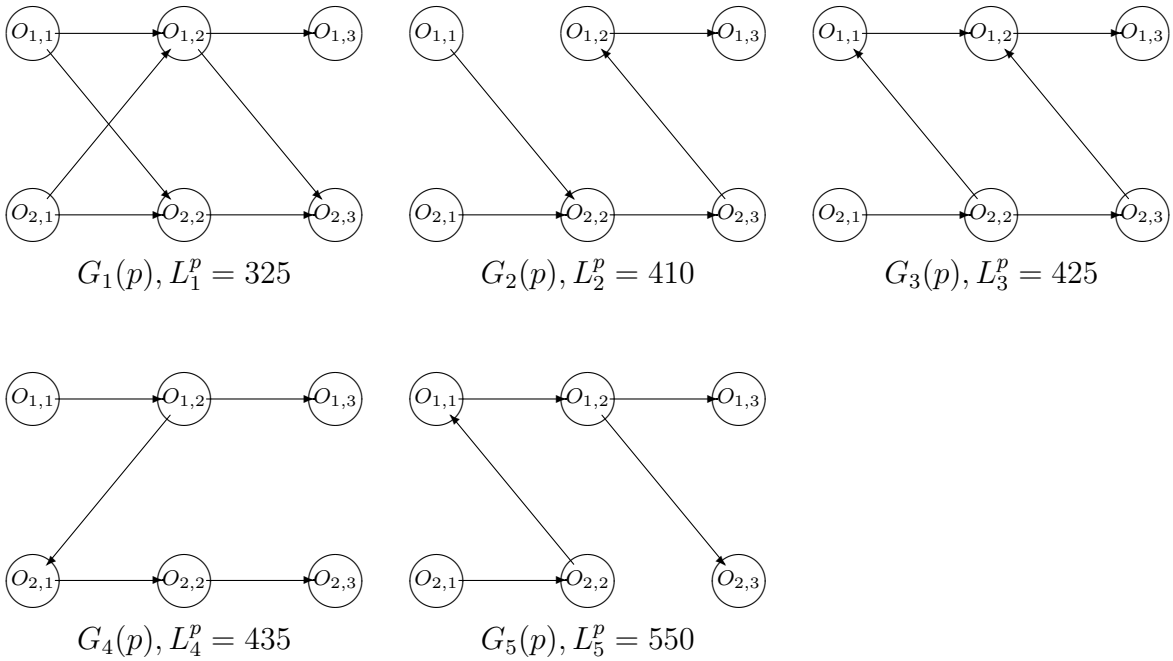


Figure 2.8: *Digraphs $\Lambda(G) = \{G_1, G_2, \ldots, G_5\}$ numbered in non-decreasing order of the objective function values $\sum \mathcal{C}_i$*

Table 2.9: *Auxiliary information for problem $\mathcal{J}3/n=2/\sum \mathcal{C}_i$*

G_k	$\Omega_k^u, u = 1, 2, \ldots, \omega_k$	$n_{1,1}$	$n_{1,2}$	$n_{1,3}$	$n_{2,1}$	$n_{2,2}$	$n_{2,3}$	$\sum_{\nu \in \Omega_k^u} l^p(\nu)$
1	2	3						4
G_1	$\Omega_1^1 = \{O_{1,1}, O_{1,2}, O_{1,3}; O_{1,1}, O_{1,2}, O_{2,3}\}$	2	2	1	0	0	1	320
	$\Omega_1^2 = \{O_{1,1}, O_{1,2}, O_{1,3}; O_{2,1}, O_{1,2}, O_{2,3}\}$	1	2	1	1	0	1	305
	$\Omega_1^3 = \{O_{1,1}, O_{1,2}, O_{1,3}; O_{1,1}, O_{2,2}, O_{2,3}\}$	2	1	1	0	1	1	325
	$\Omega_1^4 = \{O_{1,1}, O_{1,2}, O_{1,3}; O_{2,1}, O_{2,2}, O_{2,3}\}$	1	1	1	1	1	1	310
	$\Omega_1^5 = \{O_{2,1}, O_{1,2}, O_{1,3}; O_{1,1}, O_{1,2}, O_{2,3}\}$	1	2	1	1	0	1	305
	$\Omega_1^6 = \{O_{2,1}, O_{1,2}, O_{1,3}; O_{2,1}, O_{1,2}, O_{2,3}\}$	0	2	1	2	0	1	290
	$\Omega_1^7 = \{O_{2,1}, O_{1,2}, O_{1,3}; O_{1,1}, O_{2,2}, O_{2,3}\}$	1	1	1	1	1	1	310
	$\Omega_1^8 = \{O_{2,1}, O_{1,2}, O_{1,3}; O_{2,1}, O_{2,2}, O_{2,3}\}$	0	1	1	2	1	1	295
G_2	$\Omega_2^1 = \{O_{1,1}, O_{2,2}, O_{2,3}, O_{1,2}, O_{1,3}; O_{1,1}, O_{2,2}, O_{2,3}\}$	2	1	1	0	2	2	410
	$\Omega_2^2 = \{O_{1,1}, O_{2,2}, O_{2,3}, O_{1,2}, O_{1,3}; O_{2,1}, O_{2,2}, O_{2,3}\}$	1	1	1	1	2	2	395
	$\Omega_2^3 = \{O_{2,1}, O_{2,2}, O_{2,3}, O_{1,2}, O_{1,3}; O_{1,1}, O_{2,2}, O_{2,3}\}$	1	1	1	1	2	2	395
	$\Omega_2^4 = \{O_{2,1}, O_{2,2}, O_{2,3}, O_{1,2}, O_{1,3}; O_{2,1}, O_{2,2}, O_{2,3}\}$	0	1	1	2	2	2	380
G_3	$\Omega_3^1 = \{O_{2,1}, O_{2,2}, O_{1,1}, O_{1,2}, O_{1,3}; O_{2,1}, O_{2,2}, O_{2,3}\}$	1	1	1	2	2	1	425
	$\Omega_3^2 = \{O_{2,1}, O_{2,2}, O_{2,3}, O_{1,2}, O_{1,3}; O_{2,1}, O_{2,2}, O_{2,3}\}$	0	1	1	2	2	2	380
G_4	$\Omega_4^1 = \{O_{1,1}, O_{1,2}, O_{1,3}; O_{1,1}, O_{1,2}, O_{2,1}, O_{2,2}, O_{2,3}\}$	2	2	1	1	1	1	435
G_5	$\Omega_5^1 = \{O_{2,1}, O_{2,2}, O_{1,1}, O_{1,2}, O_{1,3}; O_{2,1}, O_{2,2}, O_{1,1}, O_{1,2}, O_{2,3}\}$	2	2	1	2	2	1	550

The calculation of $\overline{\varrho}_1(p)$ by formula (1.40) is given in Table 2.10, which presents the results of the computations for each $\beta = 1, 2, \ldots, q - m$, where

Table 2.10: *Calculation of the stability radius $\overline{\varrho}_1(p)$ for problem $\mathcal{J}3/n=2/\sum \mathcal{C}_i$*

G_k	$\lvert\Omega_{1k}\rvert$	$\Omega_1^v \in \Omega_{1k}$	Ω_k^u, $1 \le u \le \omega_k$	β	$p_{ij_{(m+\beta)}}$, $1 \le \beta \le q-m$	$\frac{\sum_{\alpha=1}^{m+\beta} p_{ij_{(\alpha)}}(n_{ij_{(\alpha)}}(\Omega_k^u) - n_{ij_{(\alpha)}}(\Omega_1^v))}{\sum_{\alpha=1}^{m+\beta} \lvert n_{ij_{(\alpha)}}(\Omega_k^u) - n_{ij_{(\alpha)}}(\Omega_1^v)\rvert}$	$\max\limits_{\beta}$	$\max\limits_{\Omega_k^u}$	$\min\limits_{\Omega_1^v}$
1	2	3	4	5	6	7	8	9	10
G_2	4	Ω_1^1	Ω_2^1	1	$p_{ij_{(5)}} = 55$	$\frac{50(1-2)+55(2-0)}{\lvert 1-2\rvert+\lvert 2-0\rvert} = 20$	22.5	22.5	22.5
				2	$p_{ij_{(6)}} = 30$	$\frac{50(1-2)+55(2-0)+30(2-1)}{\lvert 1-2\rvert+\lvert 2-0\rvert+\lvert 2-1\rvert} = 22.5$			
			Ω_2^2, Ω_2^3	1	$p_{ij_{(4)}} = 60$	$\frac{75(1-2)+50(1-2)+60(1-0)}{1+1+1} = \frac{-65}{3}$	12.5		
				2	$p_{ij_{(5)}} = 55$	$\frac{-65+55(2-0)}{3+2} = \frac{45}{5} = 9$			
				3	$p_{ij_{(6)}} = 30$	$\frac{45+30(2-1)}{5+1} = 12.5$			
			Ω_2^4	1	$p_{ij_{(4)}} = 60$	$\frac{75(0-2)+50(1-2)+60(2-0)}{2+1+2} = \frac{-80}{5}$	7.5		
				2	$p_{ij_{(5)}} = 60$	$\frac{-80+55(2-0)}{5+2} = \frac{30}{7} = 4\frac{2}{3}$			
				3	$p_{ij_{(6)}} = 30$	$\frac{30+30(2-1)}{7+1} = 7.5$			
		Ω_1^2, Ω_1^5	Ω_2^1	1	$p_{ij_{(4)}} = 75$	$\frac{50(1-2)+60(0-1)+75(2-1)}{1+1+1} = \frac{-35}{3}$	17.5	22.5	
				2	$p_{ij_{(5)}} = 55$	$\frac{-35+55(2-0)}{3+2} = \frac{75}{5} = 15$			
				3	$p_{ij_{(6)}} = 30$	$\frac{75+30(2-1)}{5+1} = 17.5$			
			Ω_2^2, Ω_2^3	1	$p_{ij_{(5)}} = 55$	$\frac{50(1-2)+55(2-0)}{1+2} = \frac{60}{3} = 20$	22.5		
				2	$p_{ij_{(6)}} = 30$	$\frac{60+30(2-1)}{3+1} = 22.5$			
			Ω_2^4	1	$p_{ij_{(4)}} = 60$	$\frac{75(0-1)+50(1-2)+60(2-1)}{1+1+1} = \frac{-65}{3}$	12.5		
				2	$p_{ij_{(5)}} = 55$	$\frac{-65+55(2-0)}{3+2} = \frac{45}{5} = 9$			
				3	$p_{ij_{(6)}} = 30$	$\frac{45+30(2-1)}{5+1} = 12.5$			
		Ω_1^6	Ω_2^1	1	$p_{ij_{(4)}} = 75$	$\frac{50(1-2)+60(0-2)+75(2-0)}{1+2+2} = \frac{-20}{5}$	15	22.5	
				2	$p_{ij_{(5)}} = 55$	$\frac{-20+55(2-0)}{5+2} = \frac{90}{7} = 12\frac{6}{7}$			
				3	$p_{ij_{(6)}} = 30$	$\frac{90+30(2-1)}{7+1} = 15$			
			Ω_2^2, Ω_2^3	1	$p_{ij_{(4)}} = 75$	$\frac{50(1-2)+60(1-2)+75(1-0)}{1+1+1} = \frac{-35}{3}$	17.5		
				2	$p_{ij_{(5)}} = 55$	$\frac{-35+55(2-0)}{3+2} = \frac{75}{5} = 15$			
				3	$p_{ij_{(6)}} = 30$	$\frac{75+30(2-1)}{5+1} = 17.5$			
			Ω_2^4	1	$p_{ij_{(5)}} = 55$	$\frac{50(1-2)+55(2-0)}{1+2} = \frac{60}{3} = 20$	22.5		
				2	$p_{ij_{(6)}} = 30$	$\frac{60+30(2-1)}{3+1} = 22.5$			
G_3	5	Ω_1^1	Ω_3^1	1	$p_{ij_{(5)}} = 60$	$\frac{75(1-2)+50(1-2)+60(2-0)}{1+1+2} = \frac{-5}{4}$	17.5	17.5	17.5
				2	$p_{ij_{(6)}} = 55$	$\frac{-5+55(2-0)}{4+2} = 17.5$			
			Ω_3^2	1	$p_{ij_{(4)}} = 60$	$\frac{75(0-2)+50(1-2)+60(2-0)}{2+1+2} = \frac{-80}{5}$	7.5		
				2	$p_{ij_{(5)}} = 55$	$\frac{-80+55(2-0)}{5+2} = \frac{30}{7} = 4\frac{2}{7}$			
				3	$p_{ij_{(6)}} = 30$	$\frac{30+30(2-1)}{7+1} = 7.5$			
		Ω_1^2, Ω_1^5	Ω_3^1	1	$p_{ij_{(5)}} = 60$	$\frac{50(1-2)+60(2-1)}{1+1} = \frac{10}{2} = 5$	30	30	
				2	$p_{ij_{(6)}} = 55$	$\frac{10+55(2-0)}{2+2} = 30$			
			Ω_3^2	1	$p_{ij_{(4)}} = 60$	$\frac{75(0-1)+50(1-2)+60(1+1-1)}{1+1+1} = \frac{-65}{3}$	12.5		
				2	$p_{ij_{(5)}} = 55$	$\frac{-65+55(2-0)}{3+2} = \frac{45}{5} = 9$			
				3	$p_{ij_{(6)}} = 30$	$\frac{45+30(2-1)}{5+1} = 12.5$			

Table 2.10 (continuation): *Calculation of the stability radius* $\overline{\varrho}_1(p)$ *for problem* $\mathcal{J}3/n{=}2/\sum \mathcal{C}_i$

G_k	$\lvert\Omega_{1k}\rvert$	$\Omega_1^v \in \Omega_{1k}$	Ω_k^u, $1 \le u \le \omega_k$	β	$p_{ij_{(m+\beta)}}$, $1 \le \beta \le q-m$	$\frac{\sum_{\alpha=1}^{m+\beta} p_{ij_{(\alpha)}}(n_{ij_{(\alpha)}}(\Omega_k^u) - n_{ij_{(\alpha)}}(\Omega_1^v))}{\sum_{\alpha=1}^{m+\beta} \lvert n_{ij_{(\alpha)}}(\Omega_k^u) - n_{ij_{(\alpha)}}(\Omega_1^v)\rvert}$	$\max_{\beta}$	$\max_{\Omega_k^u}$	$\min_{\Omega_1^v}$
1	2	3	4	5	6	7	8	9	10
		Ω_1^3	Ω_3^1	1	$p_{ij_{(5)}} = 60$	$\frac{75(1-2)+60(2-0)}{1+2} = \frac{45}{3} = 15$	25	25	
				2	$p_{ij_{(6)}} = 55$	$\frac{45+55(2-1)}{3+1} = 25$			
			Ω_3^2	1	$p_{ij_{(4)}} = 60$	$\frac{75(0-2)+60(2-0)}{2+2} = \frac{-30}{4}$	$9\frac{1}{6}$		
				2	$p_{ij_{(5)}} = 55$	$\frac{-30+55(2-1)}{4+1} = \frac{25}{5} = 5$			
				3	$p_{ij_{(6)}} = 30$	$\frac{25+30(2-1)}{5+1} = 9\frac{1}{6}$			
		Ω_1^6	Ω_3^1	1	$p_{ij_{(5)}} = 75$	$\frac{50(1-2)+75(1-0)}{1+1} = \frac{25}{2} = 12.5$	33.75	33.75	
				2	$p_{ij_{(6)}} = 55$	$\frac{25+55(2-0)}{2+2} = 33.75$			
			Ω_3^2	1	$p_{ij_{(5)}} = 55$	$\frac{50(1-2)+55(2-0)}{1+2} = \frac{60}{3} = 20$	22.5		
				2	$p_{ij_{(6)}} = 30$	$\frac{60+30(2-1)}{3+1} = 22.5$			
G_4	2	Ω_1^6	Ω_4^1	1	$p_{ij_{(5)}} = 75$	$\frac{60(1-2)+75(2-0)}{1+2} = \frac{90}{3} = 30$	36.25	36.25	35
				2	$p_{ij_{(6)}} = 55$	$\frac{90+55(1-0)}{3+1} = 36.25$			
		Ω_1^8	Ω_4^1	1	$p_{ij_{(5)}} = 75$	$\frac{60(1-2)+75(2-0)}{1+2} = \frac{90}{3} = 30$	35	35	
				2	$p_{ij_{(6)}} = 55$	$\frac{90+55(2-1)}{3+1} = 35$			
G_5	0								

m is the number of operations $O_{ij} \in \Omega_1^v \cup \Omega_k^u$, $\Omega_1^v \in \Omega_{1k}$, for which $n_{ij}(\Omega_1^v) < n_{ij}(\Omega_k^u)$. The cardinality of set $\Omega_{1k}, k \in \{1,2,3,4,5\}$, and the elements Ω_1^ν of this set are presented in column 2 and column 3, respectively. The elements of set $\Omega_k^u, u = 1,2,\ldots,\omega_k$, for which $\sum_{\nu\in\Omega_k^u} l^p(\nu) \ge L_1^p = 325$ are presented in column 4.

Since the vector $n(\Omega_k^u) = (n_{1,1}(\Omega_k^u), n_{1,2}(\Omega_k^u), \ldots, n_{2,3}(\Omega_k^u))$ is the same for both sets Ω_1^2 and Ω_1^5, for both sets Ω_1^4 and Ω_1^7, and for both sets Ω_2^2 and Ω_2^3 (see Table 2.9), the results calculated by formula (1.40) are the same for these pairs of sets, too. Therefore, we combine these calculations in column 7 in Table 2.10. In column 6 we give the sequence of the processing times of the operations $O_{ij} \in \Omega_1^v \cup \Omega_k^u$ with $n_{ij}(\Omega_1^v) < n_{ij}(\Omega_k^u)$ ordered in the following way: $p_{ij_{(m+1)}} \ge p_{ij_{(m+2)}} \ge \ldots \ge p_{ij_{(q)}}$. Note that in column 7, we do not write the components with $n_{ij}(\Omega_1^v) = n_{ij}(\Omega_k^u)$ in the fraction from formula (1.40). For the sets Ω_1^1 and Ω_2^1, we give a more detailed computation and for each other pair of sets Ω_1^v and Ω_k^u in each following iteration, we use the value of the fraction obtained in the previous iteration. From the derived values in column 7, we write their maximum for $\beta = 1,2,\ldots,q-m$, the maximum

for $\Omega_k^u, u = 1, 2, \ldots, \omega_k$, *and the minimum for* $\Omega_1^v \in \Omega_{1k}$, *respectively, in columns 8, 9 and 10. Using formula (1.39), we take the minimum value from column 10 and obtain* $\overline{\varrho}_1(p) = 17.5$.

We consider a job shop problem $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\sum \mathcal{C}_i$ *(Example 2.1) with the aim to illustrate the idea of constructing a G-solution to this problem mentioned in Remark 2.9. The structural input data are given by the mixed graph G in Figure 2.1. The numerical input data are given in Table 2.3. Obviously, the set of all feasible digraphs* $\Lambda(G)$ *is identical for both criteria* $\mathcal{C}_{max}$ *and* $\sum \mathcal{C}_i$, *and here we number these digraphs in non-decreasing order of the values* $\sum \mathcal{C}_i$ *with the same initial vector* $p = (75, 50, 40, 60, 55, 30)$ *as for problem* $\mathcal{J}3/n = 2, a_i \le p_i \le b_i/\mathcal{C}_{max}$ *considered in Chapter 2:* $L_1^p \le L_2^p \le \ldots \le L_5^p$ *(see Figure 2.8). Using the modification of the critical path method described on page 120, we can simplify digraphs* $G_1, G_2, \ldots, G_5$. *For these input data (see Table 2.3), the corresponding digraphs* $G_1^T, G_2^T, \ldots, G_5^T$ *remain the same. This means that the number of sets of representatives* ω_k^T *is equal to the number* ω_k *for each digraph* $G_k, k \in \{1, 2, 3, 4, 5\}$, *(Table 2.9).*

We calculate the relative stability radius $\overline{\varrho}_1^{\Lambda(G)}(p \in T)$ *for the optimal digraph* $G_1(p)$ *presented in Figure 2.8. First, due to Remark 2.7, we have to construct the set* Ω_{1k}^* *for each digraph* $G_k, k \in \{2, 3, 4, 5\}$. *To this end, we construct an auxiliary Table 2.11, where for each combination of the sets* $\Omega_1^v, v = 1, 2, \ldots, \omega_1^T$, *and* $\Omega_k^u, u = 1, 2, \ldots, \omega_k^T, k \in \{2, 3, 4, 5\}$, *we obtain the vector* p^* *according to formula (2.52) (see column 4) and check inequality (2.53) (see column 5). As we made in Table 2.10, we combine the same calculations for each pair of sets* Ω_1^2 *and* Ω_1^5, Ω_1^4 *and* Ω_1^7, Ω_2^2 *and* Ω_2^3. *Since* $\Omega_{sk}^* \subseteq \Omega_{sk}$, *we do not perform such a calculation for the sets* Ω_k^u, *which do not belong to the sets* $\Omega_{1k}, k \in \{2, 3, 4, 5\}$, *(see Table 2.10). So, it follows from column 5 that there is no set of representatives* Ω_1^v, $v \in \{1, 2, \ldots, \omega_1^T\}$, *such that inequality (2.53) holds for each set of representatives* Ω_k^u, $u \in \{1, 2, \ldots, \omega_k^T\}$.

Thus, $\Omega_{1k}^* = \emptyset$ *for each digraph* $G_k \in B = \Lambda(G) \setminus \{G_1\}$. *Therefore, from Theorem 2.6 and Theorem 2.7, it follows:* $\overline{\varrho}_1^{\Lambda(G)}(p \in T) = \infty$. *(For the numerical input data presented in Table 2.3, digraph* G_1 *dominates all digraphs* $G_k \in \Lambda(G)$ *in polytope T and remains the best for all feasible vectors* $x \in T$ *of the processing times.) In such a case, we obtain a single-element minimal G-solution* $\Lambda^T(G) = \{G_1\}$.

To illustrate the case of formula (2.55) from Theorem 2.6, we give the following example.

Example 2.3 *We consider a job shop problem* $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\sum \mathcal{C}_i$

Table 2.11: *Auxiliary information for the construction of the sets* $\Omega^*_{1k}, k \in \{2,3,4,5\}$, *for problem* $\mathcal{J}3/n=2, a_i \le p_i \le b_i / \sum \mathcal{C}_i$

G_k	Ω^v_1	Ω^u_k	$p^* \in T$	$\sum_{\mu\in\Omega^v_1} l^{p^*}(\mu) > \sum_{\mu\in\Omega^v_1} l^{p^*}(\mu)$	Ω^*_{1k}
1	2	3	4	5	6
G_2	Ω^1_1	Ω^1_2	$(75,90,40,60,45,20)$	$390 \not> 410$	$\Omega^1_1 \notin \Omega^*_{1,2}$
		Ω^2_2, Ω^3_2	$(100,90,40,50,45,20)$	$440 > 410$	
		Ω^4_2	$(100,90,40,50,45,20)$	$440 > 410$	
	Ω^2_1, Ω^5_1	Ω^1_2	$(35,90,40,80,45,20)$	$400 > 330$	$\{\Omega^2_1, \Omega^5_1\} \not\subseteq \Omega^*_{1,2}$
		Ω^2_2, Ω^3_2	$(75,90,40,60,45,20)$	$375 \not> 395$	
		Ω^4_2	$(100,90,40,50,45,20)$	$390 > 360$	
	Ω^3_1	Ω^1_2	$(75,50,40,60,45,20)$	$305 \not> 370$	$\Omega^3_1 \notin \Omega^*_{1,2}$
		Ω^2_2, Ω^3_2	$(100,50,40,50,45,20)$	$355 \not> 370$	
		Ω^4_2	$(100,50,40,50,45,20)$	$355 > 320$	
	$\{\Omega^4_1, \Omega^7_1\} \not\subseteq \Omega_{1,2}$				$\{\Omega^4_1, \Omega^7_1\} \not\subseteq \Omega^*_{1,2}$
	Ω^6_1	Ω^1_2	$(35,90,40,80,45,20)$	$400 > 330$	$\Omega^6_1 \notin \Omega^*_{1,2}$
		Ω^2_2, Ω^3_2	$(35,90,40,80,45,20)$	$400 > 375$	
		Ω^4_2	$(75,90,40,60,45,20)$	$360 \not> 380$	
	$\Omega^8_1 \notin \Omega_{1,2}$				$\Omega^8_1 \notin \Omega^*_{1,2}$
G_3	Ω^1_1	Ω^1_3	$(100,90,40,50,45,30)$	$450 \not> 450$	$\Omega^1_1 \notin \Omega^*_{1,3}$
		Ω^2_3	$(100,90,40,50,45,20)$	$440 > 360$	
	Ω^2_1, Ω^5_1	Ω^1_3	$(75,90,40,50,45,30)$	$375 \not> 425$	$\{\Omega^2_1, \Omega^5_1\} \not\subseteq \Omega^*_{1,3}$
		Ω^2_3	$(100,90,40,50,45,20)$	$390 > 360$	
	Ω^3_1	Ω^1_3	$(100,50,40,50,45,30)$	$365 \not> 410$	$\Omega^3_1 \notin \Omega^*_{1,3}$
		Ω^2_3	$(100,50,40,50,45,20)$	$355 > 320$	
	$\{\Omega^4_1, \Omega^7_1\} \not\subseteq \Omega_{1,3}$				$\{\Omega^4_1, \Omega^7_1\} \not\subseteq \Omega^*_{1,3}$
	Ω^6_1	Ω^1_3	$(35,90,40,60,45,30)$	$370 \not> 405$	$\Omega^6_1 \notin \Omega^*_{1,3}$
		Ω^2_3	$(75,90,40,60,45,20)$	$360 \not> 380$	
	$\Omega^8_1 \notin \Omega_{1,3}$				$\Omega^8_1 \notin \Omega^*_{1,3}$
G_4	$\Omega^1_1 \notin \Omega_{1,4}$				$\Omega^1_1 \notin \Omega^*_{1,4}$
	$\{\Omega^2_1, \Omega^5_1\} \not\subseteq \Omega_{1,4}$				$\{\Omega^2_1, \Omega^5_1\} \not\subseteq \Omega^*_{1,4}$
	$\Omega^3_1 \notin \Omega_{1,4}$				$\Omega^3_1 \notin \Omega^*_{1,4}$
	$\{\Omega^4_1, \Omega^7_1\} \not\subseteq \Omega_{1,4}$				$\{\Omega^4_1, \Omega^7_1\} \not\subseteq \Omega^*_{1,4}$
	Ω^6_1	Ω^1_4	$(35,50,40,80,45,30)$	$330 \not> 365$	$\Omega^6_1 \notin \Omega^*_{1,4}$
	Ω^8_1	Ω^1_4	$(35,40,40,80,55,30)$	$325 \not> 355$	$\Omega^8_1 \notin \Omega^*_{1,4}$
G_5	$\{\Omega_{1,5}\} = \emptyset$				$\{\Omega^*_{1,5}\} = \emptyset$

with the same structural input data (Figure 2.1), but with different numerical data (Table 2.12). We do not simplify the digraphs $G_1, G_2, \ldots, G_5$ *for the new numerical input data, i.e., the corresponding digraphs* $G^T_1, G^T_2, \ldots, G^T_5$ *have the same sets of representatives*

$$\{\Omega^u_k : u = 1,2,\ldots,\omega^T_k, \omega^T_k = \omega_k,\ k \in \{1,2,3,4,5\}\}.$$

For the initial vector $p = (75,50,40,60,55,30)$, *we have the same optimal digraph* $G_1(p)$ *and all feasible digraphs* $\Lambda(G)$ *are numbered as for the above problem (Figure 2.8). We calculate the relative stability radius* $\overline{\varrho}_1^{\Lambda(G)}(p \in T)$ *on the basis of Theorem 2.6. For the combination of the*

Table 2.12: *Numerical data for problem* $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\sum \mathcal{C}_i$

i	1	1	1	2	2	2
j	1	2	3	1	2	3
a_{ij}	35	30	40	45	10	15
b_{ij}	105	100	75	85	65	50

numerical input data given in Table 2.12, we construct the following sets: $\Omega^*_{1,2} = \{\Omega^1_1, \Omega^2_1, \Omega^5_1, \Omega^6_1\}$, $\Omega^*_{1,3} = \{\Omega^1_1, \Omega^2_1, \Omega^5_1, \Omega^6_1\}$, $\Omega^*_{1,4} = \{\Omega^6_1\}$, $\Omega^*_{1,5} = \emptyset$. *This means that digraph* G_1 *does not dominate digraphs* G_2, G_3, G_4 *in polytope* T *and due to Theorem 2.7, we have* $\overline{\varrho}_1^{\Lambda(G)}(p \in T) \neq \infty$. *In Table 2.13, one can observe the calculation of the relative stability radius* $\overline{\varrho}_1^{\Lambda(G)}(p \in T)$ *for the vector* $p = (75, 50, 40, 60, 55, 30)$. *Following Theorem 2.6, we must compare digraph* G_1 *with each digraph* $G_k, k \in \{2, 3, 4\}$, *for which* $\Omega^*_{1k} \neq \emptyset$. *Thus, we perform the calculations due to formulas (2.48) and (2.55) for each set* $\Omega^v_1 \in \Omega^*_{1k}$ *(see column 2) and each set* Ω^u_k *(see column 3). For the sets* Ω^u_k *in column 3, inequality* $\sum_{\nu \in \Omega^u_k} l^p(\nu) \ge L^p_1 = 325$ *holds (see Table 2.9). Column 5 contains the values* $\Delta^{ij}_\beta(\Omega^v_1, \Omega^u_k), \beta = 0, 1, \ldots, |N(\Omega^v_1, \Omega^u_k)| - 1$, *defined by formula (2.46) (see page 129) for each operation*

$$O_{ij} \in N(\Omega^u_k, \Omega^v_1) = \left\{ \bigcup_{\mu \in \Omega^u_k \cup \Omega^v_1} [\mu] : \ n_{ij}(\Omega^u_k) \neq n_{ij}(\Omega^v_1) \right\}.$$

The order of these values is defined by (2.47). The corresponding values $N_\beta(\Delta)$ *are given in column 6. Column 8 contains the value* $r_{\Omega^v_1, \Omega^u_k}$ *which is equal to the maximum of the values given in column 7 for* $\beta = 0, 1, \ldots, |N(\Omega^v_1, \Omega^u_k)| - 1$ *(see formula (2.48)). The values*

$$\overline{r}^B_{k1} = \min_{\Omega^v_1 \in \Omega^*_{1k}} \max_{\substack{u \in \{1,2,\ldots,\omega^T_k\}, \\ \sum_{\nu \in \Omega^u_k} l^p(\nu) \ge L^p_1}} r_{\Omega^v_1, \Omega^u_k}$$

calculated according to (2.54) are given in column 10.

As follows from Theorem 2.6, the last step is to take the minimum value in column 10: $\overline{\varrho}_1^{\Lambda(G)}(p \in T) = \min\{\overline{r}^B_{k1}, \ k \in \{2, 3, 4\}\} = \overline{r}_{3,1} = 18.75$.

As follows from Remark 2.9, we also construct an increasing sequence of relative stability radii $\overline{\varrho}_1^{\Lambda(G)}(p \in T) = \overline{r}_{3,1} = 18.75$, $\overline{\varrho}_1^{\Lambda(G)\setminus\{G_3\}}(p \in T) = \overline{r}_{2,1} = 25$, $\overline{\varrho}_1^{\Lambda(G)\setminus\{G_3,G_2\}}(p \in T) = \overline{r}_{4,1} = 40$, $\overline{\varrho}_1^{\Lambda(G)\setminus\{G_3,G_2,G_4\}}(p \in T) = \infty$ *(see Figure 2.9) and a sequence of nested sets of competitive digraphs* G_k *of digraph* G_1*:* $\Gamma_1 = \{G_3\}$, $\Gamma_2 = \{G_2\}$, $\Gamma_3 = \{G_4\}$, *for which dominance relation* $G_1 \preceq_T G_k$ *does not hold.*

Table 2.13: *Calculation of the relative stability radius* $\overline{\varrho}_1^{\Lambda(G)}(p \in T)$ *for problem* $\mathcal{J}3/n = 2, a_i \le p_i \le b_i / \sum \mathcal{C}_i$

G_k	$\Omega_1^v \in \Omega_{1k}^*$	Ω_k^u	β	$\Delta_\beta^{ij}(\Omega_1^v, \Omega_k^u)$	$N_\beta(\Delta)$	$\dfrac{\sum\limits_{\nu\in\Omega_k^u} l^p(\nu) - \sum\limits_{\mu\in\Omega_1^v} l^p(\mu) - \sum\limits_{\alpha=0}^{\beta} \Delta_\alpha^{ij}(\Omega_1^v, \Omega_k^u)\cdot N_\alpha(\Delta)}{\sum\limits_{O_{ij}\in Q^J} \lvert n_{ij}(\Omega_k^u) - n_{ij}(\Omega_1^v)\rvert - \sum\limits_{\alpha=0}^{\beta} \lvert N_\alpha(\Delta)\rvert}$	$\max\limits_{\beta}$	$\max\limits_{\Omega_k^u}$	$\min\limits_{\Omega_1^v}$
1	2	3	4	5	6	7	8	9	10
G_2	Ω_1^1	Ω_2^1	0	$\Delta_0^{ij}(\Omega_1^1, \Omega_2^1) = 0$	0	$\frac{410-320-0}{4-0} = \frac{90}{4} = 22.5$	25	25	25
			1	$\Delta_1^{2,3}(\Omega_1^1, \Omega_2^1) = 15$	1	$\frac{90-15}{4-1} = \frac{75}{3} = 25$			
			2	$\Delta_2^{2,2}(\Omega_1^1, \Omega_2^1) = 45$	2	$\frac{75-45\cdot 2}{3-2} < 0$			
		Ω_2^2, Ω_2^3	0	$\Delta_0^{ij}(\Omega_1^1, \Omega_2^2) = 0$	0	$\frac{395-320-0}{6-0} = \frac{75}{6} = 12.5$	12.5		
			1	$\Delta_1^{2,1}(\Omega_1^1, \Omega_2^2) = 15$	1	$\frac{75-15}{6-1} = \frac{60}{5} = 12$			
			2	$\Delta_2^{2,3}(\Omega_1^1, \Omega_2^2) = 15$	1	$\frac{60-15}{5-1} = \frac{45}{4} = 11.25$			
			3	$\Delta_3^{1,1}(\Omega_1^1, \Omega_2^2) = 30$	1	$\frac{45-30}{4-1} = \frac{15}{3} = 5$			
			4	$\Delta_4^{2,2}(\Omega_1^1, \Omega_2^2) = 45$	2	$\frac{15-45\cdot 2}{3-2} < 0$			
		Ω_2^4	0	$\Delta_0^{ij}(\Omega_1^1, \Omega_2^4) = 0$	0	$\frac{380-320-0}{8-0} = \frac{60}{8} = 7.5$	7.5		
			1	$\Delta_1^{2,3}(\Omega_1^1, \Omega_2^4) = 15$	1	$\frac{60-15}{8-1} = \frac{45}{7} = 6\frac{3}{7}$			
			2	$\Delta_2^{2,1}(\Omega_1^1, \Omega_2^4) = 15$	2	$\frac{45-15\cdot 2}{7-2} = \frac{15}{5} = 3$			
			3	$\Delta_3^{1,1}(\Omega_1^1, \Omega_2^4) = 30$	2	$\frac{15-30\cdot 2}{5-2} < 0$			
	Ω_1^2, Ω_1^5	Ω_2^1	0	$\Delta_0^{ij}(\Omega_1^2, \Omega_2^1) = 0$	0	$\frac{410-305-0}{6-0} = \frac{105}{6} = 17.5$	18	25	
			1	$\Delta_1^{2,3}(\Omega_1^2, \Omega_2^1) = 15$	1	$\frac{105-15}{6-1} = \frac{90}{5} = 18$			
			2	$\Delta_2^{2,1}(\Omega_1^2, \Omega_2^1) = 25$	1	$\frac{90-25}{5-1} = \frac{65}{4} = 16.25$			
			3	$\Delta_3^{1,1}(\Omega_1^2, \Omega_2^1) = 40$	1	$\frac{65-40}{4-1} = \frac{25}{3} = 8\frac{1}{3}$			
			4	$\Delta_4^{2,2}(\Omega_1^2, \Omega_2^1) = 45$	2	$\frac{25-44\cdot 2}{3-2} < 0$			
		Ω_2^2, Ω_2^3	0	$\Delta_0^{ij}(\Omega_1^2, \Omega_2^2) = 0$	0	$\frac{395-305-0}{4-0} = \frac{90}{4} = 22.5$	25		
			1	$\Delta_1^{2,3}(\Omega_1^2, \Omega_2^2) = 15$	1	$\frac{90-15}{4-1} = \frac{75}{3} = 25$			
			2	$\Delta_2^{2,2}(\Omega_1^2, \Omega_2^2) = 45$	2	$\frac{75-45\cdot 2}{3-2} < 0$			
		Ω_2^4	0	$\Delta_0^{ij}(\Omega_1^2, \Omega_2^4) = 0$	0	$\frac{380-305-0}{6-0} = \frac{75}{6} = 12.5$	12.5		
			1	$\Delta_1^{2,1}(\Omega_1^2, \Omega_2^4) = 15$	1	$\frac{75-15}{6-1} = \frac{60}{5} = 12$			
			2	$\Delta_2^{2,3}(\Omega_1^2, \Omega_2^4) = 15$	1	$\frac{60-15}{5-1} = \frac{45}{4} = 11.25$			
			3	$\Delta_3^{1,1}(\Omega_1^2, \Omega_2^4) = 30$	1	$\frac{45-30}{4-1} = \frac{15}{3} = 5$			
			4	$\Delta_4^{2,2}(\Omega_1^2, \Omega_2^4) = 45$	2	$\frac{15-45\cdot 2}{3-2} < 0$			
	Ω_1^6	Ω_2^1	0	$\Delta_0^{ij}(\Omega_1^6, \Omega_2^1) = 0$	0	$\frac{410-290-0}{8-0} = \frac{120}{8} = 15$	15	25	
			1	$\Delta_1^{2,3}(\Omega_1^6, \Omega_2^1) = 15$	1	$\frac{120-15}{8-1} = \frac{105}{7} = 15$			
			2	$\Delta_2^{2,1}(\Omega_1^6, \Omega_2^1) = 25$	2	$\frac{105-25\cdot 2}{7-2} = \frac{55}{5} = 11$			
			3	$\Delta_3^{1,1}(\Omega_1^6, \Omega_2^1) = 40$	2	$\frac{55-40\cdot 2}{5-2} < 0$			
		Ω_2^2, Ω_2^3	0	$\Delta_0^{ij}(\Omega_1^6, \Omega_2^2) = 0$	0	$\frac{395-290-0}{6-0} = \frac{105}{6} = 17.5$	18		
			1	$\Delta_1^{2,3}(\Omega_1^6, \Omega_2^2) = 15$	1	$\frac{105-15}{6-1} = \frac{90}{5} = 18$			
			2	$\Delta_2^{2,1}(\Omega_1^6, \Omega_2^2) = 25$	1	$\frac{90-25}{5-1} = \frac{65}{4} = 16.25$			
			3	$\Delta_3^{1,1}(\Omega_1^6, \Omega_2^2) = 40$	1	$\frac{65-40}{4-1} = \frac{25}{3} = 8\frac{1}{3}$			
			4	$\Delta_4^{2,2}(\Omega_1^6, \Omega_2^2) = 45$	2	$\frac{25-45\cdot 2}{3-2} < 0$			
		Ω_2^4	0	$\Delta_0^{ij}(\Omega_1^6, \Omega_2^4) = 0$	0	$\frac{380-290-0}{4-0} = \frac{90}{4} = 22.5$	25		
			1	$\Delta_1^{2,3}(\Omega_1^6, \Omega_2^4) = 15$	1	$\frac{90-15}{4-1} = \frac{75}{3} = 25$			
			2	$\Delta_2^{2,2}(\Omega_1^6, \Omega_2^4) = 45$	2	$\frac{75-45\cdot 2}{3-2} < 0$			

Table 2.13 (continuation)**:** *Calculation of the relative stability radius* $\overline{\varrho}_1^{\Lambda(G)}(p \in T)$ *for problem* $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\sum \mathcal{C}_i$

G_k	$\Omega_1^v \in \Omega_{1k}^*$	Ω_k^u	β	$\Delta_\beta^{ij}(\Omega_1^v, \Omega_k^u)$	$N_\beta(\Delta)$	$\dfrac{\sum\limits_{\nu\in\Omega_k^u} l^p(\nu) - \sum\limits_{\mu\in\Omega_1^v} l^p(\mu) - \sum\limits_{\alpha=0}^{\beta} \Delta_\alpha^{ij}(\Omega_1^v, \Omega_k^u)\cdot N_\alpha(\Delta)}{\sum\limits_{O_{ij}\in Q^J} \lvert n_{ij}(\Omega_k^u) - n_{ij}(\Omega_1^v)\rvert - \sum\limits_{\alpha=0}^{\beta} \lvert N_\alpha(\Delta)\rvert}$	$\max\limits_{\beta}$	$\max\limits_{\Omega_k^u}$	$\min\limits_{\Omega_1^v}$
1	2	3	4	5	6	7	8	9	10
G_3	Ω_1^1	Ω_3^1	0	$\Delta_0^{ij}(\Omega_1^1, \Omega_3^1) = 0$	0	$\frac{425-320-0}{6-0} = \frac{105}{6} = 17.5$	18.75	18.75	18.75
			1	$\Delta_1^{2,1}(\Omega_1^1, \Omega_3^1) = 15$	2	$\frac{105-15\cdot 2}{6-2} = \frac{75}{4} = 18.75$			
			2	$\Delta_2^{1,1}(\Omega_1^1, \Omega_3^1) = 30$	1	$\frac{75-30}{4-1} = \frac{45}{3} = 15$			
			3	$\Delta_3^{2,2}(\Omega_1^1, \Omega_3^1) = 45$	2	$\frac{45-45\cdot 2}{3-2} < 0$			
		Ω_3^2	0	$\Delta_0^{ij}(\Omega_1^1, \Omega_3^2) = 0$	0	$\frac{380-290-0}{8-0} = \frac{60}{8} = 7.5$	7.5		
			1	$\Delta_1^{2,1}(\Omega_1^1, \Omega_3^2) = 15$	1	$\frac{60-15}{8-1} = \frac{45}{7} = 6\frac{3}{7}$			
			2	$\Delta_2^{2,3}(\Omega_1^1, \Omega_3^2) = 15$	2	$\frac{45-15\cdot 2}{7-2} = \frac{15}{5} = 3$			
			3	$\Delta_3^{1,1}(\Omega_1^1, \Omega_3^2) = 30$	2	$\frac{15-30\cdot 2}{5-2} < 0$			
	Ω_1^2, Ω_1^5	Ω_3^1	0	$\Delta_0^{ij}(\Omega_1^2, \Omega_3^1) = 0$	0	$\frac{425-305-0}{4-0} = \frac{120}{4} = 30$	35	35	
			1	$\Delta_1^{2,1}(\Omega_1^2, \Omega_3^1) = 15$	1	$\frac{120-15}{4-1} = \frac{105}{3} = 35$			
			2	$\Delta_2^{2,2}(\Omega_1^2, \Omega_3^1) = 45$	2	$\frac{105-45\cdot 2}{3-2} = 15$			
		Ω_3^2	0	$\Delta_0^{ij}(\Omega_1^2, \Omega_3^2) = 0$	0	$\frac{380-305-0}{6-0} = \frac{75}{6} = 12.5$	12.5		
			1	$\Delta_1^{2,1}(\Omega_1^2, \Omega_3^2) = 15$	1	$\frac{75-15}{6-1} = \frac{60}{5} = 12$			
			2	$\Delta_2^{2,2}(\Omega_1^2, \Omega_3^2) = 45$	1	$\frac{60-15}{5-1} = \frac{45}{4} = 11.25$			
			3	$\Delta_3^{1,1}(\Omega_1^2, \Omega_3^2) = 30$	1	$\frac{45-30}{4-1} = \frac{15}{3} = 5$			
			4	$\Delta_4^{2,2}(\Omega_1^2, \Omega_3^2) = 45$	2	$\frac{15-45\cdot 2}{3-2} < 0$			
	Ω_1^6	Ω_3^1	0	$\Delta_0^{ij}(\Omega_1^6, \Omega_3^1) = 0$	0	$\frac{425-290-0}{4-0} = \frac{135}{4} = 33.75$	33.75	33.75	
			1	$\Delta_1^{1,1}(\Omega_1^6, \Omega_3^1) = 40$	1	$\frac{135-40}{4-1} = \frac{95}{3} = 31\frac{2}{3}$			
			2	$\Delta_2^{2,2}(\Omega_1^6, \Omega_3^1) = 45$	2	$\frac{95-45\cdot 2}{3-2} = 5$			
		Ω_3^2	0	$\Delta_0^{ij}(\Omega_1^6, \Omega_3^2) = 0$	0	$\frac{380-290-0}{4-0} = \frac{90}{4} = 22.5$	25		
			1	$\Delta_1^{2,3}(\Omega_1^6, \Omega_3^2) = 15$	1	$\frac{90-15}{4-1} = \frac{75}{3} = 25$			
			2	$\Delta_2^{2,2}(\Omega_1^6, \Omega_3^2) = 45$	2	$\frac{75-45\cdot 2}{3-2} < 0$			
G_4	Ω_1^6	Ω_4^1	0	$\Delta_0^{ij}(\Omega_1^6, \Omega_4^1) = 0$	0	$\frac{435-290-0}{4-0} = \frac{145}{4} = 36.25$	40	40	40
			1	$\Delta_1^{2,1}(\Omega_1^6, \Omega_4^1) = 25$	1	$\frac{145-25}{4-1} = \frac{120}{3} = 40$			
			2	$\Delta_2^{1,1}(\Omega_1^6, \Omega_4^1) = 40$	2	$\frac{120-40\cdot 2}{3-2} = 40$			

We draw the projections of the stability balls in Figure 2.9 for the same components $p_{1,3}$ *and* $p_{2,2}$ *of vector* p *as for problem* $\mathcal{J}3/n = 2, a_i \le p_i \le b_i/\mathcal{C}_{max}$ *from Example 2.1 in Section 2.1 (see Figure 2.4 and Figure 2.5). From Theorem 2.4 and Remark 2.9, it follows that the set* $\Lambda^*(G) = \{G_1\} \cup \{\cup_{i=1}^3 \Gamma_i\} = \{G_1\} \cup \{G_k : G_1 \not\preceq_T G_k\} = \{G_1, G_2, G_3, G_4\}$ *is a G-solution of problem* $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\sum \mathcal{C}_i$*. Moreover, this G-solution is minimal since for each digraph* $G_k \in \Lambda^*(G)$*, there exists a feasible vector for which this digraph is the unique optimal one (see Table 2.14).*

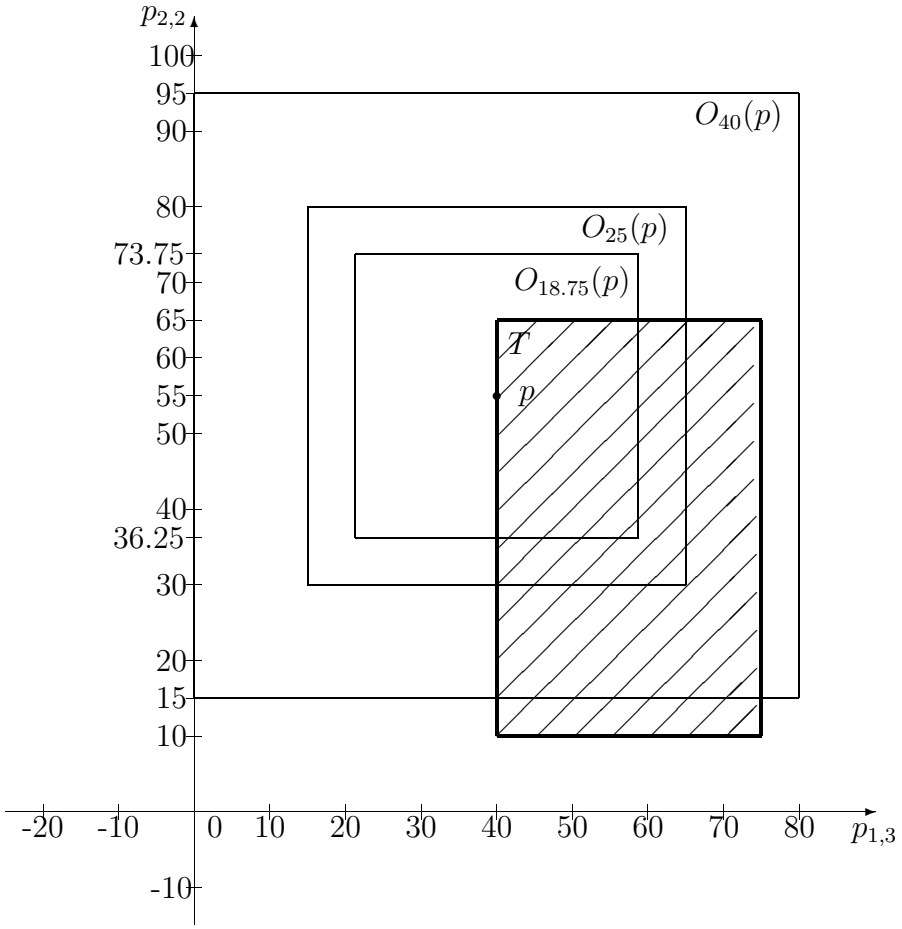


Figure 2.9: *Projections of the stability balls with the center* $p = (75, 50, 40, 60, 55, 30)$ *on the plane for problem* $\mathcal{J}3/n=2, a_i \le p_i \le b_i / \sum \mathcal{C}_i$

Table 2.14: *Optimal digraphs for problem* $\mathcal{J}3/n=2, a_i \le p_i \le b_i / \sum \mathcal{C}_i$ *with different initial vectors* $p \in T$

Initial vector $p^j \in T$	Objective function values $\sum \mathcal{C}_i$	Optimal digraph G_s	$\overline{\varrho}_s^{\Lambda(G)}(p^j \in T)$	Competitive digraph of G_s
1	2	3	4	5
$p^1=(75,95,40,60,10,30)$	$L_2^{p^1}=365, L_3^{p^1}=380,$ $L_1^{p^1}=410, L_4^{p^1}=480,$ $L_5^{p^1}=550$	G_2	$\overline{r}_{3,2}^{\Lambda(G)}=3.75$	G_3
$p^2=(80,95,40,55,10,35)$	$L_3^{p^2}=380, L_2^{p^2}=385,$ $L_1^{p^2}=425, L_4^{p^2}=490,$ $L_5^{p^2}=555$	G_3	$\overline{r}_{2,3}^{\Lambda(G)}=1.25$	G_2
$p^3=(35,35,50,85,10,30)$	$L_4^{p^3}=315, L_2^{p^3}=320,$ $L_1^{p^3}=335, L_3^{p^3}=340,$ $L_5^{p^3}=410$	G_4	$\overline{r}_{1,4}^{\Lambda(G)}=1.25$	G_1

As we have noted for criterion $\mathcal{C}_{max}$ (Remark 2.5), fixing the vector $p \in T$ and the choice of an optimal digraph $G_s(p)$ may have a large influence on

the resulting G-solution of problem $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$.

In the rest of this section, we prove a bound for the relative stability radii $\overline{\varrho}_s^B(p \in T)$, which is analogous to the bound proven for the relative stability radius $\hat{\varrho}_s^B(p \in T)$ for the makespan criterion (see Lemma 2.4 on page 114). This bound can restrict the number of feasible digraphs from set B which have to be considered while calculating the relative stability radius.

Redundant Digraphs for Calculating $\overline{\varrho}_s^B(p \in T)$

To calculate the relative stability radius $\overline{\varrho}_s^B(p \in T)$ of an optimal digraph G_s, we can use formulas (2.48) and (2.55) from Theorem 2.6. More exactly, one must compare each set $\Omega_s^v, v = 1, 2, \ldots, \omega_s^T$, of representatives of the family of sets $(H_s^i)_{J_i \in J}$, with the sets $\Omega_k^u, u = 1, 2, \ldots, \omega_k^T$, of representatives of the family of sets $(H_k^i)_{J_i \in J}$ of each digraph $G_k \in B \subseteq \Lambda(G)$, $k = 1, 2, \ldots, |B|$, $k \neq s$. The following bound, in which $\overline{r}_{ks}^B$ is defined by formula (2.54) on page 130, restricts the number of feasible digraphs G_k with which a comparison of the optimal digraph G_s has to be done during the calculation of the relative stability radius $\overline{\varrho}_s^B(p \in T)$.

Lemma 2.8 *If $\overline{\varrho}_s^B(p \in T) < \infty$ and there exists a digraph $G_k \in B$ such that*

$$\overline{r}_{ks}^B \le \frac{L_t^p - L_s^p}{nq - n} \tag{2.63}$$

for some digraph $G_t \in B$, then it is not necessary to consider digraph G_t during the calculation of the relative stability radius $\overline{\varrho}_s^B(p \in T)$.

Proof. To calculate the relative stability radius $\overline{\varrho}_s^B(p \in T)$ using Theorem 2.6, we have to compare the optimal digraph G_s consecutively with each feasible digraph $G_i, i \neq s$, from set B. Let us compare the optimal digraph G_s with a feasible digraph G_t, $t \neq k$. Digraph $G_t, t \neq s$, is a competitive digraph for G_s if we can construct a vector $\overline{x} \in T$ that satisfies condition (a^*) given on page 127, i.e., equality (2.38) holds: $L_s^{\overline{x}} = L_t^{\overline{x}}$. Moreover, for any real $\epsilon > 0$, which may be as small as desired, there must exist a vector $\overline{p}^\epsilon \in T$ such that equality $d(\overline{x}, \overline{p}^\epsilon) = \epsilon$ holds and inequality (2.39) $L_s^{\overline{p}^\epsilon} > L_t^{\overline{p}^\epsilon}$ is satisfied for digraph G_t (see condition (b^*)). More precisely, we must construct a vector $\overline{x}$ of the form $\overline{x} = p(\overline{r}_{ts}^B) = (p_{1,1}(\overline{r}_{ts}^B), p_{1,2}(\overline{r}_{ts}^B), \ldots, p_{nn_n}(\overline{r}_{ts}^B))$ with the components $p_{ij}(\overline{r}_{ts}^B)$ from the set $\{p_{ij}, p_{ij} + \min\{\overline{r}_{ts}^B, b_{ij} - p_{ij}\}, p_{ij} - \min\{\overline{r}_{ts}^B, p_{ij} - a_{ij}\}\}$ according to formula (2.45). Due to condition (c^*) (see page 127), the distance

$d(p,\overline{x}) = d(p,p(\overline{r}^B_{ts})) = \overline{r}^B_{ts}$ must achieve the minimal value among the distances between vector p and the other vectors in polytope T which satisfy both conditions (a^*) and (b^*).

We suppose that the conditions of Lemma 2.8 are satisfied, i.e., inequality (2.63) holds, and vector $\overline{x} = p(\overline{r}^B_{ks})$ satisfies both conditions (a^*) and (b^*). Then we show that the distance $d(p,p(\overline{r}^B_{ts}))$ cannot become less than the distance $d(p,p(\overline{r}^B_{ks}))$. Next, we show that inequality $\overline{r}^B_{ks} \le \overline{r}^B_{ts}$ follows from condition (2.63). We have:

$$\overline{r}^B_{ks} \le \frac{L^p_t - L^p_s}{nq-n} = \frac{\sum_{\nu\in\Omega^{u^*}_t} l^p(\nu) - \sum_{\mu\in\Omega^{v^*}_s} l^p(\mu)}{n(q-1)} = \overline{r}',$$

where $\Omega^{u^*}_t, u^* \in \{1,2,\ldots,\omega^T_t\}$, and $\Omega^{v^*}_s, v^* \in \{1,2,\ldots,\omega^T_s\}$, are critical sets for the digraphs G_t and G_s, respectively. Since $\sum_{O_{ij}\in Q^J} |n_{ij}(\Omega^{u^*}_k) - n_{ij}(\Omega^{v^*}_s)| < n$, we get the following inequalities:

$$\begin{aligned}
\overline{r}' &< \frac{\sum_{\nu\in\Omega^{u^*}_t} l^p(\nu) - \sum_{\mu\in\Omega^{v^*}_s} l^p(\mu)}{(\sum_{O_{ij}\in Q^J} |n_{ij}(\Omega^{u^*}_k) - n_{ij}(\Omega^{v^*}_s)|)(q-1)} \\
&\le \max_{u\in\{1,2,\ldots,\omega^T_t\}} \frac{\sum_{\nu\in\Omega^{u}_t} l^p(\nu) - \sum_{\mu\in\Omega^{v^*}_s} l^p(\mu)}{\sum_{O_{ij}\in Q^J} |n_{ij}(\Omega^{u}_t) - n_{ij}(\Omega^{v^*}_s)|} \\
&\le \min_{\Omega^v_s\in\Omega_{s,t}} \max_{u\in\{1,2,\ldots,\omega^T_t\}} \frac{\sum_{\nu\in\Omega^{u}_t} l^p(\nu) - \sum_{\mu\in\Omega^{v}_s} l^p(\mu)}{\sum_{O_{ij}\in Q^J} |n_{ij}(\Omega^{u}_t) - n_{ij}(\Omega^{v}_s)|} \le \quad \overline{r}^B_{ts}.
\end{aligned}$$

Thus, the value $\overline{r}^B_{ts}$ cannot become less than $\overline{r}^B_{ks}$ and therefore, digraph G_t need not to be considered during the calculation of the relative stability radius $\overline{\varrho}^B_s(p \in T)$.

$\diamond$

Bound (2.63) is tight. This lemma implies the following corollary.

Corollary 2.12 *Let set $B = \{G_s = G_{i_1}, G_{i_2}, \ldots, G_{i_{|B|}}\}$ be sorted in non-decreasing order of the objective function values $L^p_{i_1} \le L^p_{i_2} \le \ldots \le L^p_{i_{|B|}}$. If for the currently compared digraph G_{i_k} from set $B \subseteq \Lambda(G)$, inequality*

$$\overline{r}^B_{i_k s} \le \frac{L^p_{i_t} - L^p_{i_1}}{nq-n} \tag{2.64}$$

holds for digraph $G_{i_t} \in B$ with $L^p_{i_k} \le L^p_{i_t}$, then it is possible to exclude the digraphs $G_{i_t}, G_{i_{t+1}}, \ldots, G_{i_{|B|}}$ from further considerations during the calculation of the relative stability radius $\overline{\varrho}^B_s(p \in T)$.

PROOF. Since set $B \subseteq \Lambda(G)$ is sorted in non-decreasing order of the objective function values and inequality (2.64) holds for digraph G_{i_t}, inequality $\overline{r}^{B}_{i_k s} \leq \frac{L^p_{i_j} - L^p_{i_1}}{nq-n}$ holds for each digraph $G_{i_j}, j = t+1, t+2, \ldots, |B|$. Due to Lemma 2.8, these digraphs need not to be considered during the calculation of the relative stability radius.

$\diamond$

Using Corollary 2.12, we can compare the optimal digraph $G_s = G_{i_1}$ consecutively with the digraphs $G_{i_2}, G_{i_3}, \ldots, G_{i_{|B|}}$ from set B in non-decreasing order of the objective function values: $L^p_{i_1} \leq L^p_{i_2} \leq \ldots \leq L^p_{i_{|B|}}$. If for the currently compared digraph $G_k = G_{i_r}$, inequality (2.63) holds, we can exclude digraphs $G_{i_r}, G_{i_{r+1}}, \ldots, G_{i_{|B|}}$ from further considerations.

Since $\overline{\varrho}_s(p) = \overline{\varrho}^{\Lambda(G)}_s(p \in R^q_+)$, Corollary 2.12 implies Corollary 2.13 which allows us to restrict the number of feasible digraphs while calculating the stability radius $\overline{\varrho}_s(p)$ (see Definition 1.2 on page 28).

Corollary 2.13 *Let set $\Lambda(G) = \{G_s = G_{i_1}, G_{i_2}, \ldots, G_{i_\lambda}\}$ be sorted in non-decreasing order of the objective function values: $L^p_{i_1} \leq L^p_{i_2} \leq \ldots \leq L^p_{i_\lambda}$. If for the currently compared digraph G_{i_k} from set $\Lambda(G) = \{G_s = G_{i_1}, G_{i_2}, \ldots, G_{i_k}, \ldots, G_{i_t}, \ldots, G_{i_\lambda}\}$, inequality*

$$\overline{r}^{\Lambda(G)}_{i_k s} \leq \frac{L^p_{i_t} - L^p_{i_1}}{q} \tag{2.65}$$

holds for digraph $G_{i_t} \in \Lambda(G)$ with $L^p_{i_k} \leq L^p_{i_t}$, then it is possible to exclude the digraphs $G_{i_t}, G_{i_{t+1}}, \ldots, G_{i_\lambda}$ from further considerations during the calculation of the stability radius $\overline{\varrho}_s(p)$.

2.6. Algorithms for Problem $\mathcal{J}/a_i \leq p_i \leq b_i/\Phi$

In this section, we focus on the criterion $\sum \mathcal{C}_i$ but we indicate the necessary changes for the criterion $\mathcal{C}_{max}$ as well. Using the above mathematical background, we propose Algorithm $SOL\_\sum \mathcal{C}_i$ for finding a G-solution $\Lambda^*(G) \subseteq \Lambda(G)$ with a 'relatively small' cardinality. As the input data for Algorithm $SOL\_\sum \mathcal{C}_i$, a set of schedules $B \subseteq \Lambda(G)$, which is a G-solution of problem $\mathcal{J}/a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$, and a vector $p \in T$ of the processing times are used. This algorithm generates a covering of polytope T (Theorem 2.4) by nested closed balls $O_r(p)$ with the common center $p \in T$ and different radii r which are relative stability radii $\overline{\varrho}^{B}_s(p \in T)$ of the same digraph G_s but for different nested sets B. Let set $B \subseteq \Lambda(G)$ be a given G-solution to problem

$\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$ (in the worst case, the whole set $\Lambda(G)$ of digraphs may be used as set B). We also fix a vector $p \in T$ of the processing times and the number the digraphs of set $B = \{G_1, G_2, \ldots, G_{|B|}\}$ in non-decreasing order of the values of the objective function. An 'expected' vector of the processing times (or a vector which has been considered in a previous calculation or some other suitable vector from polytope T) may be used as the *input vector* p in the following algorithm. In square brackets we give the changes of this algorithm in the case of criterion $\mathcal{C}_{max}$.

Algorithm $SOL\_\sum \mathcal{C}_i$ [**Algorithm** $SOL\_\mathcal{C}_{max}$]

Input: Fixed vector $p = (p_{1,1}, p_{1,2}, \ldots, p_{nn_n}) \in T$,
set $B = \{G_1, G_2, \ldots, G_{|B|}\}$ such that $L^p_1 \le L^p_2 \le \ldots \le L^p_{|B|}$
for criterion $\sum \mathcal{C}_i$ [$l^p_1 \le l^p_2 \le \ldots \le l^p_{|B|}$ for criterion $\mathcal{C}_{max}$].
Output: Relative stability radius $\overline{\varrho}^B_1(p \in T)$ [$\widehat{\varrho}^B_1(p \in T)$]
of the optimal digraph G_1 and a G-solution $\Lambda^*(G)$.

Step 1: Set $k = 2$ and $\Lambda = \emptyset$.
Step 2: For digraph $G_k \in B$, test the dominance relation
$G_1 \preceq_T G_k$ using Lemma 2.5 with the
objective function $\Phi^p_k = L^p_k$ [$\Phi^p_k = l^p_k$].
Step 3: **IF** $G_1 \not\preceq_T G_k$ **THEN** calculate value $\overline{r}^B_{k1}$
[value $\widehat{r}^B_{k1}$] using formulas (2.48) and (2.54)
[formula (2.25)] for the input vector p
ELSE GOTO *Step 5*.
Step 4: Set $\Lambda := \Lambda \cup \{G_k\}$.
Step 5: Set $k := k + 1$.
IF $k \le |B|$ **THEN GOTO** *Step 2*.
ELSE using Theorem 2.6
[Theorem 2.3 and Remark 2.6]
calculate $\overline{\varrho}^B_1(p \in T) = \min\{\overline{r}^B_{k1} : \ G_1 \not\preceq_T G_k\}$
[$\widehat{\varrho}^B_1(p \in T) = \min\{\widehat{r}^B_{k1} : \ G_1 \not\preceq_T G_k\}$].
Set $\Lambda^*(G) = \Lambda \cup \{G_1\}$ **STOP**

It is easy to see that set $\Lambda^*(G) = \{G_{i_1=1}, G_{i_2}, \ldots, G_{i_{|\Lambda^*(G)|}}\}$, $i_1 < i_2 < \ldots < i_{|\Lambda^*(G)|}$, generated by Algorithm $SOL\_\sum \mathcal{C}_i$ is a G-solution of problem $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$. Indeed, set $\Lambda^*(G)$ is a subset of set B which is assumed to be a G-solution of problem $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$ and set $\Lambda^*(G)$ includes digraph G_1 and also each digraph G_k, $2 \le k \le |B|$, provided that dominance relation $G_1 \preceq_T G_k$ does not hold.

Along with a G-solution $\Lambda^*(G)$, Algorithm $SOL\_\sum C_i$ calculates the value $\overline{r}^B_{k1}$ for each digraph $G_k \in B$ such that dominance relation $G_1 \preceq_T G_k$ does not hold (see Step 3). The value $\overline{r}^B_{k1}$ denotes the largest distance $d(p, p')$ such that inequality $L_1^{p'} > L_k^{p'}$ is guaranteed for each vector $p' \in T$ of the processing times. Therefore, dominance relation $G_1 \preceq_{T^*} G_k$ holds for each polytope $T^* = T \cap O_r(p)$ if $r \leq \overline{r}^B_{k1}$, and we have $G_1 \not\preceq_{T^*} G_k$ for polytope $T^* = T \cap O_r(p)$ if $r > \overline{r}^B_{k1}$. Let us put the digraphs in set $\Lambda^*(G)$ in non-decreasing order of the values $\overline{r}^B_{k1}$: $\Lambda^*(G) = \{G_{j_1=1}, G_{j_2}, \ldots, G_{j_{|\Lambda^*(G)|}}\}$, where $\overline{r}^B_{j_2 1} \leq \overline{r}^B_{j_3 1} \leq \ldots \leq \overline{r}^B_{j_{|\Lambda^*(G)|}1}$. Due to Theorem 2.6, it follows that $\overline{\varrho}^B_1(p \in T) = \overline{r}^B_{j_2 1}$. Similarly, for set $B \setminus \{G_{j_2}\}$, we have equality $\overline{\varrho}_1^{B\setminus\{G_{j_2}\}}(p \in T) = \overline{r}^B_{j_3 1}$, and in general, we have equality $\overline{\varrho}_1^{B\setminus\{\cup_{k=2}^{l} G_{j_k}\}}(p \in T) = \overline{r}^B_{j_{l+1}1}$, where $1 < l < |\Lambda^*(G)|$.

These values $\overline{r}^B_{k1}$ will be used in Algorithm $MINSOL\_\sum C_i$ which follows. Moreover, they may be used in a realization of the best schedule. Indeed, to realize a G-solution $\Lambda^*(G)$ (when the values $\overline{r}^B_{k1}$ are known), we can start with digraph G_1 which is the optimal digraph (one of the optimal digraphs) for the 'expected' vector $p \in T$ of the processing times. If we will get additional information about the error r of the processing times p_{ij}, we can use r for a suitable modification of the schedule which is currently realized. To this end, we select $\overline{r}^B_{j_l 1}$ such that inequalities $\overline{r}^B_{j_l 1} < r \leq \overline{r}^B_{j_{l+1}1}$ hold, and we can find a better digraph in set $\cup_{u=1}^{l} G_{i_u}$ which may be realized further instead of the initial digraph G_1. It is practically important that, if the possible error of the given processing times is no more than r, we have the guarantee that set $\cup_{u=1}^{l} G_{i_u}$ contains at least one optimal digraph.

Note that the G-solution $\Lambda^*(G)$ generated by Algorithm $SOL\_\sum C_i$ may be not minimal. To exclude redundant digraphs, we can test the dominance relation $\preceq_T$ between the digraphs from set $\Lambda^*(G) \setminus \{G_{i_1=1}\}$ which may be done as follows. First, we exclude all digraphs G_{i_k}, $2 < k \leq |\Lambda^*(G)|$, from the set $\Lambda^*(G)$ for which dominance relation $G_{i_2} \preceq_T G_{i_k}$ holds. To this end, we repeat Algorithm $SOL\_\sum C_i$ with the set $\Lambda^*(G) \setminus \{G_{i_1=1}\}$ being used instead of set B. Then, similarly, we can exclude all digraphs from the G-solution which are dominated by digraph G_{i_3} and so on. After no more than $|\Lambda^*(G)| - 2$ repetitions of Algorithm $SOL\_\sum C_i$, we can remove all redundant digraphs (or an essential part of the redundant digraphs) from the set $\Lambda^*(G)$. As a result, we often obtain a minimal G-solution $\Lambda^T(G)$. Next, we give a formal algorithm for finding a minimal G-solution on the basis of the above repetitions of Algorithm $SOL\_\sum C_i$ (Step 3) and the verification of the strong dominance relation (Step 5). We set $\Lambda' = \Lambda^*(G) \setminus \{G_{i_1=1}\} =$

$\{G_{i_2}, G_{i_3}, \ldots, G_{i_{|\Lambda^*(G)|}}\}$, where $\Lambda^*(G)$ is obtained by Algorithm $SOL\_\sum \mathcal{C}_i$ provided that inequalities $L^p_2 \le L^p_3 \le \ldots \le L^p_{|\Lambda^*(G)|}$ hold.

Algorithm $MINSOL\_\sum \mathcal{C}_i$ [**Algorithm** $MINSOL\_\mathcal{C}_{max}$]

Input: Set $\Lambda' = \Lambda^*(G) \setminus \{G_{i_1=1}\} = \{G_{i_2}, G_{i_3}, \ldots, G_{i_{|\Lambda^*(G)|}}\}$.
Output: Minimal dominant set $\Lambda^T(G)$.

Step 1: Set $\Lambda^T(G) = \{G_{i_1=1}\}$.
Step 2: Set $B = \Lambda'$ and change the subscripts of the digraphs as follows: $G_u := G_{i_{u+1}}$, $1 \le u < |\Lambda'| - 1$, i.e., in the following *Steps 3* and *4* the ordered set $(G_{i_2}, G_{i_3}, \ldots, G_{|\Lambda'|+1})$ will be referred to as the ordered set $(G_1, G_2, \ldots, G_{|\Lambda'|})$.
Step 3: Implement Algorithm $SOL\_\sum \mathcal{C}_i$ [Algorithm $SOL\_\mathcal{C}_{max}$] with the input set $B = \{G_1, G_2, \ldots, G_{|B|}\}$ defined in *Step 2* and with the same vector p.
Step 4: Set $\Lambda' := \Lambda' \setminus \{G_1\}$ and $\Lambda^T(G) := \Lambda^T(G) \cup \{G_1\}$.
IF $|\Lambda'| \ge 2$ **THEN GOTO** *Step 2*
ELSE GOTO *Step 5*.
Step 5: **FOR** each digraph $G_s \in \Lambda^T(G)$ **DO**
BEGIN
Calculate vector $p^{(s)} \in T$ such that the strong dominance relation $G_s \prec_{p^{(s)}} G_k$ holds for each digraph $G_k \in \Lambda^T(G) \setminus \{G_s\}$.
IF there does not exist such a vector $p^{(s)} \in T$
THEN set $\Lambda^T(G) := \Lambda^T(G) \setminus \{G_s\}$
END STOP

Obviously, the G-solution $\Lambda^T(G)$ generated by Algorithm $MINSOL\_\sum \mathcal{C}_i$ satisfies the conditions of Theorem 2.5 and hence, this G-solution is minimal. However, Step 5 may be rather complicated. It needs to be discussed in more detail. As the desired vector $p^{(s)}$ for digraph G_s, we can test the vector $p_{ij}(r)$ calculated by formula (2.45) in Algorithm $SOL\_\sum \mathcal{C}_i$, where $r = \overline{r}^B_{ks} + \epsilon$ with ϵ being a small positive real number. This vector will be either sufficient for Step 5 or not. In the latter case, i.e., when for vector $p^{(s)}$ the strong dominance relation $G_s \prec_{p^{(s)}} G_k$ does not hold for at least one digraph $G_k \in \Lambda^T(G) \setminus \{G_s\}$, the realization of Step 5 in Algorithm $MINSOL\_\sum \mathcal{C}_i$ may be more sophisticated.

In our computational experiments we test Algorithm $MINSOL^*\_\sum\mathcal{C}_i$ (Algorithm $MINSOL^*\_\mathcal{C}_{max}$) which consists of Steps 1 – 4 of the above Algorithm $MINSOL\_\sum\mathcal{C}_i$ (Algorithm $MINSOL\_\mathcal{C}_{max}$). If for the G-solution $\Lambda^T(G) = \Lambda^*(G)$ generated by Algorithm $MINSOL^*\_\sum\mathcal{C}_i$ inequality $|\Lambda^*(G)| \leq 2$ holds, then set $\Lambda^*(G)$ obviously satisfies the conditions of Theorem 2.5 and therefore, this G-solution is minimal. If $|\Lambda^*(G)| > 2$, the G-solution $\Lambda^*(G)$ may be not minimal. Indeed, even if $\Lambda^*(G) = \{G_1, G_2, G_3\}$, Algorithm $MINSOL^*\_\sum\mathcal{C}_i$ only guarantees that no digraph from set $\Lambda^*(G)$ dominates another digraph from set $\Lambda^*(G)$. However, it might be that two digraphs 'jointly dominate' the remaining one which is not recognized by Algorithm $MINSOL^*\_\sum\mathcal{C}_i$. Nevertheless, Algorithm $MINSOL^*\_\sum\mathcal{C}_i$ often constructs a minimal G-solution even if $|\Lambda^T(G)| > 2$. Indeed, it is easy to see that, if a schedule is the unique optimal schedule in the interior of its stability region, then dominance relation $\preceq_D$ implies the strong dominance relation $\prec_D$ (except for points at the boundary of the stability region, where an optimal schedule usually is not unique). Fortunately, as it was demonstrated in [334] by experiments, a mean flow time optimal schedule is uniquely determined for most randomly generated job shop problems provided that the processing times are real numbers (not necessarily integers as it is often assumed in classical scheduling theory), and thus, due to the test of the dominance relation $\preceq_D$, Algorithm $MINSOL^*\_\sum\mathcal{C}_i$ usually constructs a minimal G-solution.

Next, we present three algorithms for constructing a G-solution B (for any regular criterion Φ) used as input set in Algorithm $SOL\_\sum\mathcal{C}_i$ (Algorithm $SOL\_\mathcal{C}_{max}$). The first one (called Algorithm $EXPL$) is based on an explicit enumeration of all semiactive schedules for the case of a classical job shop problem. The other two algorithms (called $B\&B1$ and $B\&B2$) are of the branch-and-bound type and may be used for the job shop problem $\mathcal{J}/a_i \leq p_i \leq b_i/\Phi$ with uncertain numerical input data and any regular criterion.

Algorithm $EXPL$

Input: Polytope T, mixed graph $G(p) = (Q(p), A, E), p \in T$.
Output: Optimal digraph $G_s(p)$, set $B = \{G_1^T, G_2^T, \ldots, G_{|B|}^T\} \subseteq \Lambda(G)$.

Step 1: Generate the feasible digraphs $\Lambda(G) = \{G_1, G_2, \ldots, G_\lambda\}$ via an explicit enumeration of the permutations of the operations Q_k for $k = 1, 2, \ldots, m$. Test whether the generated digraph has a circuit.

Step 2: Calculate the values Φ_k^a, Φ_k^b and Φ_k^p for each digraph G_k, $k = 1, 2, \ldots, \lambda$. Transform digraph G_k into digraph G_k^T.
Step 3: Find a digraph G_s^T such that $\Phi_s^b = \min\{\Phi_k^b : G_k \in \Lambda(G)\}$.
Step 4: Set $B = \{G_k^T : G_k \in \Lambda(G),\ \Phi_s^b > \Phi_k^a\}$ **STOP**

As follows from Lemma 2.5, set $\Lambda(G) \setminus B$ contains only digraphs G_k such that dominance relation $G_s^T \preceq_T G_k^T$ holds (Step 4 of Algorithm $EXPL$), and Algorithm $EXPL$ excludes only such digraphs from the set $\Lambda(G)$. To present branch-and-bound algorithms, we need the following arguments. Both branch-and-bound algorithms realize an implicit enumeration scheme which may be represented by a branching tree. Each vertex of this tree is a mixed graph $G_{(s)} = (Q, A_{(s)}, E_{(s)})$ with $A \subseteq A_{(s)}$ and $E_{(s)} \subseteq E$. The root of the tree is a mixed graph $G = G_{(1)}$, and a pair $G_{(s)}$ and $G_{(k)}$ is connected by the arc $(G_{(s)}, G_{(k)})$ if and only if the mixed graph $G_{(k)}$ is obtained directly from the mixed graph $G_{(s)}$ by orientating one edge. In both branch-and-bound algorithms under consideration, an edge is oriented only if it is a conflict one, i.e., when both orientations of this edge imply a conflict with previously calculated earliest starting times. Next, we give a formal definition of a conflict edge. For a mixed graph $G_{(s)} = (Q, A_{(s)}, E_{(s)})$ with $[O_{ij}, O_{uv}] \in E_{(s)}$, let us define the following three digraphs:

$$G_{s^0} = (Q, A_{(s)}, \emptyset),$$
$$G_{s'} = (Q, A_{(s)} \cup \{(O_{ij}, O_{uv})\}, \emptyset) \text{ and}$$
$$G_{s''} = (Q, A_{(s)} \cup \{(O_{uv}, O_{ij})\}, \emptyset).$$

Definition 2.7 *An edge $[O_{ij}, O_{uv}] \in E_{(s)}$ of the mixed graph $G_{(s)}$ is called a conflict edge, if there exists a vector $p \in T$ such that*

$$l_{s^0}^p(O_{uv}) < l_{s'}^p(O_{uv}), \tag{2.66}$$

$$l_{s^0}^p(O_{ij}) < l_{s''}^p(O_{ij}). \tag{2.67}$$

Obviously, if inequalities (2.66) and (2.67) hold, then each orientation of the edge $[O_{ij}, O_{uv}]$ implies an increase of value $l_{s'}^p(O_{uv})$ or value $l_{s''}^p(O_{ij})$. To verify whether an edge is a conflict one, one can use the following conditions.

Lemma 2.9 *An edge $[O_{ij}, O_{uv}] \in E_{(s)}$ is not a conflict edge if one of the following inequalities (2.68) or (2.69) holds:*

$$l_{s^0}^a(O_{uv}) \ge l_{s^0}^b(O_{ij}) + b_{ij}, \tag{2.68}$$

$$l_{s^0}^a(O_{ij}) \ge l_{s^0}^b(O_{uv}) + b_{uv}. \tag{2.69}$$

PROOF. It is easy to see that inequality (2.66) may hold only if a maximal path ending in vertex O_{uv} includes arc (O_{ij}, O_{uv}), i.e., if equality

$$l^p_{s'}(O_{uv}) = l^p_{s^0}(O_{ij}) + p_{ij} \tag{2.70}$$

holds. Similarly, inequality (2.67) may hold only if

$$l^p_{s''}(O_{ij}) = l^p_{s^0}(O_{uv}) + p_{uv}. \tag{2.71}$$

First, suppose that inequality (2.68) holds. For any vector $p \in T$, we obtain $l^p_{s^0}(O_{uv}) \geq l^a_{s^0}(O_{uv}) \geq l^b_{s^0}(O_{ij}) + b_{ij} \geq l^p_{s^0}(O_{ij}) + p_{ij}$. Taking into account (2.70), we conclude that inequality $l^p_{s^0}(O_{uv}) \geq l^p_{s'}(O_{uv})$ holds which means that edge $[O_{ij}, O_{uv}]$ is not a conflict one.

Now, suppose that inequality (2.69) holds. For any vector $p \in T$, we have $l^p_{s^0}(O_{ij}) \geq l^a_{s^0}(O_{ij}) \geq l^b_{s^0}(O_{uv}) + b_{uv} \geq l^p_{s^0}(O_{uv}) + p_{uv}$. Taking into account (2.71), we conclude that inequality $l^p_{s^0}(O_{ij}) \geq l^p_{s''}(O_{ij})$ holds which means that edge $[O_{ij}, O_{uv}]$ is not a conflict one.

$\diamond$

For each edge $[O_{ij}, O_{uv}] \in E_{(s)}$, one can calculate the *conflict measure*:

$$\min\{\max\{0,\ l^p_{s^0}(O_{ij}) + p_{ij} - \bar{l}^p_{s^0}(O_{uv})\},\ \max\{0,\ l^p_{s^0}(O_{uv}) + p_{uv} - \bar{l}^p_{s^0}(O_{ij})\}\},$$

where $\bar{l}^p_{s^0}(O_{ij})$ denotes the *latest starting time* of operation O_{ij}, i.e., the difference between the weight of the critical path μ in digraph G_{s^0} and the maximal weight of the path in G_{s^0} starting from vertex O_{ij}:

$$\bar{l}^p_s(O_{ij}) = l^p(\mu) - \sum_{O_{uv} \in [\nu]} p_{uv}.$$

Here path ν has the maximal weight among all paths in digraph G_{s^0} starting from O_{ij} and ending in vertex $O_{ln_l}, J_l \in J$.

This conflict measure gives the smallest possible increase of the earliest starting time of the operation due to the orientation of this edge (e.g., for a non-conflict edge this measure is equal to zero). So, in order to branch a set $\Lambda(G_{(s)})$ into two subsets $\Lambda(G_{(s')})$ and $\Lambda(G_{(s'')})$, where

$$G_{(s')} = (Q,\ A_{(s)} \cup \{(O_{ij}, O_{uv})\},\ E_{(s)} \setminus \{[O_{ij}, O_{uv}]\}) \text{ and}$$
$$G_{(s'')} = (Q,\ A_{(s)} \cup \{[O_{uv}, O_{ij}]\},\ E_{(s)} \setminus \{[O_{ij}, O_{uv}]\}),$$

we select the edge $[O_{ij}, O_{uv}]$ which has the largest value of the conflict measure. We use the lower bound (2.72) in both branch-and-bound algorithms. Indeed, for any digraph $G_t = (Q, A_{(s)} \cup A_t, \emptyset) \in \Lambda(G_{(s)})$, the bound

$$\sum_{i=1}^{n} l^p_t(O_{in_i}) \geq \sum_{i=1}^{n} l^p_{s^0}(O_{in_i}) \tag{2.72}$$

is valid since the set of arcs in digraph $G_{s^0} = (Q, A_{(s)}, \emptyset)$ is a subset of the arcs in digraph G_t. Note that, if digraph $G_{(s)}$ has no conflict edge, there exists a digraph $G_t \in \Lambda(G_{(s)})$ such that condition (2.72) is realized as equality. To construct such a digraph, we have to replace each remaining edge $[O_{ij}, O_{uv}] \in E_s$ by the arc (O_{ij}, O_{uv}) if inequality (2.68) holds, or by the arc (O_{uv}, O_{ij}) if inequality (2.69) holds. Obviously, for each p_{ij} and p_{uv} with $a_{ij} \le p_{ij} \le b_{ij}$ and $a_{uv} \le p_{uv} \le b_{uv}$, all operations in the resulting digraph will have the same earliest starting times as in digraph G_{s^0}. We use the latter as a stopping rule for branching the set $\Lambda(G_{(s)})$. Next, we present an algorithm for constructing a set of k schedules which are the best for the input vector $p \in T$ of the processing times and which will be used as the input set B in Algorithm $SOL\_\sum \mathcal{C}_i$ and Algorithm $SOL\_\mathcal{C}_{max}$ depending on the chosen objective function values $\Phi_s^p = L_s^p$ and $\Phi_s^p = l_s^p$, respectively.

Algorithm *B&B*1

Input: Polytope T, mixed graph $G(p) = (Q(p), A, E), p \in T$, number k of the best generated digraphs.
Output: Optimal digraph $G_s(p)$, set $B = \{G_1, G_2, \ldots, G_k\} \subseteq \Lambda(G)$.

Step 1: Set $X = \{G\} := \{G_{(1)}\}$, $Y = \emptyset$ and $\Phi = \infty$.
Step 2: **IF** $X = \emptyset$ **THEN GOTO** *Step 8*;
ELSE select a mixed graph $G_{(s)} \in X$ with the smallest value $\Phi_{s^0}^p$.
Set $X := X \setminus \{G_{(s)}\}$.
Step 3: **IF** the mixed graph $G_{(s)}$ has no conflict edge **THEN GOTO** *Step 6*.
Step 4: Select a conflict edge $[O_{ij}, O_{uv}] \in E_{(s)}$ with the largest conflict measure.
Step 5: **IF** $\Phi_{s'}^p < \Phi$ **THEN** set $X := X \cup \{G_{(s')}\}$;
IF $\Phi_{s''}^p < \Phi$ **THEN** set $X := X \cup \{G_{(s'')}\}$
GOTO *Step 2*.
Step 6: **IF** $|Y| < k$ **THEN** set $Y := Y \cup \{G_{(s)}\}$
GOTO *Step 2*;
ELSE IF $\Phi_{s^0}^p < \Phi$ (where $\Phi = \Phi_t^p$)
THEN set $Y := Y \cup \{G_{(s)}\} \setminus \{G_{(t)}\}$.
Step 7: Calculate $\Phi = \max\{\Phi_t^p \ : \ G_{(t)} \in Y\}$
GOTO *Step 2*.
Step 8: Construct the set $\Lambda(G_{(t)})$ for each mixed graph $G_{(t)} \in Y$.

Step 9: Select the subset B of the k best digraphs from set $\cup_{G_{(t)}\in Y}\Lambda(G_{(t)})$.
Step 10: Calculate $\Phi^* = \min\{\Phi^b_s \ : \ G_s \in B\}$.
Set $B := B \setminus \{G_t : \Phi^a_t \geq \Phi^*\}$ **STOP**

In Algorithm $B\&B1$, the lower bound for the objective function is calculated in Step 7, branching is realized in Step 5, and the stopping rule of branching is realized in Step 3. Step 6 has a special form in order to construct the k best schedules (instead of only one optimal schedule). Steps 8 and 9 are necessary only if $k > 1$. Indeed, if $k = 1$, then it is sufficient to consider only one best schedule from the set $\Lambda(G_{(s)})$, and for any mixed graph $G_{(s)} = (Q, A_{(s)}, E_{(s)})$, set $\Lambda(G_{(s)})$ has at least one best schedule $G_u \in \Lambda(G_{(s)})$ for which Φ^p_u reaches the minimal possible value Φ^p_s, where $G_{s^0} = (Q, A_{(s)}, \emptyset)$ (condition (2.72) turns into an equality). If $k > 1$, we have to generate also other schedules from the set $\Lambda(G_{(s)})$. Unfortunately, we cannot use Algorithm $EXPL$ for a fast generation of set $\Lambda(G_{(s)})$ because the edges of the set $E \setminus E_{(s)}$ are already oriented. Step 8 realizes a procedure based on the sequential orientation of non-conflict edges, which is essentially slower than the permutation enumeration used in Algorithm $EXPL$.

Using sufficiency of Lemma 2.5, Algorithm $B\&B2$ aims to construct a set of schedules which necessarily dominate all other schedules from the set $\Lambda(G)$ in polytope T. Steps 1–5 and Steps 8–10 in Algorithm $B\&B2$ are similar to those in Algorithm $B\&B1$. So, we describe only Steps 6 and 7 of Algorithm $B\&B2$, which are different from those in Algorithm $B\&B1$:

Algorithm $B\&B2$ **(specific part)**

Step 6: **IF** $\Phi^a_{s^0} \leq \Phi$ **THEN** set $Y := Y \cup \{G_{(s)}\}$.
Step 7: Calculate $\Phi = \min\{\Phi^b_{(t)} \ : \ G_{(t)} \in Y\}$ **GOTO** *Step 2.*

In Section 2.7, we present some computational results for randomly generated job shop problems solved by the above algorithms coded in Fortran-77.

Example 2.2 (continued). *As it was noted, the G-solution $\Lambda^*(G)$ and the minimal G-solution $\Lambda^T(G)$ of the problem with uncertain processing times may be not unique. From Remark 2.5, it follows that fixing the vector $p \in T$ and the choice of an optimal digraph $G_s(p)$ have a large influence on the resulting G-solution for criterion $\mathcal{C}_{max}$. For the job shop problem $\mathcal{J}3/n = 3, a_i \leq p_i \leq b_i/\sum\mathcal{C}_i$ from Example 2.2, we find a G-solution set $\Lambda^*(G)$ with different initial vectors $p \in T$ (see column 1 in Table 2.15) the components of which are taken from the closed intervals $[a_{ij}, b_{ij}]$ (vectors a and b are given*

Table 2.15: *G-solution of problem* $\mathcal{J}3/n=3, a_i \le p_i \le b_i/\sum \mathcal{C}_i$ *for different initial vectors* $p \in T$

Initial vector $p^j \in T$	G_s	Set B	$G_k,\ G_s \not\preceq_T G_k$	$\overline{r}^B_{ks}$
1	2	3	4	5
$p^1=(60,20,46,$ $30,70,80,50,30)$	G_5	$B=\{G_5,G_2,G_1,G_9,G_3,G_4,$ $G_7,G_6,G_8,G_{12},G_{11},G_{10}\}$	G_1	$\overline{\varrho}^B_5(p^1\in T)=\overline{r}^B_{1,5}=0.5$
			G_2	$\overline{r}^B_{2,5}=0.6667$
			G_3	$\overline{r}^B_{3,5}=5.6667$
			G_4	$\overline{r}^B_{4,5}=7.7143$
			G_6	$\overline{r}^B_{6,5}=12.3333$
			G_8	$\overline{r}^B_{8,5}=12.8333$
			G_7	$\overline{r}^B_{7,5}=14$
			G_{10}	$\overline{r}^B_{10,5}=19$
	G-solution: $\Lambda^*(G)=\{G_1,G_2,G_3,G_4,G_5,G_6,G_8,G_7,G_{10}\}$ Minimal G-solution: $\Lambda^T(G)=\{G_1,G_2,G_5\}$			
$p^2=(60,20,50,$ $30,50,80,50,30)$	G_2	$B=\{G_2,G_1,G_5,G_9,G_3,G_4,$ $G_7,G_6,G_8,G_{10},G_{12},G_{11}\}$	G_1	$\overline{\varrho}^B_2(p^2\in T)=\overline{r}^B_{1,2}=3.3333$
			G_5	$\overline{r}^B_{5,2}=15$
			G_4	$\overline{r}^B_{4,2}=16.6667$
			G_8	$\overline{r}^B_{8,2}=16.6667$
			G_7	$\overline{r}^B_{7,2}=20$
	G-solution: $\Lambda^*(G)=\{G_1,G_2,G_4,G_5,G_7,G_8\}$ Minimal G-solution: $\Lambda^T(G)=\{G_1,G_2,G_5\}$			
$p^3=(80,20,50,$ $10,65,60,45,35)$	G_2	$B=\{G_2,G_1,G_5,G_3,G_4,G_7,$ $G_8,G_9,G_6,G_{10},G_{11},G_{12}\}$	G_1	$\overline{\varrho}^B_2(p^3\in T)=\overline{r}^B_{1,2}=1$
			G_8	$\overline{r}^B_{8,2}=11$
			G_4	$\overline{r}^B_{4,2}=15$
			G_5	$\overline{r}^B_{5,2}=18.3333$
			G_7	$\overline{r}^B_{7,2}=20$
	G-solution: $\Lambda^*(G)=\{G_1,G_2,G_4,G_5,G_7,G_8\}$ Minimal G-solution: $\Lambda^T(G)=\{G_1,G_2,G_5\}$			
$p^4=(60,20,45,$ $10,50,60,30,30)$	G_1	$B=\{G_1,G_2,G_3,G_4,G_5,G_6,$ $G_8,G_7,G_9,G_{10},G_{11},G_{12}\}$	G_2	$\overline{\varrho}^B_1(p^4\in T)=\overline{r}^B_{2,1}=1.25$
			G_5	$\overline{r}^B_{5,1}=18.3333$
	G-solution: $\Lambda^*(G)=\{G_1,G_2,G_5\}$ Minimal G-solution: $\Lambda^T(G)=\{G_1,G_2,G_5\}$			
$p^5=(70,30,52.5,$ $20,60,70,40,35)$	G_1	$B=\{G_1,G_2,G_3,G_4,G_5,G_6,$ $G_7,G_8,G_9,G_{10},G_{11},G_{12}\}$	G_2	$\overline{\varrho}^B_1(p^5\in T)=\overline{r}^B_{2,1}=2.5$
			G_5	$\overline{r}^B_{5,1}=9.2857$
	G-solution: $\Lambda^*(G)=\{G_1,G_2,G_5\}$ Minimal G-solution: $\Lambda^T(G)=\{G_1,G_2,G_5\}$			
$p^6=(80,40,60,$ $30,70,80,50,40)$	G_1	$B=\{G_1,G_2,G_4,G_5,G_3,G_6,$ $G_7,G_8,G_9,G_{11},G_{10},G_{12}\}$	G_2	$\overline{\varrho}^B_1(p^6\in T)=\overline{r}^B_{2,1}=8$
			G_5	$\overline{r}^B_{5,1}=18.75$
	G-solution: $\Lambda^*(G)=\{G_1,G_2,G_5\}$ Minimal G-solution: $\Lambda^T(G)=\{G_1,G_2,G_5\}$			
$p^7=(80,40,60,$ $30,65,60,30,35)$	G_1	$B=\{G_1,G_4,G_7,G_2,G_5,G_8,$ $G_3,G_6,G_{10},G_9,G_{11},G_{12}\}$	G_2	$\overline{\varrho}^B_1(p^7\in T)=\overline{r}^B_{2,1}=12$
			G_5	$\overline{r}^B_{5,1}=19.2857$
	G-solution: $\Lambda^*(G)=\{G_1,G_2,G_5\}$ Minimal G-solution: $\Lambda^T(G)=\{G_1,G_2,G_5\}$			

Table 2.15 (continuation): *G-solution of problem* $\mathcal{J}3/n=3, a_i \le p_i \le b_i/\sum C_i$ *for different initial vectors* $p \in T$

Initial vector $p^j \in T$	G_s	Set B	$G_k,\ G_s \not\preceq_T G_k$	$\overline{r}^B_{ks}$
1	2	3	4	5
$p^8=(60,20,49,$ $30,69,80,50,40)$	G_2	$B=\{G_2,G_5,G_1,G_9,G_3,G_4,$ $G_7,G_6,G_8,G_{12},G_{10},G_{11}\}$	G_5	$\overline{\varrho}^B_2(p^8\in T)=\overline{r}^B_{5,2}=0$
			G_1	$\overline{r}^B_{1,2}=0.1667$
			G_4	$\overline{r}^B_{4,2}=15$
			G_8	$\overline{r}^B_{8,2}=17.0909$
			G_7	$\overline{r}^B_{7,2}=20$
	G-solution: $\Lambda^*(G)=\{G_1,G_2,G_4,G_5,G_7,G_8\}$ Minimal G-solution: $\Lambda^T(G)=\{G_1,G_2,G_5\}$			
	G_5	$B=\{G_5,G_2,G_1,G_9,G_3,G_4,$ $G_7,G_6,G_8,G_{12},G_{10},G_{11}\}$	G_2	$\overline{\varrho}^B_5(p^8\in T)=\overline{r}^B_{2,5}=0$
			G_1	$\overline{r}^B_{1,5}=0.125$
			G_3	$\overline{r}^B_{3,5}=5.1667$
			G_4	$\overline{r}^B_{4,5}=7.6250$
			G_6	$\overline{r}^B_{6,5}=12$
			G_8	$\overline{r}^B_{8,5}=12.5455$
			G_7	$\overline{r}^B_{7,5}=14$
			G_{10}	$\overline{r}^B_{10,5}=18.8$
	G-solution: $\Lambda^*(G)=\{G_1,G_2,G_3,G_4,G_5,G_6,G_7,G_8,G_{10}\}$ Minimal G-solution: $\Lambda^T(G)=\{G_1,G_2,G_5\}$			
$p^9=(60,20,50,$ $30,70,80,50,30)$	G_1	$B=\{G_1,G_2,G_5,G_9,G_3,G_4,$ $G_7,G_6,G_8,G_{12},G_{11},G_{10}\}$	G_2	$\overline{\varrho}^B_1(p^9\in T)=\overline{r}^B_{2,1}=0$
			G_5	$\overline{r}^B_{5,2}=0$
	G-solution: $\Lambda^*(G)=\{G_1,G_2,G_5\}$ Minimal G-solution: $\Lambda^T(G)=\{G_1,G_2,G_5\}$			
	G_2	$B=\{G_2,G_1,G_5,G_9,G_3,G_4,$ $G_7,G_6,G_8,G_{12},G_{11},G_{10}\}$	G_1	$\overline{\varrho}^B_2(p^9\in T)=\overline{r}^B_{1,2}=0$
			G_5	$\overline{r}^B_{5,2}=0$
			G_4	$\overline{r}^B_{4,2}=15$
			G_8	$\overline{r}^B_{8,2}=17.2727$
			G_7	$\overline{r}^B_{7,2}=20$
	G-solution: $\Lambda^*(G)=\{G_1,G_2,G_4,G_5,G_7,G_8\}$ Minimal G-solution: $\Lambda^T(G)=\{G_1,G_2,G_5\}$			
	G_5	$B=\{G_5,G_1,G_2,G_9,G_3,G_4,$ $G_7,G_6,G_8,G_{12},G_{11},G_{10}\}$	G_1	$\overline{\varrho}^B_5(p^9\in T)=\overline{r}^B_{1,5}=0$
			G_2	$\overline{r}^B_{2,5}=0$
			G_9	$\overline{r}^B_{9,5}=5$
			G_4	$\overline{r}^B_{4,5}=7.1429$
			G_6	$\overline{r}^B_{6,5}=12$
			G_8	$\overline{r}^B_{8,5}=12.7273$
			G_7	$\overline{r}^B_{7,5}=14$
			G_{10}	$\overline{r}^B_{10,5}=19$
	G-solution: $\Lambda^*(G)=\{G_1,G_2,G_4,G_5,G_6,G_7,G_8,G_9,G_{10}\}$ Minimal G-solution: $\Lambda^T(G)=\{G_1,G_2,G_5\}$			

in Table 2.7). The three algorithms EXPL, B&B1 and B&B2 construct the same set $B=\{G_1,G_2,\ldots,G_{12}\}$ *for Example 2.2 as it was constructed with the initial vector* p^0 *(see page 122), but the digraphs from the set B*

form another order according to non-decreasing mean flow time objective function values with different feasible vectors (see column 3 in Table 2.15). We calculate the following sums of the job completion times with the initial vectors $p^1, p^2, \ldots, p^9$ from Table 2.15 (to avoid a confusion, we leave the same subscript of the digraphs indicating the location according to non-decreasing values of function $\sum \mathcal{C}_i$ calculated with the vector p^0 (see page 122)):
$L_5^{p^1} = 482, L_2^{p^1} = 486, L_1^{p^1} = 486, L_9^{p^1} = 512, L_3^{p^1} = 516, L_4^{p^1} = 536, L_7^{p^1} = 566, L_6^{p^1} = 596, L_8^{p^1} = 636, L_{12}^{p^1} = 666, L_{11}^{p^1} = 676, L_{10}^{p^1} = 686;$
$L_2^{p^2} = 450, L_1^{p^2} = 470, L_5^{p^2} = 470, L_9^{p^2} = 500, L_3^{p^2} = 500, L_4^{p^2} = 520, L_7^{p^2} = 550, L_6^{p^2} = 580, L_8^{p^2} = 580, L_{10}^{p^2} = 630, L_{12}^{p^2} = 650, L_{11}^{p^2} = 660;$
$L_2^{p^3} = 455, L_1^{p^3} = 460, L_5^{p^3} = 505, L_3^{p^3} = 510, L_4^{p^3} = 515, L_7^{p^3} = 520, L_8^{p^3} = 520, L_9^{p^3} = 555, L_6^{p^3} = 570, L_{10}^{p^3} = 575, L_{11}^{p^3} = 635, L_{12}^{p^3} = 645;$
$L_1^{p^4} = 365, L_2^{p^4} = 370, L_3^{p^4} = 395, L_4^{p^4} = 415, L_5^{p^4} = 420, L_6^{p^4} = 435, L_8^{p^4} = 435, L_7^{p^4} = 445, L_9^{p^4} = 450, L_{10}^{p^4} = 485, L_{11}^{p^4} = 495, L_{12}^{p^4} = 505;$
$L_1^{p^5} = 438, L_2^{p^5} = 460, L_3^{p^5} = 497.5, L_4^{p^5} = 502.5, L_5^{p^5} = 510, L_6^{p^5} = 532.5, L_7^{p^5} = 538.5, L_8^{p^5} = 547.5, L_9^{p^5} = 560, L_{10}^{p^5} = 612.5, L_{11}^{p^5} = 617.5, L_{12}^{p^5} = 627.5;$
$L_1^{p^6} = 510, L_2^{p^6} = 550, L_4^{p^6} = 590, L_5^{p^6} = 600, L_3^{p^6} = 600, L_6^{p^6} = 630, L_7^{p^6} = 630, L_8^{p^6} = 660, L_9^{p^6} = 670, L_{11}^{p^6} = 740, L_{10}^{p^6} = 740, L_{12}^{p^6} = 750;$
$L_1^{p^7} = 460, L_4^{p^7} = 515, L_7^{p^7} = 520, L_2^{p^7} = 520, L_5^{p^7} = 550, L_8^{p^7} = 560, L_3^{p^7} = 580, L_6^{p^7} = 585, L_{10}^{p^7} = 635, L_9^{p^7} = 640, L_{11}^{p^7} = 660, L_{12}^{p^7} = 670;$
$L_2^{p^8} = 497, L_5^{p^8} = 497, L_1^{p^8} = 498, L_9^{p^8} = 527, L_3^{p^8} = 528, L_4^{p^8} = 558, L_7^{p^8} = 578, L_6^{p^8} = 618, L_8^{p^8} = 646, L_{12}^{p^8} = 698, L_{10}^{p^8} = 706, L_{11}^{p^8} = 708;$
$L_1^{p^9} = 490, L_2^{p^9} = 490, L_5^{p^9} = 490, L_9^{p^9} = 520, L_3^{p^9} = 520, L_4^{p^9} = 540, L_7^{p^9} = 570, L_6^{p^9} = 600, L_8^{p^9} = 640, L_{12}^{p^9} = 670, L_{11}^{p^9} = 680, L_{10}^{p^9} = 690.$

First, we construct a G-solution $\Lambda^*(G)$ by Algorithm $SOL\_\sum \mathcal{C}_i$, and then a minimal G-solution $\Lambda^T(G)$ by Algorithm $MINSOL^*\_\sum \mathcal{C}_i$. In column 2 of Table 2.15, we give the chosen optimal digraph $G_s(p^j)$ for the fixed vector p^j. The set $B := B \setminus \{G_s\}$ ordered according to non-decreasing values $L_u^{p^j}, u \in \{1, 2, \ldots, |B|\}$, is presented in column 4. For digraph $G_k \in B$, we test dominance relation $G_s \preceq_T G_k$ using Lemma 2.5 with the objective function values $\Phi_s^p = L_s^p$ (Step 2 of Algorithm $SOL\_\sum \mathcal{C}_i$). For all digraphs G_k with $G_s \not\preceq_T G_k$ (see column 4), we calculate the value $\overline{r}_{ks}^B$ using formulas (2.48) and (2.54) from Theorem 2.6 for the input vector p^j. Column 5 presents a non-decreasing order of the values $\overline{r}_{ks}^B$ calculated according to (2.54). Due to Theorem 2.6, it follows that the minimal value of $\overline{r}_{ks}^B$ is equal to the relative stability radius $\overline{\varrho}_s^B(p^j \in T)$. An optimal digraph G_s and all digraphs G_k, for which the dominance relation $G_s \preceq_T G_k$ does not hold,

form a G-solution $\Lambda^*(G)$ of problem $\mathcal{J}3/n=3, a_i \le p_i \le b_i/\sum \mathcal{C}_i$. As we see, different choices of the initial vector $p \in T$ may give different G-solutions. The best choice of such a feasible vector is still an open question.

We fix, for example, vector p^4 (vector p^6) to be equal to the vector of given lower bounds a (upper bounds b, respectively), and fix vector p^5 with the components $p_{ij} = \frac{1}{2}(b_{ij} - a_{ij})$. Such a choice gives the following G-solution: $\Lambda^*(G) = \Lambda^T(G) = \{G_1, G_2, G_5\}$. Note that there is no minimal G-solution with a smaller cardinality than set $\Lambda^T(G) = \{G_1, G_2, G_5\}$ has. Moreover, each of the digraphs G_1, G_2 and G_5 is the unique optimal one for some vector $p \in T$, i.e., for example, the following strong dominance relations hold (see column 3 of Table 2.15):

$$
\begin{aligned}
&G_1 \prec_{p^0} G_k, \quad && G_k \in \Lambda(G) \setminus \{G_1\},\\
&G_2 \prec_{p^2} G_k, \quad && G_k \in \Lambda(G) \setminus \{G_2\},\\
&G_5 \prec_{p^1} G_k, \quad && G_k \in \Lambda(G) \setminus \{G_5\}.
\end{aligned}
$$

This means that there is no proper subset of set $\{G_1, G_2, G_5\}$ which is a G-solution of problem $\mathcal{J}3/n = 3, a_i \le p_i \le b_i/\sum \mathcal{C}_i$, and so the G-solution $\Lambda^*(G) = \{G_1, G_2, G_5\}$ is minimal in the sense of inclusion and in the sense of a cardinality equal to 3. As we see, the developed Algorithm $SOL\_\sum \mathcal{C}_i$ may construct some redundant schedules, which are not necessarily in a minimal G-solution $\Lambda^T(G)$ of problem $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$. As it was noted, for the scheduling problem with the makespan criterion (Remark 2.5), not only fixing the initial vector $p \in T$ has a large influence on the resulting G-solution, but also the choice of an optimal digraph G_s for the further calculations, if it is not uniquely determined. In particular, for vectors p^8 and p^9, the optimal schedule is not unique. For example, there are two optimal digraphs $G_2(p^8)$ and $G_5(p^8)$ for vector p^8, therefore, we run Algorithm $SOL\_\sum \mathcal{C}_i$ twice. First, we order the digraphs in set B as follows: $\{G_2, G_5, G_1, G_9, G_3, G_4, G_7, G_6, G_8, G_{12}, G_{10}, G_{11}\}$, and we make all calculations according to Algorithm $SOL\_\sum \mathcal{C}_i$ for the first digraph G_2 in set B. Secondly, we order the digraphs in set B as follows: $\{G_5, G_2, G_1, G_9, G_3, G_4, G_7, G_6, G_8, G_{12}, G_{10}, G_{11}\}$, and we make all calculations for the first digraph G_5. Thus in the first case, the G-solution $\Lambda^*(G)$ consists of six schedules since there are five digraphs G_k for which dominance relation $G_2 \preceq_T G_k$ does not hold. In the second case, the G-solution $\Lambda^*(G)$ consists of nine schedules since there are eight digraphs G_k, $G_5 \not\preceq_T G_k$. Since there are three optimal digraphs for vector p^9 at all, the corresponding cardinalities of the obtained G-solutions $\Lambda^*(G)$ are 3, 6 and 9, respectively (Table 2.15).

As we see from Table 2.15, the covering of polytope T by a minimal number of stability balls (cardinality-minimal covering) is an interesting question for further research. Note that in the general case, a cardinality-minimal covering is a more difficult problem than an inclusion-minimal covering. An algorithm for constructing a cardinality-minimal covering of polytope T is presented in the last chapter of this book but only for a single-machine scheduling problem with minimizing the weighted sum of job completion times.

2.7. Computational Results

The algorithms derived in Section 2.6 were coded in Fortran-77 and were tested on a PC 486 (120 MHz) for an exact G-solution and on a PC 486 (50 MHz) for a heuristic G-solution of problem $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$ and on a PC 486 (133 MHz) for an exact and a heuristic G-solution of problem $\mathcal{J}/a_i \le p_i \le b_i/\mathcal{C}_{max}$. Here the term 'exact G-solution' is used for indicating a set $\Lambda^*(G)$ which satisfies Definition 2.5 in contrast to the 'heuristic G-solution' indicating a set $\Lambda \subset \Lambda(G)$ which generally may not contain an optimal schedule for each vector $p \in T$.

The experimental design was as follows. First, we considered series of instances of problem $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$ with small n and m for which an exact G-solution and the exact minimal G-solution may be calculated within one hour on a PC 486 (120 MHz). After finding upper bounds for such n and m, we started experiments with medium size problems in order to find at least their heuristic G-solutions. The experiments were continued on both computers in order to find upper bounds on n and m for a 'good' heuristic G-solution on a PC 486 (50 MHz) (see Table 2.20), and to increase the problem size for an exact G-solution on a PC 486 (120 MHz) (see Table 2.18).

Table 2.16: *Types of problems considered in the experiments*

	Types of problems	Errors of the processing times				Types of problems	
Exact solutions: sets $B, \Lambda^*(G)$ and $\Lambda^T(G)$	A	5%,	10%,	15%,	20%		
	B	2%,	6%,	8%,	10%	B	Heuristic solution: set B
	C	1%,	3%,	5%,	7%	C	
		1%,	2%,	3%,	4%	D	
		0.1%,	0.2%,	0.3%,	0.4%	E	

We tested the algorithms for the makespan criterion from Section 2.6 with

Table 2.17: *Minimal lower and maximal upper bounds for the processing times*

Errors	Lower bound	Upper bound	The actual processing time p_{ij}^*
20%	$(1-0.2)p_{ij}$	$(1+0.2)p_{ij}$	$0.8p_{ij} \le p_{ij}^* \le 1.2p_{ij}$
15%	$(1-0.15)p_{ij}$	$(1+0.15)p_{ij}$	$0.85p_{ij} \le p_{ij}^* \le 1.15p_{ij}$
10%	$(1-0.1)p_{ij}$	$(1+0.1)p_{ij}$	$0.9p_{ij} \le p_{ij}^* \le 1.1p_{ij}$
8%	$(1-0.08)p_{ij}$	$(1+0.08)p_{ij}$	$0.92p_{ij} \le p_{ij}^* \le 1.08p_{ij}$
7%	$(1-0.07)p_{ij}$	$(1+0.07)p_{ij}$	$0.93p_{ij} \le p_{ij}^* \le 1.07p_{ij}$
6%	$(1-0.06)p_{ij}$	$(1+0.06)p_{ij}$	$0.94p_{ij} \le p_{ij}^* \le 1.06p_{ij}$
5%	$(1-0.05)p_{ij}$	$(1+0.05)p_{ij}$	$0.95p_{ij} \le p_{ij}^* \le 1.05p_{ij}$
4%	$(1-0.04)p_{ij}$	$(1+0.04)p_{ij}$	$0.96p_{ij} \le p_{ij}^* \le 1.04p_{ij}$
3%	$(1-0.03)p_{ij}$	$(1+0.03)p_{ij}$	$0.97p_{ij} \le p_{ij}^* \le 1.03p_{ij}$
2%	$(1-0.02)p_{ij}$	$(1+0.02)p_{ij}$	$0.98p_{ij} \le p_{ij}^* \le 1.02p_{ij}$
1%	$(1-0.01)p_{ij}$	$(1+0.01)p_{ij}$	$0.99p_{ij} \le p_{ij}^* \le 1.01p_{ij}$
0.4%	$(1-0.004)p_{ij}$	$(1+0.004)p_{ij}$	$0.996p_{ij} \le p_{ij}^* \le 1.004p_{ij}$
0.3%	$(1-0.003)p_{ij}$	$(1+0.003)p_{ij}$	$0.997p_{ij} \le p_{ij}^* \le 1.003p_{ij}$
0.2%	$(1-0.002)p_{ij}$	$(1+0.002)p_{ij}$	$0.998p_{ij} \le p_{ij}^* \le 1.002p_{ij}$
0.1%	$(1-0.001)p_{ij}$	$(1+0.001)p_{ij}$	$0.999p_{ij} \le p_{ij}^* \le 1.001p_{ij}$

the corresponding changes for criterion $\mathcal{C}_{max}$ for the same randomly generated test problems. Heuristic G-solutions of problem $\mathcal{J}/a_i \le p_i \le b_i/\mathcal{C}_{max}$ are presented in Table 2.21 and exact G-solutions are given in Table 2.19.

For criterion $\Phi = \sum \mathcal{C}_i$ (Tables 2.19 and 2.21 for criterion $\Phi = \mathcal{C}_{max}$), both Tables 2.18 and 2.20 present computational results only for *classical* job shop problems. (In a *classical* job shop, each job has to be processed by each machine exactly once). So, each randomly generated instance $\mathcal{J}m/n = k, a_i \le p_i \le b_i/\Phi$ has $|Q^J| = mn$ operations and the corresponding mixed graph (Q^J, A^J, E^J) has $(m-1)\cdot n$ arcs and $\binom{n}{2}\cdot m$ edges (note that the latter parameter has the most influence on the running times of our algorithms). For more than 700 classical job shop problems with different combinations of $n \le 10$ and $m \le 8$, we calculated the average number of all feasible schedules λ, the average cardinality $|B|$ of set B, the average cardinality $|\Lambda^*(G)|$ of set $\Lambda^*(G)$, and the average cardinality $|\Lambda^T(G)|$ of set $\Lambda^T(G)$ for both criteria $\sum \mathcal{C}_i$ and $\mathcal{C}_{max}$.

For each combination of n and m under consideration, three types of series (called A, B and C) of instances were considered for the case of an exact G-solution (see Table 2.18 and Table 2.19). Each series consists of 10 instances with randomly generated technological routes. The expected processing times, which form the *input vector* p, are real numbers uniformly distributed in the segment $[10, 100]$. In each instance of types A, B and

Table 2.18: *Exact G-solutions of randomly generated problems* $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$

$n \times m$ type λ	λ' λ^* λ^T	CPU time for Scheme I	Scheme II	Scheme III	λ' λ^* λ^T	CPU time for Scheme IV	λ' λ^* λ^T	CPU time for Scheme V
1	2	3	4	5	6	7	8	9
3×3	5.2	0.6	0.7	0.6	4.8	0.4	4.8	0.2
C	1.9	1.0	1.1	1.0	1.9	0.8	1.9	0.6
91	1.9	1.1	1.3	1.1	1.9	0.9	1.9	0.7
3×3	9.3	0.7	0.8	0.5	6.1	0.2	6.6	0.2
B	2.9	1.2	1.3	1.0	2.7	0.6	2.7	0.6
91	2.6	1.4	1.5	1.2	2.4	0.9	2.4	0.9
3×3	16.5	0.7	0.8	0.9	10.6	0.3	11.3	0.3
A	3.3	1.5	1.7	1.8	2.9	1.0	3.0	1.0
77	3.3	1.7	1.9	2.0	2.9	1.4	3.0	1.4
3×4	3.4	1.9	1.6	1.1	3.0	0.8	3.0	0.3
C	1.7	2.2	1.9	1.8	1.5	1.0	1.5	0.6
262	1.7	2.3	2.0	1.5	1.5	1.1	1.5	0.6
3×4	15.1	2.4	2.6	2.5	9.5	0.9	10.0	0.7
B	3.0	5.9	5.9	5.9	2.5	3.2	2.5	3.0
301	2.7	6.0	6.0	6.1	2.4	3.5	2.4	3.3
3×4	32.5	2.7	3.2	3.4	15.2	0.8	16.8	0.9
A	5.1	12.0	12.1	12.3	3.8	5.6	3.9	6.4
277	4.1	14.1	14.1	14.3	3.1	7.5	3.2	8.3
3×5	4.7	8.3	6.3	5.8	4.1	1.4	4.2	1.0
C	1.5	8.6	6.6	6.2	1.5	1.8	1.5	1.4
605	1.4	8.7	6.8	6.3	1.5	1.9	1.5	1.4
3×5	12.9	10.5	11.6	10.6	9.5	2.2	10.2	1.1
B	3.4	12.8	13.7	12.8	3.3	3.8	3.3	2.9
894	3.1	13.0	14.0	13.0	3.0	4.3	3.0	3.3
3×5	77.6	12.9	20.8	21.3	30.5	2.4	33.0	2.8
A	11.9	63.4	68.9	70.1	8.7	19.4	9.0	27.1
897	10.8	96.5	100.6	102.2	7.7	40.5	8.0	52.8
3×6	9.3	48.8	51.4	34.2	7.1	2.5	7.3	1.5
C	3.5	49.2	52.0	34.8	2.8	2.9	2.8	2.0
1556	2.7	49.5	52.8	35.6	2.1	3.2	2.1	2.3
3×6	21.0	49.1	66.5	58.9	13.9	3.6	14.1	2.3
B	4.6	71.9	88.2	80.6	4.5	19.5	4.5	18.3
1761	4.2	77.2	93.2	85.7	4.1	24.7	4.1	23.6
3×6	65.3	48.3	119.2	139.5	19.2	2.4	23.4	2.7
A	7.6	198.3	262.0	282.7	4.8	51.8	6.0	54.1
1559	7.0	476.0	526.2	548.3	4.5	111.5	5.6	289.4
3×7	5.4	307.6	343.0	310.0	4.1	5.4	4.4	1.6
C	1.5	308.3	343.7	310.7	1.5	5.9	1.5	2.2
4611	1.5	308.4	343.9	310.8	1.5	6.0	1.5	2.3
3×7	38.6	313.8	751.0	769.7	21.2	7.3	23.0	7.2
B	6.6	363.8	797.2	817.0	4.9	39.9	5.1	43.3
4805	5.7	371.1	804.1	824.0	4.3	45.3	4.5	49.3
3×7	156.0	279.7	1319.0	1274.7	21.5	2.4	27.0	3.7
A	19.2	923.4	1934.1	1897.3	9.2	91.4	10.2	108.6
2743	17.9	1032.7	2038.5	2003.1	8.2	123.0	9.2	144.8

Table 2.18 (continuation)**:** *Exact G-solutions of randomly generated problems* $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$

$n \times m$ type λ	λ' λ^* λ^T	CPU time for Scheme I	Scheme II	Scheme III	λ' λ^* λ^T	CPU time for Scheme IV	λ' λ^* λ^T	CPU time for Scheme V
1	2	3	4	5	6	7	8	9
3×8	20.4	2338.2	2106.5	1935.0	15.7	19.7	15.8	7.1
C	4.7	2348.2	2115.2	1938.5	4.0	28.1	4.0	15.5
21923	4.6	2354.6	2121.7	1943.6	3.9	33.2	3.9	20.6
$3 \times 8^*$	28.7	2060.8	4192.0	4018.8	26.0	9.9	27.0	7.1
B	7.4	2172.5	4282.2	4107.4	7.2	57.5	7.2	55.4
8961	7.3	2196.4	4302.5	4120.9	6.3	80.3	6.3	78.2
$3 \times 8^*$	141.5	2054.5	4119.2	3742.9	56.3	9.0	59.2	12.7
A	17.3	3999.9	5938.1	5556.0	19.1	1477.5	21.0	1463.0
8296	16.3	4501.5	6039.7	5811.8	17.6	1952.4	19.6	1963.1
4×3	13.3	19.2	14.2	9.5	11.4	6.1	11.7	2.3
C	3.2	20.3	15.4	10.7	3.2	7.3	3.2	3.5
2907	2.8	21.0	17.4	12.7	2.8	8.1	2.8	4.3
4×3	38.9	17.5	27.8	28.8	29.9	5.4	30.7	3.4
B	6.3	41.4	50.8	51.9	5.5	24.5	5.5	22.6
2217	5.5	45.4	54.8	55.8	4.8	28.3	4.8	26.4
4×3	290.9	28.6	70.7	69.0	110.1	5.9	115.2	8.1
A	31.0	414.0	440.4	442.0	22.5	288.4	22.8	293.9
2990	27.7	688.6	699.1	703.9	19.6	501.9	19.9	509.8
4×4	34.2	303.1	867.3	852.4	23.9	15.8	24.1	5.9
C	7.5	328.5	891.5	876.5	6.5	33.4	6.5	23.6
17159	6.3	330.5	893.0	878.0	5.5	35.1	5.5	25.3
$4 \times 4^*$	88.3	308.2	1501.2	1354.7	52.2	14.4	52.9	9.0
B	16.1	1293.8	2444.9	2297.2	13.5	782.8	13.5	780.5
17763	14.5	1574.0	2771.7	2580.5	12.0	999.6	12.0	997.6
4×4	477.7	319.4	2682.9	2505.1	131.7	15.4	132.0	23.8
A	30.8	3070.1	5355.7	5180.0	24.8	2680.7	24.8	2639.4
16143	30.1	3466.2	5771.4	5594.7	24.0	2905.8	24.0	2879.2

C, all operations are partitioned into four approximately equal parts with different maximal errors of the processing times (see Table 2.16). For an instance of type C, these errors are 1%, 3%, 5% and 7%, for an instance of type B, the errors are 2%, 6%, 8% and 10%, and for an instance of type A, the errors are 5%, 10%, 15% and 20% (see Table 2.16). In particular, the operations of the fourth part of an instance of type A have the most uncertain processing times: If the *input* (expected) processing time is supposed to be equal to p_{ij}, then the lower bound for the actual processing time is equal to $(1-0.2)p_{ij}$ and the upper bound is equal to $(1+0.2)p_{ij}$ (see Table 2.17). On the other hand, the operations of the first part of an instance of type C have the processing times with the smallest error: If the *input* processing time is supposed to be equal to p_{ij}, then the lower bound is equal to $(1-0.01)p_{ij}$ and the upper bound is equal to $(1+0.01)p_{ij}$. Table 2.18 for $\sum \mathcal{C}_i$ (Table 2.19 for $\mathcal{C}_{max}$) presents the results for the following three computational schemes, in

Table 2.19: *Exact G-solutions of randomly generated problems* $\mathcal{J}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$

$n \times m$ type λ	λ' λ^* λ^T	CPU time for Scheme I	Scheme II	Scheme III	λ' λ^* λ^T	CPU time for Scheme IV	λ' λ^* λ^T	CPU time for Scheme V
1	2	3	4	5	6	7	8	9
3×3	4.2	0.17	0.31	0.15	3.4	0.22	3.4	0.10
C	1.3	0.17	0.31	0.16	1.3	0.26	1.3	0.15
91	1.1	0.17	0.31	0.16	1.2	0.26	1.3	0.16
3×3	16.0	0.18	0.28	0.28	7.4	0.14	7.8	0.14
B	2.6	0.18	0.29	0.30	2.0	0.19	2.0	0.19
91	2.4	0.20	0.30	0.30	1.7	0.20	1.6	0.19
3×3	12.8	0.14	0.31	0.30	8.3	0.19	8.5	0.17
A	2.2	0.15	0.32	0.31	2.2	0.24	2.2	0.23
77	2.0	0.15	0.32	0.31	2.0	0.25	2.0	0.23
3×4	7.0	0.76	0.86	0.53	4.0	0.48	4.3	0.22
C	1.5	0.77	0.86	0.54	1.5	0.52	1.5	0.25
262	1.3	0.77	0.86	0.54	1.3	0.53	1.3	0.25
3×4	18.1	0.85	1.0	0.92	10.9	0.55	11.0	0.42
B	3.0	0.90	1.1	0.95	2.5	0.61	2.7	0.49
301	2.2	0.90	1.1	0.96	2.0	0.64	2.0	0.51
3×4	36.4	0.80	1.22	1.14	13.6	0.46	15.0	0.45
A	7.1	0.85	1.27	1.19	5.1	0.54	5.5	0.55
277	5.8	0.89	1.29	1.21	4.2	0.57	4.5	0.59
3×5	8.0	4.05	2.87	2.24	5.4	0.87	5.5	0.40
C	2.2	4.06	2.88	2.25	1.9	0.93	1.9	0.45
605	1.6	4.07	2.89	2.25	1.7	0.94	1.7	0.45
3×5	11.8	4.79	4.48	3.22	7.2	1.33	7.6	0.44
B	3.1	4.83	4.50	3.23	2.4	1.41	2.4	0.52
894	2.3	4.83	4.51	3.23	2.1	1.42	2.1	0.52
3×5	103.7	5.25	9.19	8.80	24.9	1.43	29.2	1.32
A	17.8	5.48	9.41	9.02	8.2	1.61	8.5	1.52
897	13.8	5.62	9.48	9.08	6.6	1.70	6.8	1.63
3×6	7.0	25.98	20.83	11.42	4.1	1.49	4.7	0.57
C	2.4	25.99	20.83	11.43	2.1	1.54	2.3	0.62
1556	2.2	25.99	20.83	11.43	1.9	1.54	2.1	0.63
3×6	21.6	25.73	22.48	19.36	11.5	2.15	12.2	1.08
B	4.3	25.79	22.54	19.42	3.9	2.26	3.9	1.23
1761	3.5	25.83	22.57	19.43	3.1	2.31	3.1	1.29
3×6	67.8	25.37	54.80	54.62	18.1	1.48	19.5	1.30
A	12.5	25.64	55.06	54.89	7.2	1.68	8.0	1.51
1559	8.8	25.78	55.18	55.01	4.8	1.83	5.4	1.68
3×7	7.5	169.47	114.97	99.24	4.0	3.18	4.1	0.60
C	1.7	169.51	114.99	99.26	1.4	3.26	1.4	0.68
4611	1.5	169.51	115.00	99.27	1.3	3.28	1.3	0.70
3×7	42.4	180.42	264.59	261.54	17.8	3.93	19.7	2.29
B	7.1	180.65	264.80	261.74	4.4	4.17	5.3	2.55
4805	5.7	180.71	269.88	261.78	3.9	4.29	4.4	2.69
3×7	90.4	152.22	604.40	523.97	16.6	1.39	19.2	1.66
A	17.7	152.76	604.89	524.45	7.6	1.59	8.6	1.91
2743	13.2	153.00	605.10	524.66	5.6	1.72	6.3	2.06

Table 2.19 (continuation)**:** *Exact G-solutions of randomly generated problems* $\mathcal{J}/a_i \le p_i \le b_i/\mathcal{C}_{max}$

$n \times m$ type λ	λ' λ^* λ^T	CPU time for Scheme I	Scheme II	Scheme III	λ' λ^* λ^T	CPU time for Scheme IV	λ' λ^* λ^T	CPU time for Scheme V
1	2	3	4	5	6	7	8	9
3×8	19.3	1297.43	841.55	795.93	11.3	11.39	11.2	2.66
C	5.2	1297.54	841.65	796.02	3.9	11.53	4.3	2.83
21923	4.6	1297.58	841.68	796.05	3.4	11.60	3.6	2.92
$3 \times 8^*$	32.9	1190.78	1959.74	1938.03	13.5	5.74	14.4	2.10
B	7.2	1191.12	1960.05	1938.35	4.6	6.01	4.9	2.37
8961	4.4	1191.19	1960.11	1938.39	3.2	6.18	3.2	2.55
$3 \times 8^*$	160.3	1161.51	3411.75	3022.93	40.9	5.21	44.8	6.73
A	23.6	1164.83	3414.66	3026.13	12.2	6.39	13.1	7.99
8296	19.0	1166.78	3415.74	3027.31	9.9	7.69	10.2	9.42
4×3	18.1	8.47	6.35	3.93	12.1	3.47	12.4	1.11
C	3.3	8.55	6.41	3.99	3.1	3.56	3.1	1.20
2907	2.1	8.56	6.41	3.99	2.4	3.61	2.1	1.24
4×3	40.9	7.35	10.01	8.67	27.7	3.19	29.3	1.29
B	7.2	7.48	10.10	8.77	5.0	3.34	7.0	1.45
2217	4.2	7.50	10.12	8.78	2.0	3.39	3.8	1.53
4×3	286.4	8.94	23.07	21.36	92.3	3.50	96.3	3.64
A	24.8	9.61	23.64	21.95	17.2	4.08	16.3	4.25
2990	21.3	9.74	23.75	22.03	14.1	4.45	13.3	4.61
4×4	41.8	164.20	199.87	201.76	19.9	7.84	19.9	2.53
C	6.4	164.41	200.05	201.95	3.8	7.99	3.8	2.67
17159	2.4	164.43	200.05	201.96	2.4	7.04	2.4	2.71
$4 \times 4^*$	79.0	169.08	199.25	190.39	27.1	7.52	27.3	2.47
B	14.7	169.58	199.69	190.85	7.2	7.79	6.9	2.73
17763	9.5	169.62	199.72	190.88	4.5	7.94	4.4	2.86
4×4	434.9	164.40	729.63	638.22	104.2	8.27	112.8	9.52
A	43.5	165.76	730.90	639.48	20.5	9.34	25.7	10.74
16143	34.8	166.43	731.42	640.01	15.6	10.17	20.0	11.70

which Algorithms $EXPL$, $B\&B1$ and $B\&B2$ are used with $\Phi_s^p = L_s^p$ (with $\Phi_s^p = l_s^p$, respectively) for the mean flow time criterion (for the makespan criterion).

Scheme I: Algorithm $EXPL \rightarrow$
Algorithm $SOL\_\sum \mathcal{C}_i$ (Algorithm $SOL\_\mathcal{C}_{max}$) $\rightarrow$
Algorithm $MINSOL^*\_\sum \mathcal{C}_i$ (Algorithm $MINSOL^*\_\mathcal{C}_{max}$)

Scheme II: Algorithm $B\&B1 \rightarrow$
Algorithm $SOL\_\sum \mathcal{C}_i$ (Algorithm $SOL\_\mathcal{C}_{max}$) $\rightarrow$
Algorithm $MINSOL^*\_\sum \mathcal{C}_i$ (Algorithm $MINSOL^*\_\mathcal{C}_{max}$)

Scheme III: Algorithm $B\&B2 \rightarrow$
Algorithm $SOL\_\sum \mathcal{C}_i$ (Algorithm $SOL\_\mathcal{C}_{max}$) $\rightarrow$
Algorithm $MINSOL^*\_\sum \mathcal{C}_i$ Algorithm $MINSOL^*\_\mathcal{C}_{max}$

Each of these schemes constructs first a G-solution B, then a G-solution

Table 2.20: *Heuristic G-solutions of randomly generated problems* $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$

n x m; type		k	$B\&B1^*$				$B\&B2^*$			
			λ'			CPU	λ'			CPU
			MIN	AVE	MAX	time	MIN	AVE	MAX	time
1		2	3	4	5	6	7	8	9	10
4 x 4;	B	150	18	52.3	102	34.3	18	52.9	103	21.5
4 x 4;	C	150	4	22.4	53	35.8	4	22.6	53	13.0
4 x 4;	D	150	1	11.1	39	47.1	1	12.2	50	12.2
4 x 5;	C	150	5	35.8	140	102.9	5	37.0	145	41.3
4 x 5;	D	150	2	8.6	18	72.8	2	8.6	18	18.8
4 x 6;	C	150	15	33.8	78	148.5	15	35.6	78	63.1
4 x 6;	D	150	1	7.2	12	100.5	1	7.3	12	22.3
4 x 7;	C	150	6	52.0	134	170.6	5	54.6	136	98.9
4 x 7;	D	150	3	12.8	29	177.3	3	12.9	29	56.4
4 x 8;	C	150	15	54.4	120	416.4	14	58.7	122	292.0
4 x 8;	D	150	7	27.5	57	287.8	7	27.8	55	134.2
4 x 9;	C	150	6	78.0	150	495.1	8	80.3	150	335.6
4 x 9;	D	150	3	22.9	56	458.4	4	22.9	54	156.1
4 x 10;	C	150	25	86.6	150	682.9	24	87.8	150	852.5
4 x 10;	D	150	3	28.5	70	707.9	3	29.1	66	362.5
5 x 3;	C	150	19	62.0	146	85.9	19	62.8	147	65.2
5 x 3;	D	150	2	38.6	150	95.1	2	38.5	150	51.9
5 x 4;	C	150	11	63.1	150	191.8	11	64.3	150	154.0
5 x 4;	D	150	2	23.2	50	182.6	2	23.4	52	106.7
5 x 5;	C	150	63	114.5	150	500.5	62	116.2	150	854.6
5 x 5;	D	150	11	36.9	133	499.0	11	37.4	139	291.1
5 x 5;	E	100	1	1.7	4	366.0	1	1.7	4	86.6
5 x 6;	C	150	15	81.4	150	862.3	16	82.7	150	1220.3
5 x 6;	D	150	7	49.0	89	761.5	7	48.6	88	493.6
5 x 7;	D	150	9	47.9	150	1390.3	9	48.9	150	1642.0
5 x 7;	E	50	1	2.6	7	539.3	1	2.6	7	214.7
5 x 8;	D	100	18	78.5	100	1803.5	18	80.5	100	2446.7
5 x 8;	E	50	1	3.2	6	1054.5	1	3.2	6	328.1
5 x 9;	E	50	1	2.5	6	1531.3	1	2.5	6	653.4
5 x 10;	E	50	1	2.5	5	2071.7	1	2.5	5	617.9
6 x 3;	D	150	19	101.3	150	538.4	19	100.3	150	621.4
6 x 3;	E	50	1	4.2	18	456.8	1	4.2	18	309.8
6 x 4;	D	150	20	99.9	150	1197.8	18	81.3	150	1858.1
6 x 4;	E	100	1	2.3	6	936.7	1	2.3	6	403.6
6 x 5;	D	100	6	90.1	100	1671.0	6	88.1	100	3022.7
6 x 5;	E	50	1	2.8	8	1382.4	1	2.8	8	724.1
6 x 6;	C	50	50	50	50	2389.6	50	50	50	7350.4
6 x 6;	D	50	15	46.5	50	1997.6	15	46.5	50	5252.0
6 x 6;	E	50	1	4.1	12	1997.6	1	3.5	12	1226.2
7 x 3;	D	150	42	122.5	150	1311.9	76	131.8	150	2302.3
7 x 4;	E	100	1	7.1	20	2204.5	1	7.0	24	3608.4
7 x 5;	E	50	1	8.4	39	3074.2	2	15.7	50	6139.9
8 x 3;	E	50	1	4.5	9	1781.5	1	5.1	11	3103.3
9 x 2;	E	100	1	14.1	100	1297.3	1	14.9	100	1958.7
10 x 2;	E	50	2	14.1	50	1651.6	2	9.3	50	2781.4

$\Lambda^*(G)$ by Algorithm $SOL\_\sum \mathcal{C}_i$ ($SOL\_\mathcal{C}_{max}$) and finally a minimal G-solution $\Lambda^T(G)$ by Algorithm $MINSOL^*\_\sum \mathcal{C}_i$ ($MINSOL^*\_\mathcal{C}_{max}$). In Table 2.18

Table 2.21: *Heuristic G-solutions of randomly generated problems* $\mathcal{J}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$

n x m; type		$B\&B1^*$				$B\&B2^*$			
		λ'			CPU	λ'			CPU
		MIN	AVE	MAX	time	MIN	AVE	MAX	time
1		2	3	4	5	6	7	8	9
4 x 4;	B	5	27.2	51	7.29	5	27.3	50	2.46
4 x 4;	C	2	19.9	62	7.59	2	19.9	64	2.51
4 x 4;	D	4	11.8	28	9.70	4	12.5	35	1.81
4 x 5;	C	7	32.1	69	15.97	7	33.1	74	6.58
4 x 5;	D	1	5.2	19	14.10	1	5.4	20	1.56
4 x 6;	C	6	28.0	79	27.53	6	29.7	82	8.69
4 x 6;	D	2	10.6	45	21.91	2	10.7	44	3.82
4 x 7;	C	6	38.7	118	31.93	5	41.3	129	16.43
4 x 7;	D	1	12.4	45	35.68	1	13.4	45	7.85
4 x 8;	C	14	60.6	143	63.53	18	66.0	134	38.91
4 x 8;	D	3	18.3	45	51.04	3	18.5	45	13.59
4 x 9;	C	8	55.1	145	76.46	8	60.6	150	42.13
4 x 9;	D	1	24.5	101	74.88	1	25.8	109	25.55
4 x 10;	C	16	74.9	150	100.86	17	82.3	150	78.79
4 x 10;	D	5	29.6	81	95.54	5	31.8	86	51.35
5 x 3;	C	22	101.0	150	13.06	22	100.4	150	11.64
5 x 3;	D	19	93.1	150	13.15	19	93.3	150	12.18
5 x 4;	C	17	86.1	150	26.86	17	88.4	150	21.81
5 x 4;	D	17	42.1	102	29.31	17	44.1	102	15.63
5 x 5;	B	57	133.1	150	46.29	60	136.4	150	128.36
5 x 5;	C	86	142.2	150	55.33	88	143.3	150	125.10
5 x 5;	D	13	59.7	150	55.67	13	60.7	150	31.70
5 x 5;	E	1	14.6	84	55.43	1	15.1	89	12.20
5 x 6;	C	19	91.3	150	83.44	23	95.9	150	94.73
5 x 6;	D	7	49.8	104	88.35	7	54.0	131	52.46
5 x 7;	D	35	103.9	150	148.23	35	104.6	150	364.76
5 x 7;	E	1	8.2	24	112.03	1	8.2	24	21.18
5 x 8;	D	22	92.1	150	211.49	21	91.9	150	331.82
5 x 8;	E	2	11.3	58	179.53	2	10.9	58	49.70
5 x 9;	E	1	8.9	57	233.89	1	9.1	57	49.22
5 x 10;	E	1	5.9	21	309.72	1	5.9	21	53.87
6 x 3;	D	25	132.7	150	57.70	25	137.5	150	141.25
6 x 3;	E	5	102.9	150	65.60	8	119.7	150	160.52
6 x 4;	D	78	139.7	150	130.70	86	140.5	150	327.15
6 x 4;	E	2	59.2	150	100.75	2	59.6	150	62.80
6 x 5;	D	109	135.8	150	155.49	112	139.3	150	319.53
6 x 5;	E	1	37.0	143	157.26	1	37.0	143	125.21
6 x 6;	C	150	150	150	271.68	150	150	150	1136.78
6 x 6;	D	25	110.0	150	255.49	42	126.3	150	550.78
6 x 6;	E	1	17.6	43	255.48	1	17.7	43	133.24
7 x 3;	D	150	150	150	169.66	110	146.0	150	414.17
7 x 4;	E	4	87.1	150	268.30	12	96.2	150	505.38
7 x 5;	E	5	87.1	150	342.51	2	124.7	150	1057.61
8 x 3;	E	2	134.4	150	234.02	36	138.6	150	585.55
9 x 2;	E	150	150	150	157.68	150	150	150	416.59
10 x 2;	E	150	150	150	209.69	150	150	150	606.73

(Table 2.19), λ denotes the average number of schedules (third row in column 1), λ' the average cardinality of set B, λ^* the average cardinality of set $\Lambda^*(G)$, and λ^T the average cardinality of set $\Lambda^T(G)$ (first, second and third rows in column 2, respectively). Of course, for each instance, $\lambda, \lambda', \lambda^*$ and λ^T are integers, but their average values are real numbers given in Table 2.18, Table 2.19 and the tables below with one decimal place.

The application of Scheme I to Example 2.2 of problem $\mathcal{J}3/n=3, a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$ described in Section 2.4, gives the following sets of schedules. First, Algorithm $EXPL$ constructs the set $\Lambda(G) = \{G_1, G_2, \ldots, G_{22}\}$ of all schedules and set $B \subset \Lambda(G)$ with $|B| = 12$ (see Step 4, where sufficiency from Lemma 2.5 is used). Then, using the G-solution B, Algorithm $SOL\_\sum \mathcal{C}_i$ constructs the set $\Lambda^*(G) = \{G_1, G_2, G_5\}$ which is also a G-solution. Finally, Algorithm $MINSOL\_\sum \mathcal{C}_i$ shows that G-solution $\Lambda^*(G)$ is minimal: $\Lambda^T(G) = \Lambda^*(G)$. For Example 2.2, we have $\lambda = 22, \lambda' = 12$ and $\lambda^* = \lambda^T = 3$. The average CPU time (in seconds) for constructing set B, set $\Lambda^*(G)$ and set $\Lambda^T(G)$ (first, second and third rows) are presented in columns 3, 4 and 5 for Schemes I, II and III, respectively. As follows from Table 2.18 and Table 2.19, in most cases for both criteria $\sum \mathcal{C}_i$ and $\mathcal{C}_{max}$, Scheme III based on Algorithm B&B2 is the best for the problems of type C, while Scheme I based on Algorithm $EXPL$ is the best for the problems of types A and B. As it was mentioned, Steps 8 and 9 for the branch-and-bound algorithm are not so fast as Step 1 of Algorithm $EXPL$. Moreover, due to a large uncertainty of the input vector p for problems A and B, Algorithms $B\&B1$ and $B\&B2$ have to construct a lot of intermediate digraphs $G_{(t)}$ in the branching tree which are not in the set $\Lambda(G)$. Unfortunately, an exact minimal G-solution was obtained within 1.5 hours by the worst of the Schemes I, II or III only for some combinations of n and m with $n \leq 4$ and $m \leq 8$ and an exact G-solution was not obtained by Scheme I for some combinations of n and m for the reason 'not enough memory' or 'limit of time' (such series are marked in the first column of Table 2.18 and Table 2.19 by an asterisk).

To solve problems with a larger size, we were forced to consider restricted variants of the branch-and-bound algorithms: Algorithm $B\&B1^*$ (Algorithm $B\&B2^*$) denotes Algorithm $B\&B1$ (Algorithm $B\&B2$, respectively) without Steps 8 and 9. In general, such modifications do not guarantee to obtain a G-solution B, but they are essentially faster. Fortunately, for almost all problems presented in Table 2.18 and Table 2.19, the restricted variants of the branch-and-bound algorithms still give a G-solution, i.e., for each $p \in T$ the set B constructed contains an optimal schedule. The main reason for this computational result is that Steps 8 and 9 often generate only schedules

which are dominated by other ones. Therefore, it is possible to exclude these schedules due to Theorem 2.4 (see page 124). Columns 6 – 9 of Table 2.18 (of Table 2.19) present computational results on a PC 486 (120 MHz) (on a PC 486 (133 MHz), respectively) for the following two computational schemes.

Scheme IV: Algorithm $B\&B1^*$ with $\Phi_s^p = L_s^p$ (with $\Phi_s^p = l_s^p$) $\rightarrow$
Algorithm $SOL\_\sum \mathcal{C}_i$ (Algorithm $SOL\_\mathcal{C}_{max}$) $\rightarrow$
Algorithm $MINSOL^*\_\sum \mathcal{C}_i$ (Algorithm $MINSOL^*\_\mathcal{C}_{max}$).

Scheme V: Algorithm $B\&B2^*$ with $\Phi_s^p = L_s^p$ (with $\Phi_s^p = l_s^p$) $\rightarrow$
Algorithm $SOL\_\sum \mathcal{C}_i$ (Algorithm $SOL\_\mathcal{C}_{max}$) $\rightarrow$
Algorithm $MINSOL^*\_\sum \mathcal{C}_i$ (Algorithm $MINSOL^*\_\mathcal{C}_{max}$).

More precisely, column 6 presents the average approximate values λ' (first row), λ^* (second row) and λ^T (third row) calculated by Algorithm $B\&B1^*$. Column 7 presents the average running times for constructing approximations of the sets $B, \Lambda^*(G)$ and $\Lambda^T(G)$ by Algorithm $B\&B1^*$. Similarly, column 8 presents the average approximate values λ', λ^* and λ^T calculated by Algorithm $B\&B1^*$. Column 9 presents the average running times for constructing approximations of the sets $B, \Lambda^*(G)$ and $\Lambda^T(G)$ by Algorithm $B\&B2^*$. From Table 2.18 for criterion $\sum \mathcal{C}_i$ and Table 2.19 for criterion $\mathcal{C}_{max}$, it follows that Algorithm $B\&B2^*$ in Scheme V is often faster than Algorithm $B\&B1^*$ in Scheme IV. There exist only 3 series of problems (all of type A), for which Algorithm $B\&B1^*$ is faster, on average, than Algorithm $B\&B2^*$. Only for some series of type A, Algorithm $B\&B1^*$ is, on average, faster than Algorithm $B\&B2^*$. Note also that Algorithm $B\&B2^*$ gives more often an exact G-solution than Algorithm $B\&B1^*$.

As it follows from Table 2.18 and Table 2.19, even the heuristic Schemes IV and V require rather large running times. So, for larger problem sizes, we used only Algorithms $B\&B1^*$ and $B\&B2^*$ for constructing the sets B heuristically, i.e., without a guarantee that the constructed set B is indeed a G-solution. Obviously, the cardinality of a G-solution increases not only with increasing the size of the problem (which in turn increases the running time), but also with increasing the uncertainty of the numerical input data. Therefore, to reduce the cardinality of a G-solution, we consider along with instances of types A, B, and C also problems of the following two types D and E with smaller errors of the given processing times, namely: Problems of type D with the errors of the processing times equal to 1 %, 2 %, 3 % and 4 %, and problems of type E with the errors of the processing times equal to 0.1 %, 0.2 %, 0.3 % and 0.4 % (see Table 2.16). Heuristic G-solutions are represented in Table 2.20 for problems $\mathcal{J}/a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$ and in Table 2.21

for problems $\mathcal{J}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ with the same structural and numerical input data. Next, we describe the design of Table 2.20. The values of k used for criterion $\sum \mathcal{C}_i$ are given in column 2. Columns 3–6 (and columns 7–10) present computational results for Algorithm $B\&B1^*$ (and Algorithm $B\&B2^*$, respectively). Column 3 (column 7) gives the minimal value of the cardinality λ' of the set B constructed, column 4 (column 8) the average value of λ', and column 5 (column 9) the maximal value of λ'. The average CPU times are given in column 6 for Algorithm $B\&B1^*$ and in column 10 for Algorithm $B\&B2^*$. Table 2.21 has a similar design with the exception of the column with the values of k used. For criterion $\mathcal{C}_{max}$, we set $k = 150$ for all computational results presented for problems $\mathcal{J}/a_i \leq p_i \leq b_i/\mathcal{C}_{max}$. As follows from Table 2.20 and Table 2.21, if the problems have a small size, Algorithm $B\&B2^*$ is superior to Algorithm $B\&B1^*$ (both in running time and the quality of the G-solution constructed). However, if the number of potentially optimal schedules is large (due to a large problem size or due to a large uncertainty of the numerical input data), then Algorithm $B\&B1^*$ has a smaller running time, however, the quality of the G-solution constructed by Algorithm $B\&B2^*$ remains still better. Moreover, the value of k has a large influence on the quality and the running time of Algorithm $B\&B1^*$ in contrast to Algorithm $B\&B2^*$ which is independent of k. In principle, we use the parameter k in Algorithm $B\&B2^*$ mainly to have the same conditions for the comparison with Algorithm $B\&B1^*$.

2.8. Comments and References

In spite of obvious practical importance, the literature on stability and sensitivity analysis in sequencing and scheduling is not numerous. Outside the approach considered in this book, one can mention the papers [24, 26, 88, 89, 92, 107, 108, 123, 155, 162, 184, 186, 187, 214, 221, 227, 228, 230, 231, 234, 255, 259, 260, 267, 363, 370], where close or different measures for stability and sensitivity of an optimal (approximate) sequence or schedule were introduced and studied. Note that the stability notions in the general OR literature have been introduced almost simultaneously with the development of the first methods for solving mathematical programming problems. However, a direct transformation of most stability and sensitivity results obtained in non-integer mathematical programming to discrete optimization problems (in particular, to scheduling problems) has provided only simple conclusions. The first papers, presenting specific stability concepts for scheduling problems, were published since the late 1980s.

Since a schedule contains both *combinatorial* and *temporal structures*, scheduling problems provide sufficiently wide issues for a stability and sensitivity analysis (see [155]). The combinatorial structure of a schedule includes which machine is chosen to process an operation and the sequence in which the set of jobs are processed in the schedule. The temporal structure of a schedule characterizes when a job is processed, which specifies some parts of a schedule as local in relation to others. Next, we survey some known results for stability and sensitivity analysis in sequencing and scheduling.

A study of sensitivity analysis for single-stage scheduling problems was initiated by Hall and Posner [155]. In particular, they developed a sensitivity analysis for problems $1//\sum w_i\mathcal{C}_i$, $P//\sum \mathcal{C}_i$, $1//\mathcal{L}_{max}$ and other *list scheduling problems*, i.e., those that are optimally solvable by a sequence of jobs generated by a specific priority rule. In [155], it was shown that the modified instance (when a concrete change of one processing time or one job weight is given) of some polynomially solvable problems (among them problems $1//\sum w_i\mathcal{C}_i$, $1/r_i, pmtn/\sum \mathcal{C}_i$, and $1/r_i, d_i, p_i = 1/\sum w_i\mathcal{C}_i$) can be solved more efficiently using the optimal schedule known for the original problem instance. Based on observations for integer mathematical programming, Geffrion and Nauss [131] provided several answers to the question: What can be said about the stability of an optimal schedule before it is found? In [155], the following results from [131] were represented using scheduling terminology.

If the set of feasible schedules is reduced and schedule s is still feasible, then schedule s is still optimal. Due to this quite simple claim, for a scheduling problem with any regular criterion, the following two parameter changes do not alter an optimal schedule s. Namely, an increase in the release time, which is not later than the time when the job is started with the processing in schedule s, and a decrease in the deadline, which is not earlier than the time when the processing of the job is completed in schedule s.

If job J_i is completed at its lower bound in schedule s and the weight of job J_i increases, then schedule s is still optimal. Due to this claim, for a scheduling problem with any regular criterion, the following two parameter changes do not alter an optimal schedule s: An increase in the weight of a job that is processed first in schedule s and an increase in the weight of a job the processing of which starts in schedule s at its release time.

If job J_i is completed at its upper bound in schedule s and the weight of job J_i decreases, then schedule s is still optimal. Due to this claim, for a scheduling problem with any regular criterion, the following two parameter changes do not alter an optimal schedule s: A decrease in the weight of a

job the processing of which is completed at its deadline in schedule s and a decrease in the weight of a job that is processed last in a single machine scheduling problem.

Hall and Posner [155] described also other sufficient conditions under which an increase in the set of feasible schedules does not change an optimal non-preemptive schedule for a regular criterion and without given precedence constraints. *If the deadline d_k of job J_k is increased when there is a consecutive idle time of at least* $\max_{J_i \in J}\{P_k : C_i(s) > C_k(s)\}$ *in the closed interval* $[C_k(s), d_k]$, *then the originally optimal schedule s remains optimal.* For a bottleneck criterion (like minimizing the maximum lateness), the following sufficient condition was given in [155] as well: *If an increase in the operation processing time does not create a new bottleneck or increase the current bottleneck, then the originally optimal schedule remains optimal.*

In [186], the sensitivity of a heuristic algorithm with respect to the variation of the processing time of one job was investigated. In [267], results for the traveling salesman problem were used for a single machine scheduling problem with minimizing total tardiness [204].

The stability of an optimal line balance for a fixed number of stations (and for a fixed cycle time) was investigated in [317, 320, 321, 322]. The assembly line balancing problem with variable operation times has been considered in [130, 147, 208, 287, 361]. In [130, 361], fuzzy set theory was used to represent the uncertainty of the operation times. Genetic algorithms were used either to minimize the total operation time for each station [130] or to minimize the efficiency of the fuzzy line balance [361]. In [208], the entire decision process has been decomposed into two parts: The deterministic problem and the stochastic problem. For the former problem, integer mathematical programming is used to minimize the number of stations. For the latter problem, which takes into account the variations of the operation times over different products, queuing network analysis is used to determine the necessary capacity of the material-handling system. In [287], the dynamic programming method and the branch-and-bound method have been used to minimize the total labor cost and the expected incompletion cost arising from operations not completed within the cycle time. Branch-and-bound algorithms were developed in [55, 272, 287, 290], integer mathematical programming algorithms in [96, 208, 272].

Single and multi-machine problems are analyzed considering criteria depending on the completion times and the due dates. In particular, Chu and Gordon [83], Gordon and Tarasevich [141] considered a single machine problem including both the due date assignment and the scheduling deci-

sions. In [83], it was assumed that the due dates are proportional to the job processing times. In [83, 141], the objective was to minimize the weighted earliness-tardiness and the penalty related to the size of the dates with respect to the processing times. Lahlou and Dauzere-Peres [199] considered a single machine scheduling problem provided that the processing time of a job depends on the time at which the job is started (the planning horizon is divided into time windows and with each one a coefficient is associated that is used to determine the actual processing time of a job starting in this time window). It is proven that such a problem with the makespan criterion is NP-hard even with two time windows being given.

The complexity of a sensitivity and stability analysis of a discrete optimization problem was studied in [63, 135, 137, 163, 164, 261, 276, 309, 312]. In particular, Ramaswamy and Chakravarti [276] have shown that the problem of determining the *arc tolerance* for a discrete optimization problem is as hard as the original problem itself (the arc tolerance is the maximum change, i.e., increase or decrease, of a single weight, which does not destroy the optimality of a solution). This means that in the case of the traveling salesman problem, the arc tolerance problem is NP-hard even if an optimal tour is given. Gordeev [135] proved the NP-hardness of the problem of calculating the stability radius for the polynomially solvable shortest path problem in a digraph without negative circuits. In [190], the definitions of the stability radius of an ϵ-approximate solution for the Boolean problem of minimizing a linear form was introduced. Sotskov [309, 312] has shown that the stability radius of an ϵ-approximate solution may be calculated in polynomial time if the number of unstable components grows rather slowly, namely as $O(\log_2 N)$, where N is the number of cities in the traveling salesman problem. There has been a lot of published works on a sensitivity analysis of specific discrete optimization problems such as the minimum spanning tree, the shortest path, the minimum Hamiltonian path and the traveling salesman problem with min-sum or min-max objectives [38, 154, 227, 228, 230, 277, 298, 356]. A stability analysis for the minimum spanning tree problem was considered by Gusfield [154] and Tarjan [356]. Shier and Witzgall [298] proposed algorithms for finding the exact tolerances for the shortest path tree problem. The algorithms of Gusfield [154] and Tarjan [356] for a stability analysis of the minimum spanning tree may also be used with minor modifications for the shortest path tree problem as pointed out by the authors in their respective papers. Libura [227, 228] developed a stability analysis for the minimum Hamiltonian path and the traveling salesman problems. Libura et al. [230] argued that it is rather convenient

from a computational point of view to use a set of k shortest tours when applying a stability analysis to the symmetric traveling salesman problem.

It is clear that a learning (forgetting) effect may decrease (increase) the processing time of a job. Alidaee and Womer [8] cited a lot of examples from practice, where the processing time of a job may increase or decrease with its starting time in the production duration. In [34, 79, 128, 140, 174, 249, 250, 251], single-stage processing systems have been considered provided that the processing time of a job is dependent on its position in a schedule. In particular, Gawiejnowicz [128] considered a learning effect in a single machine scheduling problem with the $\mathcal{C}_{max}$ criterion. Biskup [34] has shown that single machine problems of minimizing the deviation from a common due date and of minimizing total flow time remain polynomially solvable even if a learning effect being taken into account. Mosheiov [249, 250] provided a polynomial solution for some multi-criteria single machine problems and for the minimum flow time scheduling problem on parallel identical machines. Mosheiov and Sidney [251] extended the learning effect to the case when the jobs are associated with a specific learning rate. Cheng and Wang [79] studied a piecewise-linear model of a learning effect for the case of a single machine problem with minimizing maximum lateness.

To model scheduling in an uncertain environment, a two-person non-zero sum game was introduced by Chrysslouris et al. [82], where the decision-maker was considered as player 1 and the 'nature' as player 2. An agent-based negotiation approach was developed in [366] to integrate planning and scheduling. The negotiation protocol has enhanced the processing system robustness over the previous works. The makespan and total flow time problems were experimentally compared with those of other search techniques. The stability of an optimal situation in a finite cooperative game with a parametric concept of equilibrium was investigated in [56, 57].

In this book, it was assumed that multiple processing time changes are independent. However, in some real world scheduling problems multiple parameter changes are related. This may be due to the relationships between the numerical parameters, such as the correlation between the value and work required. Hall and Posner [155] demonstrated for some single-stage scheduling problems that a sensitivity analysis may depend on the position of the jobs with changed parameters. The results presented in this chapter were originally published in [44, 200, 201, 202, 203, 311, 318, 327, 329, 334, 339, 340, 341]. In Section 1.5, we developed algorithms for calculating the stability radii $\widehat{\varrho}_s(p)$ and $\overline{\varrho}_s(p)$ on the basis of the formulas from [44, 311, 339]. The computational results were originally published in [202, 334, 340].

Table 2.22: *Common notations*

Symbols	Description
$\mathcal{J}$	Symbol for a job shop (in the α field of the three-field notation $\alpha/\beta/\gamma$)
$\mathcal{F}$	Symbol for a flow shop
$\mathcal{O}$	Symbol for an open shop
$\mathcal{G}$	Symbol for a general shop
M	Set of machines: $M = \{M_1, M_2, \ldots, M_m\}$
J	Set of jobs: $J = \{J_1, J_2, \ldots, J_n\}$
$C_i(s)$	Completion time of job $J_i \in J$ in schedule s
Q^J	Set of all operations for processing all jobs J: $Q^J = \{Q_{ik} \ : \ J_i \in J, \ k = 1, 2, \ldots, n_i\}$
$\Phi(C_1, C_2, \ldots, C_n)$	Objective function of the job completion times where $C_i = C_i(s)$
Φ_k^p	Value of the objective function calculated for digraph $G_k \in \Lambda(G)$ (schedule k) with processing times given by vector $p \in R_+^q$
Φ	Regular criterion
$\mathcal{C}_{max} = \max_{i=1}^n C_i$	Criterion of minimizing the maximum flow time (makespan)
$\sum \mathcal{C}_i = \sum_{i=1}^n C_i$	Criterion of minimizing mean flow time
$C_i(s)$	Completion time of job $J_i \in J$ in schedule s
$B := B + C$	In this case sign := means that B is substituted by $B + C$
$\|B\|$	Cardinality of set B (number of elements in set B if it is finite)
$\|b\|$	Absolute value of the real number $b \in R^1$, i.e., $\|b\| = b$, if $b \in R_+^1$, $\|b\| = -b$, if $b \in R^1 \setminus R_+^1$
R^n	Space of n-dimensional real vectors with the maximum metric
R_+^n	Space of non-negative real vectors with the maximum metric
Q_+^n	Set of all n-dimensional (strictly) positive rational vectors
$Q_{\geq 0}^n$	Set of all n-dimensional non-negative rational vectors
N^n	Set of all n-dimensional natural vectors
$d(p, p')$	Distance between vector $p \in R^n$ and vector $p' \in R^n$: $d(p, p') = \max_{i=1}^n \|p_i - p'_i\|$
$\lceil a \rceil$	The smallest integer greater than or equal to the real number a
$[\mu] + [\nu]$	'Symmetric difference' of set $[\mu]$ and set $[\nu]$: $[\mu] + [\nu] = [\mu] \cup [\nu] \setminus [\mu] \cap [\nu]$
$G = (Q, A, E)$	Mixed graph defining the structural input data
q	Number of operations: $q = \|Q^J\| = \sum_{i=1}^n n_i = \sum_{k=1}^m \|Q_k^J\|$
$G_k = (Q, A \cup E_k, \emptyset)$	Acyclic digraph generated from the mixed graph G
E_k	Signature of schedule $k \in S$, $G_k \in \Lambda(G)$
$G(p) = (Q(p), A, E)$	Weighted mixed graph with the vector p of job processing times
$G_k(p) = (Q(p), A \cup E_k, \emptyset)$	Acyclic weighted digraph
G_k^T	Minimal subgraph of G_k containing all dominant paths with respect to polytope T
$\Lambda(G) = \{G_1, G_2, \ldots, G_\lambda\}$	Set of acyclic digraphs generated from the mixed graph G
$S = \{1, 2, \ldots, \lambda\}$	Set of semiactive schedules
λ	Number of semiactive schedules
$S^\Phi(p)$	Set of optimal semiactive schedules with respect to criterion Φ
$O_\varrho(p)$	Stability ball of an optimal digraph with radius ϱ and center $p \in R_+^q$
$\varrho_s(p)$	Stability radius of an optimal digraph G_s for a regular criterion
$\widehat{\varrho}_s(p)$	Stability radius of an optimal digraph G_s for the makespan criterion
$\overline{\varrho}_s(p)$	Stability radius of optimal digraph G_s for the mean flow time criterion
$[\mu]$	Set of vertices (operations) which are contained in path μ
$l^p(\mu)$	Weight of path μ in the digraph with processing times $p \in R_+^q$
$\tilde{H}_k^i$	Set of paths in digraph G_k ending in the last operation of job J_i
H_k^i	Set of dominant paths in $\tilde{H}_k^i$

Chapter 3

Two-Machine Flow Shop and Job Shop

Punctuality is a thief of time
Oscar Wilde

In the first two sections of this chapter, the flow shop problem with n jobs processed on two machines is addressed, where interval job processing times are given before scheduling. It is assumed that the probability distributions of the random processing times are unknown, and only their lower and upper bounds are given before scheduling. In such a case, there may not exist a unique schedule that remains optimal for all possible realizations of the job processing times, and so we look for a minimal set of schedules (permutations) which dominates all feasible schedules for the given objective function. We obtain necessary and sufficient conditions for the case when it is possible to fix the order of two jobs for the makespan criterion (in spite of the uncertainty of the numerical input data). We characterize the easiest case of the two-machine flow shop problem with interval processing times, i.e., we prove necessary and sufficient conditions for the existence of a single schedule which is dominant for the makespan criterion. On the other hand, we describe the set of the hardest problems such that any semiactive schedule may be a unique optimal one for some possible realizations of the job processing times. Along with the off-line scheduling problem, we consider the on-line problem, where a part of the schedule is already realized. We show how to use the additional information available for the on-line problem to obtain a better solution than that constructed for the off-line version of the problem. All the conditions proven may be tested in polynomial time.

3.1. Flow Shop with Interval Processing Times

We address the flow shop problem when it is impossible to obtain reliable probability distributions for the random processing times. As such, a schedule obtained by assuming certain probability distributions may not be close to an optimal schedule in a practical realization of the process.

Let two machines $M = \{M_1, M_2\}$ be given to process $n \geq 2$ jobs $J = \{J_1, J_2, ..., J_n\}$ that follow the same machine route (flow shop), i.e., each job $J_i \in J$ has to be processed by machine M_1 and then by machine M_2 without preemption on each machine (Condition 4 on page 12). All the n jobs are available to be processed from time 0. $C_i(\pi)$ denotes the completion time of job $J_i \in J$ in schedule π. $\mathcal{C}_{max}$ denotes the minimization of the schedule length $C_{max}(\pi)$:

$$\mathcal{C}_{max} = \min_{\pi \in S} C_{max}(\pi) = \min_{\pi \in S} \{\max\{C_i(\pi) \ : \ J_i \in J\}\},$$

where S denotes the set of all semiactive schedules. The set of semiactive schedules has the cardinality $|S| = (n!)^2$. In contrast to the conventional two-machine flow shop problem $\mathcal{F}2//\mathcal{C}_{max}$ [177], we assume that the processing time p_{ij} of job $J_i \in J$ on machine $M_j \in M$ is unknown before scheduling. In the realization of the process, p_{ij} may take any real value in the (closed) interval $[p_{ij}^L, \ p_{ij}^U]$, where the lower bound p_{ij}^L and the upper bound p_{ij}^U are fixed, but the probability distributions of the random processing times between these bounds are unknown. Such a two-machine minimum-length flow shop problem with *interval* processing times is denoted by $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$. Problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$ is called an *uncertain* problem in contrast to problem $\mathcal{F}2//\mathcal{C}_{max}$ which is called a *deterministic* problem.

Let $S^\pi = \{\pi_1, \pi_2, \ldots, \pi_{n!}\}$ be the set of all permutations of the n jobs from set J: $\pi_k = (J_{k_1}, J_{k_2}, \ldots, J_{k_n}), \ k \in \{1, 2, \ldots, n!\}, \ \{k_1, k_2, \ldots, k_n\} = \{1, 2, \ldots, n\}$. The set S^π defines the *permutation* schedules that are *dominant* for the deterministic problem $\mathcal{F}2//\mathcal{C}_{max}$. In other words, there exists at least one optimal schedule with the same sequence of jobs on both machine M_1 and M_2, i.e., the set S^π of permutation schedules *dominates* the set S of semiactive schedules for the deterministic problem $\mathcal{F}2//\mathcal{C}_{max}$. Let the set of all feasible vectors $p = (p_{1,1}, p_{1,2}, \ldots, p_{n1}, p_{n2})$ of the job processing times be denoted by T:

$$T = \{p \ : \ p_{ij}^L \leq p_{ij} \leq p_{ij}^U, \ J_i \in J, \ M_j \in M\}. \tag{3.1}$$

For a fixed vector $p \in T$, the uncertain problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$

turns into the deterministic problem $\mathcal{F}2//\mathcal{C}_{max}$ associated with the vector p of job processing times, and so for each vector $p \in T$, it is sufficient to look for an optimal schedule among the permutation schedules. Thus, it is sufficient to examine the set of permutation schedules for solving the uncertain problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$ as well. The set of the permutation schedules has the cardinality $|S^\pi| = n!$. Next, we shall restrict further the set of feasible permutations that are sufficient to be examined for solving problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$.

Since preemption in the processing of job $J_i \in J$ on machine $M_j \in M$ is not allowed, each permutation $\pi_k \in S^\pi$ defines a unique set of the earliest completion times $C_1(\pi_k), C_2(\pi_k), \ldots, C_n(\pi_k)$, which in turn defines a semiactive schedule for the deterministic problem $\mathcal{F}2//\mathcal{C}_{max}$. In what follows, we shall not distinguish a permutation $\pi_k \in S^\pi$ from the semiactive schedule defined by this permutation. We need such an agreement for the uncertain problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$ in which the processing times are not fixed before scheduling, and as a result, it is impossible to calculate a priori (before the realization of the process) all completion times for permutation $\pi_k \in S$. For the uncertain problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$, we use the notation $C_i(\pi_k, p)$ for the completion time of job $J_i \in J$, and the notation $C_{max}(\pi_k, p) = \max\{C_i(\pi_k, p) \ : \ J_i \in J\}$ for the makespan value, provided that a concrete vector $p \in T$ of the job processing times is just treated.

Johnson [177] proved that it takes $O(n \log_2 n)$ time to construct an optimal permutation $\pi_i = (J_{i_1}, J_{i_2}, \ldots, J_{i_n}) \in S^\pi$ for the deterministic problem $\mathcal{F}2//\mathcal{C}_{max}$. Such an optimal permutation π_i satisfies the condition

$$\min\{p_{i_k 1}, p_{i_m 2}\} \leq \min\{p_{i_m 1}, p_{i_k 2}\} \tag{3.2}$$

with $1 \leq k < m \leq n$. In what follows, each permutation π_i satisfying condition (3.2) will be called a Johnson permutation. The following simple algorithm allows us to construct at least one Johnson permutation.

Johnson's algorithm. *Partition the set of jobs into two subsets $J = N_1 \cup N_2$ with N_1 containing the jobs with $p_{i1} \leq p_{i2}$ and N_2 the jobs with $p_{i1} \geq p_{i2}$. (A job with the equality $p_{i1} = p_{i2}$ may be either in set N_1 or in set N_2.) Now construct an optimal schedule, where the jobs from set N_1 are processed first, and they are processed in a non-decreasing order of p_{i1}. The jobs from set N_2 follow the jobs from set N_1 in a non-increasing order of p_{i2}.*

Obviously, if all the processing times are different ($p_{ij} \neq p_{uv}$ if $i \neq u$ or $j \neq v$), then Johnson algorithm generates a single Johnson permutation for the deterministic problem $\mathcal{F}2//\mathcal{C}_{max}$. Otherwise, Johnson's algorithm may

generate several permutations. Each permutation π_j generated by Johnson's algorithm is a Johnson permutation, i.e., π_j satisfies condition (3.2) with $i = j$. However, the opposite claim is not correct: There may exist a Johnson permutation that cannot be constructed by Johnson's algorithm. We present this as the following remark.

Remark 3.1 Let set Π_c (set Π_a, respectively) be the set of all Johnson permutations for problem $\mathcal{F}2//\mathcal{C}_{max}$ (the set of all permutations constructed using Johnson's algorithm). Then inclusion $\Pi_a \subseteq \Pi_c$ must hold, and there are instances of problem $\mathcal{F}2//\mathcal{C}_{max}$ such that set Π_a is a proper subset of set Π_c, i.e., it may happen that $\Pi_c \setminus \Pi_a \neq \emptyset$.

The following example illustrates Remark 3.1.

Example 3.1 *We consider problem $\mathcal{F}2//\mathcal{C}_{max}$ with the job set $J = \{J_1, J_2, J_3, J_4\}$ and the processing times defined by vector $p = (3, 4, 4, 5, 4, 3, 2, 2)$. It is easy to see that there are four Johnson permutations $\pi_1 = (J_4, J_1, J_2, J_3)$, $\pi_2 = (J_1, J_4, J_2, J_3)$, $\pi_3 = (J_1, J_2, J_4, J_3)$ and $\pi_4 = (J_1, J_2, J_3, J_4)$, i.e., each permutation from set $\Pi_c = \{\pi_1, \pi_2, \pi_3, \pi_4\}$ satisfies condition (3.2). However, Johnson's algorithm can generate only two permutations, namely, π_1 and π_4. Thus, we obtain $\{\pi_1, \pi_4\} = \Pi_a \subset \Pi_c = \{\pi_1, \pi_2, \pi_3, \pi_4\}$.*

It is also useful to give the following remark.

Remark 3.2 For problem $\mathcal{F}2//\mathcal{C}_{max}$, some optimal schedule may be defined by a permutation from set S^π that does not satisfy condition (3.2).

In other words, inequalities (3.2) are *sufficient* for the optimality of a permutation $\pi_i \in S^\pi$ but they are not *necessary* for the optimality of a permutation. In particular, for problem $\mathcal{F}2//\mathcal{C}_{max}$, more general sufficient conditions for the optimality of a permutation were proven in [48, 232], and so larger sets of optimal permutations can often be constructed. Nevertheless, in this section, the set of optimal permutations for the deterministic problem $\mathcal{F}2//\mathcal{C}_{max}$ associated with a vector $p \in T$ of processing times will be restricted by Johnson ones. We argue as follows. The aim of this section is to construct a minimal set of dominant schedules for the uncertain problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$, and so we can remove all the redundant permutations $\pi_k \in S^\pi$. Specifically, if there is no vector $p \in T$ such that permutation $\pi_k \in S^\pi$ is a Johnson one for the deterministic problem $\mathcal{F}2//\mathcal{C}_{max}$ associated with the vector p of processing times, then we can remove permutation π_k from further considerations. Summarizing, we introduce the following definition of a J-solution to the uncertain problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$.

Definition 3.1 *The set of permutations $S^{\pi}(T) \subseteq S^{\pi}$ is called a J-solution to the uncertain problem $\mathcal{F}2/p_{ij}^{L} \leq p_{ij} \leq p_{ij}^{U}/\mathcal{C}_{max}$, if for each vector $p \in T$, the set $S^{\pi}(T)$ contains at least one permutation that is a Johnson one for the deterministic problem $\mathcal{F}2//\mathcal{C}_{max}$ associated with the vector p of job processing times, provided that any proper subset of set $S^{\pi}(T)$ loses such a property (i.e., any proper subset of set $S^{\pi}(T)$ is not a J-solution to the uncertain problem $\mathcal{F}2/p_{ij}^{L} \leq p_{ij} \leq p_{ij}^{U}/\mathcal{C}_{max}$).*

From Definition 3.1, it follows that set $S^{\pi}(T)$ contains at least one optimal schedule $\pi_k \in S^{\pi}(T) \subseteq S^{\pi}$ for each vector $p \in T$ of the job processing times: $C_{max}(\pi_k, p) = \min\{C_{max}(\pi_i, p) \ : \ \pi_i \in S^{\pi}\}$, and set $S^{\pi}(T)$ is a minimal set (with respect to inclusion) which possesses such a property. Thus, to solve problem $\mathcal{F}2/p_{ij}^{L} \leq p_{ij} \leq p_{ij}^{U}/\mathcal{C}_{max}$ exactly, one can restrict the search by the set of schedules $S^{\pi}(T)$. As it will be shown in Section 3.2, the set of schedules $S^{\pi}(T)$ provides more choices (than a single schedule) for a decision-maker to organize the optimal realization of the process in real-time.

The remainder of the section is organized as follows. First, we prove necessary and sufficient conditions for the easiest case of problem $\mathcal{F}2/p_{ij}^{L} \leq p_{ij} \leq p_{ij}^{U}/\mathcal{C}_{max}$ when one schedule dominates all others: $|S^{\pi}(T)| = 1$, and for the hardest case when $|S^{\pi}(T)| = n!$. Then we find necessary and sufficient conditions for fixing the order of two jobs in a J-solution. In contrast to Chapter 2, where exponential algorithms based on an exhaustive enumeration of the semiactive schedules were derived for constructing a minimal set of dominant schedules for the job-shop problem $\mathcal{J}m/p_{ij}^{L} \leq p_{ij} \leq p_{ij}^{U}/\mathcal{C}_{max}$ with $m \geq 2$ machines, in this section we develop polynomial procedures for presenting the set $S^{\pi}(T)$ for the special case of the latter problem when $m = 2$ and all the jobs have the same machine route.

Minimal (Maximal) Cardinality of a J-solution

Let us study the case when there exists a permutation $\pi_i \in S^{\pi}$ constituting a single-element set (singleton) that is a J-solution $S^{\pi}(T) = \{\pi_i\}$ to the uncertain problem $\mathcal{F}2/p_{ij}^{L} \leq p_{ij} \leq p_{ij}^{U}/\mathcal{C}_{max}$. Due to Definition 3.1, such a permutation π_i has to be a Johnson permutation for the deterministic problem $\mathcal{F}2//\mathcal{C}_{max}$ associated with any possible vector $p \in T$ of the job processing times. In other words, permutation π_i has to *dominate* each permutation $\pi_k \in S^{\pi}$, i.e., the inequality

$$C_{max}(\pi_i, p) \leq C_{max}(\pi_k, p) \tag{3.3}$$

must hold for any vector $p \in T$ and any permutation $\pi_k \in S^{\pi}$. We construct

a partition $J = \mathcal{J}_0 \cup \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}^*$ of the n jobs defined as follows:

$$\mathcal{J}_0 = \{J_i \in J \ : \ p_{i1}^U \le p_{i2}^L, p_{i2}^U \le p_{i1}^L\};$$

$$\mathcal{J}_1 = \{J_i \in J \ : \ p_{i1}^U \le p_{i2}^L, p_{i2}^U > p_{i1}^L\} = \{J_i \in J \setminus \mathcal{J}_0 \ : \ p_{i1}^U \le p_{i2}^L\};$$
$$\mathcal{J}_2 = \{J_i \in J \ : \ p_{i1}^U > p_{i2}^L, p_{i2}^U \le p_{i1}^L\} = \{J_i \in J \setminus \mathcal{J}_0 \ : \ p_{i2}^U \le p_{i1}^L\};$$

$$\mathcal{J}^* = \{J_i \in J \ : \ p_{i1}^U > p_{i2}^L, p_{i2}^U > p_{i1}^L\};$$

where some subsets $\mathcal{J}_0$, $\mathcal{J}_1$, $\mathcal{J}_2$, and/or $\mathcal{J}^*$ of set J may be empty. Obviously, for each job $J_k \in \mathcal{J}_0$, from the inequalities $p_{k1}^U \le p_{k2}^L$ and $p_{k2}^U \le p_{k1}^L$, we obtain the equalities $p_{k1}^L = p_{k1}^U = p_{k2}^L = p_{k2}^U$. Since both intervals for the processing times of job J_k on machines M_1 and M_2 turn into a point, the processing times p_{k1} and p_{k2} are fixed in problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$, and they are equal for both machines: $p_{k1} = p_{k2} := p_k$. Remark 3.3 directly follows from the definitions of the sets $\mathcal{J}_1$, $\mathcal{J}_2$ and $\mathcal{J}^*$.

Remark 3.3 The sets $\mathcal{J}_1$ and $\mathcal{J}_2$ are such that both inclusions $\mathcal{J}_1 \subseteq N_1$ and $\mathcal{J}_2 \subseteq N_2$ may hold for any vector $p \in T$ of the processing times (sets N_1 and N_2 are those used in Johnson's algorithm). The jobs from set $\mathcal{J}_0$ may be either in set N_1 or in set N_2 regardless of vector $p \in T$. The jobs from set $\mathcal{J}^*$ may be either in set N_1 or in set N_2 depending on the vector $p \in T$.

Using the partition $J = \mathcal{J}_0 \cup \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}^*$, we prove the following claim.

Theorem 3.1 *There exists a single-element J-solution* $S^\pi(T) \subset S^\pi$, $|S^\pi(T)| = 1$, *to problem* $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$ *if and only if*

a) for any pair of jobs J_i *and* J_j *from set* $\mathcal{J}_1$ *(from set* $\mathcal{J}_2$*, respectively) either* $p_{i1}^U \le p_{j1}^L$ *or* $p_{j1}^U \le p_{i1}^L$ *(either* $p_{i2}^U \le p_{j2}^L$ *or* $p_{j2}^U \le p_{i2}^L$*),*

b) $|\mathcal{J}^*| \le 1$ *and for job* $J_{i^*} \in \mathcal{J}^*$ *(if any), the following inequalities hold:*

$$p_{i^*1}^L \ge \max\{p_{i1}^U \ : \ J_i \in \mathcal{J}_1\}; \tag{3.4}$$

$$p_{i^*2}^L \ge \max\{p_{j2}^U \ : \ J_j \in \mathcal{J}_2\}; \tag{3.5}$$

$$\max\{p_{i^*1}^L, \ p_{i^*2}^L\} \ge p_k \ \ \textit{for each job } J_k \in \mathcal{J}_0. \tag{3.6}$$

Proof. *Sufficiency.* Let conditions *a)* and *b)* hold. We have to construct a permutation $\pi_j \in S^\pi$ that is a Johnson permutation for any vector $p \in T$ of the job processing times.

We shall construct the set N_1 and the set N_2 using the sets $\mathcal{J}_1$, $\mathcal{J}_2$, $\mathcal{J}_0$ and $\mathcal{J}^*$. From the definition of the sets $\mathcal{J}_1$ and $\mathcal{J}_2$, it follows that both inclusions $\mathcal{J}_1 \subseteq N_1$ and $\mathcal{J}_2 \subseteq N_2$ may hold for any vector $p \in T$ of the job processing

times (see Remark 3.3). Each job from set $\mathcal{J}_0$ may be included either into set N_1 or into set N_2 regardless of the vector $p \in T$ of job processing times (Remark 3.3). Note that, if for jobs $J_k \in \mathcal{J}_0$ and $J_{i^*} \in \mathcal{J}^*$, inequality $p^L_{i^*1} < p_k$ (inequality $p^L_{i^*2} < p_k$) holds, then we shall include job J_k into set N_2 (set N_1, respectively). Next, we show that, due to condition *a)*, we can order all the jobs from set N_1 (the jobs from set N_2, respectively) using inequalities (3.2) in the same optimal way for all possible vectors $p \in T$ of the job processing times. Indeed, if inequalities $p^U_{i1} \le p^L_{j1}$ (inequalities $p^L_{i2} \le p^L_{j2}$) hold, then due to condition (3.2), we can locate job $J_i \in \mathcal{J}_1$ before job $J_j \in \mathcal{J}_1$ (job $J_i \in \mathcal{J}_2$ after job $J_j \in \mathcal{J}_2$, respectively). Job $J_k \in \mathcal{J}_0$ from set N_1 (from set N_2, respectively) will be located after all jobs $J_i \in \mathcal{J}_1$ (before all jobs $J_i \in \mathcal{J}_2$) in such a way that inequality $p^L_{i1} < p_k$ (inequality $p^L_{i2} < p_k$) will hold. It is easy to see that job $J_k \in \mathcal{J}_0$ will be ordered with the other jobs from set N_1 (from set N_2, respectively) in such a way that conditions (3.2) hold for any vector $p \in T$ of the job processing times. If for the jobs J_i and J_j from set N_1 both inequalities $p^U_{i1} \le p^L_{j1}$ and $p^U_{j1} \le p^L_{i1}$ hold (condition *a)*), then there are two possibilities to order these two jobs with respect to inequalities (3.2). For uniqueness, we shall order these jobs with respect to their indices, i.e., job J_i will be located before job J_j provided that $i < j$. The same agreement will be used for ordering a pair of jobs from set N_2 if both inequalities $p^U_{i2} \le p^L_{j2}$ and $p^U_{j2} \le p^L_{i2}$ hold (condition *a)*). As a result, we obtain a permutation π^1 (permutation π^2) of the jobs from set N_1 (from set N_2), which are ordered with respect to inequalities (3.2) for any possible vector $p \in T$ of the job processing times. Thus, if $\mathcal{J}^* = \emptyset$, the permutation $\pi_j = (\pi^1, \pi^2) \in S^\pi$ obtained by the concatenation of permutations π^1 and π^2 provides a single-element J-solution $S^\pi(T) = \{\pi_j\}$ to problem $\mathcal{F}2/p^L_{ij} \le p_{ij} \le t^U_{ij}/\mathcal{C}_{max}$.

Let there exist a job $J_{i^*} \in \mathcal{J}^* \neq \emptyset$. Then there exist two possible cases *(i)* and *(ii)* depending on the realization of the processing times of job J_{i^*} on machine M_1 and machine M_2 (see Remark 3.3).

(i) $p_{i^*1} < p_{i^*2}$

In this case, job J_{i^*} has to belong to set N_1. Due to conditions (3.4) and (3.6), in a Johnson permutation job J_{i^*} has to be processed after all other jobs from set N_1. Therefore, a Johnson permutation has to be as follows: (π^1, J_{i^*}, π^2).

(ii) $p_{i^*1} \ge p_{i^*2}$

If $p_{i^*1} > p_{i^*2}$ (if $p_{i^*1} = p_{i^*2}$, respectively), job J_{i^*} has to belong (may belong) to set N_2. Due to conditions (3.5) and (3.6), in a Johnson permutation job J_{i^*} has to be processed before all other jobs from set N_2.

Therefore, in case *(ii)*, a Johnson permutation may be the same as in case *(i)*: $\pi_j = (\pi^1, J_{i^*}, \pi^2)$.

Thus, in both cases *(i)* and *(ii)*, we obtain the same Johnson permutation $\pi_j \in S^\pi$, i.e., permutation π_j satisfies condition (3.2) with $i = j$ for any vector $p \in T$ of the job processing times. So, due to Definition 3.1, there exists a single-element J-solution $S^\pi(T) = \{\pi_j\}$ to problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$, which completes the proof of sufficiency.

Necessity. We prove necessity of conditions *a)* and *b)* by contradiction. Assume now that there exists the singleton $S^\pi(T)$ that is a J-solution to problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$, but at least one condition of *a)* or *b)* does not hold.

1. Assume that condition *a)* does not hold. Then the following two cases *(a–1)* and *(a–2)* have to be investigated.

(a–1) There exist at least two jobs $J_i \in \mathcal{J}_1$ and $J_j \in \mathcal{J}_1$ with $p_{i1}^L < p_{j1}^U$ and $p_{j1}^L < p_{i1}^U$.

We construct two vectors $p' \in T$ with $p' = (p'_{1,1}, p'_{1,2}, \ldots, p'_{n1}, p'_{n2})$ and $p'' \in T$ with $p'' = (p''_{1,1}, p''_{1,2}, \ldots, p''_{n1}, p''_{n2})$ such that for vector p' the condition

$$p_{i1}^L = p'_{i1} < p'_{j1} = p_{j1}^U \tag{3.7}$$

holds, and for vector p'' the condition

$$p_{j1}^L = p''_{j1} < p''_{i1} = p_{i1}^U \tag{3.8}$$

holds. The other components p'_{uv} and p''_{uv} of the vectors p' and p'' may take arbitrary but the same feasible values: $p_{uv}^L \le p'_{uv} = p''_{uv} \le p_{uv}^U$. Due to condition (3.2) for vector p' of the job processing times, a Johnson permutation must have the form $\pi_k = (\ldots, J_i, \ldots, J_j, \ldots) \in S^\pi$, while for vector p'' of the job processing times, a Johnson permutation must have the form $\pi_l = (\ldots, J_j, \ldots, J_i, \ldots) \in S^\pi$. Due to the strict inequality (3.7) (strict inequality (3.8), respectively), this order of the jobs J_i and J_j has to be the same in all Johnson permutations constructed for vector p' (vector p'') of the processing times. Due to Definition 3.1, inclusion $\{\pi_k, \pi_l\} \subseteq S^\pi(T)$ must hold for any J-solution $S^\pi(T)$ to problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$, contradicting the assumption.

(a–2) The case when there exist at least two jobs $J_i \in \mathcal{J}_2$ and $J_j \in \mathcal{J}_2$ with $p_{i2}^L < p_{j2}^U$ and $p_{j2}^L < p_{i2}^U$ can be analyzed similarly to the above case *(a–1)*.

2. Assume that conditions *a)* hold and that condition *b)* does not hold. Then there exist four possible cases as follows:

(b–1) Let there exist at least two jobs $J_{i^*} \in \mathcal{J}^*$ and $J_{i^{**}} \in \mathcal{J}^*$, i.e., $|\mathcal{J}^*| > 1$, and for job J_{i^*}, inequalities $p^L_{i^*1} < p^U_{i^*2}$ and $p^L_{i^*2} < p^U_{i^*1}$ hold. For job $J_{i^{**}}$, similar inequalities hold: $p^L_{i^{**}1} < p^U_{i^{**}2}$ and $p^L_{i^{**}2} < p^U_{i^{**}1}$.

In case *(b–1)*, one can construct two vectors $p^* = (p^*_{1,1}, p^*_{1,2}, \ldots, p^*_{n2}) \in T$ and $p^{**} = (p^{**}_{1,1}, p^{**}_{1,2}, \ldots, p^{**}_{n2}) \in T$ such that for vector p^*, conditions $p^L_{i^*1} = p^*_{i^*1} < p^*_{i^*2} = p^U_{i^*2}$ and $p^L_{i^{**}2} = p^*_{i^{**}2} < p^*_{i^{**}1} = p^U_{i^{**}1}$ hold and for vector p^{**}, conditions $p^L_{i^*2} = p^{**}_{i^*2} < p^{**}_{i^*1} = p^U_{i^*1}$ and $p^L_{i^{**}1} = p^{**}_{i^{**}1} < p^{**}_{i^{**}2} = p^U_{i^{**}2}$ hold. The other components p^*_{uv} and p^{**}_{uv} of the vectors p^* and p^{**} may take arbitrary but the same feasible values: $p^L_{uv} \le p^*_{uv} = p^{**}_{uv} \le p^U_{uv}$. If there exist only two such jobs J_{i^*} and $J_{i^{**}}$ (i.e., $|\mathcal{J}^*| = 2$), then for vector p^* a Johnson permutation is as follows: $\pi_a = (\pi^{(1)}, J_{i^*}, \pi^{[1]}, \pi^{(2)}, J_{i^{**}}, \pi^{[2]}) \in S^\pi$ while for vector p^{**}, a Johnson permutation is $\pi_b = (\pi^{(1)}, J_{i^{**}}, \pi^{[1]}, \pi^{(2)}, J_{i^*}, \pi^{[2]}) \in S^\pi$, where $\pi^1 = (\pi^{(1)}, \pi^{[1]})$, $\pi^2 = (\pi^{(2)}, \pi^{[2]})$. (Hereafter, it is assumed that permutations $\pi^{[1]}$ and $\pi^{(2)}$ may be empty.) Due to Definition 3.1, inclusion $\{\pi_a, \pi_b\} \subseteq S^\pi(T)$ holds, contradicting the assumption $|S^\pi(T)| = 1$.

The case $|\mathcal{J}^*| > 2$ may be analyzed similarly.

(b–2) Let $\mathcal{J}^* = \{J_{i^*}\}$ and inequality (3.4) be violated, i.e., there exists at least one job $J_i \in \mathcal{J}_1$ such that the opposite inequality holds:

$$p^L_{i^*1} < p^U_{i1}. \tag{3.9}$$

We consider the vector $p^0 = (p^0_{1,1}, p^0_{1,2}, \ldots, p^0_{n1}, p^0_{n2}) \in T$, where $p^0_{i1} = p^U_{i1}$, $p^0_{i^*1} = p^L_{i^*1}$, $p^0_{i^*2} = p^U_{i^*2}$ and the other components p^0_{uv} may take any feasible values: $p^L_{uv} \le p^0_{uv} \le p^U_{uv}$. Since for job J_{i^*} inequality $p^L_{i^*1} < p^U_{i^*2}$ holds (which follows from the definition of set $\mathcal{J}^* \subseteq J$), we obtain

$$p^0_{i^*1} < p^0_{i^*2}. \tag{3.10}$$

Therefore, inclusion $J_{i^*} \in N_1$ must hold for vector $p^0 \in T$ of the job processing times. From (3.9), it follows

$$p^0_{i^*1} < p^0_{i1}. \tag{3.11}$$

Thus, for the vector p^0 of job processing times, both jobs J_i and J_{i^*} have to belong to set N_1, and for this vector, a Johnson permutation must be as follows: $\pi_c = (\pi^{(1)} \setminus \{J_i\}, J_{i^*}, \pi^{[1]}, J_i, \pi^2) \in S^\pi$, where $\pi^{(1)} \setminus \{J_i\}$ denotes permutation $\pi^{(1)}$ without job J_i. Due to the strict inequalities (3.10) and (3.11), this order of jobs J_{i^*} and J_i has to be fixed in any Johnson permutation, which may be constructed for the vector $p^0 \in T$ of job processing times. On the other hand, we consider vector $p = (p_{1,1}, p_{1,2}, \ldots, p_{n1}, p_{n2}) \in T$ with the same components as vector p^0 except the following three components:

$p_{i1} = p^L_{i1}$, $p_{i^*1} = p^U_{i^*1}$, $p_{i^*2} = p^L_{i^*2}$. As a result, we obtain two inclusions: $J_i \in N_1$ and $J_{i^*} \in N_2$. For the vector p of job processing times, a Johnson permutation is as follows: $\pi_d = (\pi^{(1)} \setminus \{J_i\}, J_1, \pi^{[1]}, \pi^{(2)}, J_{i^*}, \pi^{[2]}) \in S^\pi$ (note that π_c cannot be a Johnson permutation for this vector p). Due to Definition 3.1, any J-solution $S^\pi(T)$ to problem $\mathcal{F}2/p^L_{ij} \le p_{ij} \le p^U_{ij}/\mathcal{C}_{max}$ under consideration must include at least two permutations, namely: $\{\pi_c, \pi_d\} \subseteq S^\pi(T)$. We get a contradiction to the assumption $|S^\pi(T)| = 1$.

(b–3) The case when inequality (3.5) from condition *b)* is violated, i.e., there exists at least one job $J_j \in \mathcal{J}_2$ such that the opposite inequality $p^L_{i^*2} < p^U_{j2}$ holds, may be analyzed similarly to the above case *(b–2)*.

(b–4) Let inequality (3.6) be violated, i.e., there exists at least one job $J_k \in \mathcal{J}_0$ such that the opposite inequality holds:

$$\max\{p^L_{i^*1}, p^L_{i^*2}\} < p_k. \tag{3.12}$$

We consider the vector $p^{++} = (p^{++}_{1,1}, p^{++}_{1,2}, \ldots, p^{++}_{n1}, p^{++}_{n2}) \in T$, where $p^{++}_{i^*1} = p^L_{i^*1}$ and $p^{++}_{i^*2} = p^U_{i^*2}$ provided that the other components p^{++}_{uv} may take any feasible values: $p^L_{uv} \le p^{++}_{uv} \le p^U_{uv}$. Since for job J_{i^*} inequality $p^L_{i^*1} < p^U_{i^*2}$ holds, we obtain $p^{++}_{i^*1} < p^{++}_{i^*2}$. From inequality (3.12), it follows $p_k > p^{++}_{i^*1}$. Therefore, we obtain

$$\min\{p^{++}_{i^*1}, p_k\} < \min\{p^{++}_{i^*2}, p_k\}. \tag{3.13}$$

Due to condition (3.2), a Johnson permutation for the vector p^{++} of job processing times has to be as follows: $\pi_f = (\ldots, J_{i^*}, \ldots, J_k, \ldots) \in S^\pi$. Due to the strict inequality (3.13), this order of jobs J_{i^*} and J_k has to be fixed in any Johnson permutation, which may be constructed for the vector $p^{++} \in T$ of job processing times. On the other hand, we can consider vector $p^{'+} = (p^{'+}_{1,1}, p^{'+}_{1,2}, \ldots, p^{'+}_{n1}, p^{'+}_{n2}) \in T$ with the same components as vector p^{++} except the following ones: $p^{'+}_{i^*1} = p^U_{i^*1}$, $p^{'+}_{i^*2} = p^L_{i^*2}$. Similarly, we can obtain the inequalities $p^{'+}_{i^*2} < p^{'+}_{i^*1}$ and $p_k > p^{'+}_{i^*2}$. Hence, we obtain

$$\min\{p^{'+}_{i^*1}, p_k\} > \min\{p^{'+}_{i^*2}, p_k\}. \tag{3.14}$$

Due to condition (3.2), a Johnson permutation constructed for vector $p^{'+}$ has to look as follows: $\pi_g = (\ldots, J_k, \ldots, J_{i^*}, \ldots) \in S^\pi$. Due to the strict inequality (3.14), this order of jobs J_{i^*} and J_k has to be fixed in any Johnson permutation, which may be constructed for vector $p^{'+} \in T$ (note that π_f cannot be a Johnson permutation for vector $p^{'+}$, while π_g cannot be a Johnson permutation for vector p^{++}). Due to Definition 3.1, any J-solution $S^\pi(T)$ to problem $\mathcal{F}2/p^L_{ij} \le p_{ij} \le p^U_{ij}/\mathcal{C}_{max}$ must include at least

two permutations, namely: $\{\pi_f, \pi_g\} \subseteq S^\pi(T)$. We get a contradiction to the assumption $|S^\pi(T)| = 1$. Thus, equality $|S^\pi(T)| = 1$ implies conditions *a)* and *b)*, and this completes the proof.

$\diamond$

Example 3.2 *We consider problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$ with the set of jobs $J = \{J_1, J_2, J_3, J_4\}$ and the intervals of the processing times defined in Table 3.1.*

Table 3.1: *Lower and upper bounds for the job processing times of Example 3.2*

J_i	J_1	J_2	J_3	J_4
p_{i1}^L	1	3	3	2
p_{i1}^U	3	4	4	2
p_{i2}^L	3	4	1	2
p_{i2}^U	4	5	3	2

It is easy to see that the condition of Theorem 3.1 holds, and so there exists a single-element J-solution $S^\pi(T)$. Indeed, permutation $\pi_2 = (J_1, J_4, J_2, J_3)$ (and permutation $\pi_3 = (J_1, J_2, J_4, J_3)$ as well) is a Johnson permutation for any feasible vector $p \in T$ of the job processing times since condition (3.2) holds for this permutation. Thus, we obtain two single-element J-solutions $S_1^\pi(T)$ and $S_2^\pi(T)$ to the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$, namely: $S_1^\pi(T) = \{\pi_2\}$ and $S_2^\pi(T) = \{\pi_3\}$.

Note that Example 3.1 is a deterministic counterpart of Example 3.2 associated with the vector $p = (3, 4, 4, 5, 4, 3, 2, 2)$ of the upper bounds p_{ij}^U, $i \in \{1, 2, 3, 4\}, j \in \{1, 2\}$, given in Table 3.1. As it was shown for Example 3.1, Johnson's algorithm generates only two permutations $\pi_1 = (J_4, J_1, J_2, J_3)$ and $\pi_4 = (J_1, J_2, J_3, J_4)$. On the other hand, for the deterministic problem $\mathcal{F}2//\mathcal{C}_{max}$ associated with the vector $p = (1, 3, 3, 4, 3, 1, 2, 2)$ of the lower bounds p_{ij}^L, $i \in \{1, 2, 3, 4\}, j \in \{1, 2\}$, given in Table 3.1, Johnson's algorithm generates only two permutations $\pi_2 = (J_1, J_4, J_2, J_3)$ and $\pi_3 = (J_1, J_2, J_4, J_3)$.

If one will consider only permutations from set Π_a (see Remark 3.1) to be included into a dominant set of permutations, then one has to include more than one permutation into such a dominant set of permutations for the Example 3.2 of the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$. However, due to the correct definition of the Johnson permutations (i.e., those satisfying condition (3.2)), for constructing a singleton $S^\pi(T)$ for Example 3.2, one can select a permutation from the whole set $\Pi_c = \{\pi_1, \pi_2, \pi_3, \pi_4\}$ of Johnson permutations (see Remark 3.1) for a deterministic problem $\mathcal{F}2//\mathcal{C}_{max}$ associated with the vector $p \in T$ of job processing times.

Next, we analyze the opposite extreme case when a J-solution to the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$ has a maximal possible cardinality: $|S^\pi(T)| = n!$. We use the following notations: $p_{max}^L = \max\{p_{im}^L : J_i \in J, M_m \in M\}$ and $p_{min}^U = \min\{p_{im}^U : J_i \in J, M_m \in M\}$.

Theorem 3.2 *If for problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$ inequality*

$$p_{max}^L < p_{min}^U \tag{3.15}$$

holds, then $S^\pi(T) = S^\pi$.

Proof. Let $\pi_k = (J_{k_1}, J_{k_2}, \ldots, J_{k_n})$ be an arbitrary permutation from set S^π. In order to prove that this permutation has to belong to each J-solution $S^\pi(T)$ to the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$, we show that it is possible to construct a vector $p' = (p'_{1,1}, p'_{1,2}, \ldots, p'_{n1}, p'_{n2}) \in T$ with the following condition:

$$p_{max}^L = p'_{k_1 1} < p'_{k_2 1} < \ldots < p'_{k_n 1} < p'_{k_1 2} = p'_{k_2 2} = \ldots = p'_{k_n 2} = p_{min}^U. \tag{3.16}$$

Indeed, due to the strict inequality (3.15), the length of the segment $[p_{max}^L, p_{min}^U]$, $p_{max}^L < p_{min}^U$, is strictly positive, and moreover, the segment $[p_{max}^L, p_{min}^U]$ is equal to the intersection of $2n$ segments $[p_{im}^L, p_{im}^U]$, $i \in \{1, 2, \ldots, n\}$, $m \in \{1, 2\}$:

$$[p_{max}^L,\ p_{min}^U] = \left\{ \bigcap_{i=1}^{n} [p_{i1}^L,\ p_{i1}^U] \right\} \bigcap \left\{ \bigcap_{i=1}^{n} [p_{i2}^L,\ p_{i2}^U] \right\}.$$

Since the segment $[p_{max}^L, p_{min}^U]$ of non-negative real numbers is everywhere dense, it is possible to find $2n$ real numbers $p'_{1,1}, p'_{1,2}, \ldots, p'_{n1}, p'_{n2}$ which satisfy all inequalities (3.16). Thus, we can construct the above vector $p' \in T$ for the fixed permutation $\pi_k \in S^\pi$. For such a vector $p' \in T$, all the jobs from set J have to belong to set N_1 (i.e., set N_2 has to be empty). It is easy to see that within permutation π_k, all n jobs are linearly ordered with respect to inequalities (3.2). Therefore, permutation π_k is an optimal Johnson permutation for the vector p' of job processing times. Since $p'_{i1} \ne p'_{j1}$ for each pair of jobs J_i and J_j, $i \ne j$, permutation π_k is a unique Johnson permutation for the vector $p' \in T$ of job processing times. As a result, *permutation π_k has to belong to any J-solution $S^\pi(T)$* (see Definition 3.1).

Since permutation π_k was assumed to be arbitrary in set S^π, the last claim remains correct for any permutation from set S^π. Thus, all permutations from set S^π have to belong to a J-solution $S^\pi(T)$ of the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$, i.e., $S^\pi(T) = S^\pi$.

$\diamond$

The above proof of Theorem 3.2 contains the proof of the following claim.

Corollary 3.1 *If inequality (3.15) holds, then for each permutation $\pi_k \in S^\pi$, there exists a vector $p \in T$ such that π_k is a unique Johnson permutation for the deterministic problem $\mathcal{F}2//\mathcal{C}_{max}$ associated with the vector p of job processing times.*

It is clear that testing condition *a)* of Theorem 3.1 takes $O(n \log_2 n)$ time and testing condition *b)* takes $O(n)$ time. Thus, the conditions of Theorem 3.1 may be tested in $O(n \log_2 n)$ time. The complexity of calculating the values p^L_{max} and p^U_{min} defines the complexity of testing condition (3.15). Thus, testing the condition of Theorem 3.2 and Corollary 3.1 takes $O(n)$ time.

General Case of Problem $\mathcal{F}2/p^L_{ij} \le p_{ij} \le p^U_{ij}/\mathcal{C}_{max}$

We show how to delete redundant permutations from set S^π for constructing a J-solution to problem $\mathcal{F}2/p^L_{ij} \le p_{ij} \le p^U_{ij}/\mathcal{C}_{max}$. To this end, we shall fix the order of jobs $J_v \in J$ and $J_w \in J$ in the desired J-solution if there exists at least one Johnson permutation of the form $\pi_k = (s_1, J_v, s_2, J_w, s_3) \in S^\pi$ for any vector $p \in T$ of the job processing times. Hereafter, the subsequences s_1, s_2 and/or s_3 may be empty, e.g., jobs J_v and J_w may be adjacent in the permutation: $\pi_k = (s_1, J_v, J_w, s_3) \in S^\pi$. Next, we prove three sufficient conditions for fixing such an order of two jobs.

Lemma 3.1 *If the following condition holds:*

$$p^U_{w2} \le p^L_{w1} \text{ and } p^U_{v1} \le p^L_{v2}; \tag{3.17}$$

then for each vector $p \in T$ of the job processing times there exists a permutation $\pi_k = (s_1, J_v, s_2, J_w, s_3) \in S^\pi$ that is a Johnson permutation for problem $\mathcal{F}2//\mathcal{C}_{max}$ with the vector $p \in T$ of job processing times.

Proof. Let polytope T satisfy inequalities (3.17). We consider any fixed vector $p \in T$ of the job processing times. Let in the vector p, jobs J_w and J_v have the processing times p_{w1}, p_{w2}, p_{v1} and p_{v2}. From the first inequality in (3.17), it follows that $p_{w2} \le p_{w1}$ and therefore, there exists a Johnson permutation for the vector p of processing times in which job J_w is included into the set N_2. From the second inequality in (3.17), it follows that $p_{v1} \le p_{v2}$ and therefore, there exists a Johnson permutation for the vector p of processing times in which job J_v is included into the set N_1.

Let π_l denote such a Johnson permutation in which inclusions $J_v \in N_1$ and $J_w \in N_2$ simultaneously hold. Due to Johnson's rule, job J_v has to

precede job J_w in permutation π_l, i.e., permutation π_l may be represented in the following form $\pi = \pi_l = (s_1, J_v, s_2, J_w, s_3) \in S^\pi$. For this fixed vector p of the job processing times, permutation π_l dominates any other permutation from set S^π with respect to the singleton $\{p\}$. To complete the proof, we note that vector p was arbitrarily taken from set T.

$\diamond$

Lemma 3.2 *If the following condition holds:*

$$p_{v1}^U \le p_{w1}^L \text{ and } p_{v1}^U \le p_{v2}^L, \tag{3.18}$$

then for each vector $p \in T$ of the job processing times, there exists a permutation $\pi_k = (s_1, J_v, s_2, J_w, s_3) \in S^\pi$ that is a Johnson permutation for problem $\mathcal{F}2//\mathcal{C}_{max}$ with the vector $p \in T$ of job processing times.

Proof. Let polytope T satisfy inequalities (3.18). We consider any fixed vector $p \in T$ of the job processing times. Assume that in vector $p \in T$, job J_v and J_w have the processing times p_{v1}, p_{v2}, p_{w1} and p_{w2}. From the second inequality in (3.18), it follows that $p_{v1} \le p_{v2}$. Therefore, there exists a Johnson permutation π_r for the vector p of processing times in which job J_v is included into the set N_1. In permutation π_r, job J_w may be either in the set N_1 or in the set N_2. Next, we consider both cases.

Case 1. Let job J_w belong to set N_1 in the Johnson permutation π_r.

Then from the first inequality in (3.18), it follows that $p_{v1} \le p_{v1}^U \le p_{w1}^L \le p_{w1}$. Since $p_{v1} \le p_{w1}$ for jobs J_v and J_w in the set N_1, job J_v has to precede job J_w in the Johnson permutation π_r, i.e., permutation π_r may be represented in the form $\pi_r = (s_1, J_v, s_2, J_w, s_3) \in S^\pi$. For this fixed vector p of the processing times, permutation π_r dominates any other permutation from set S^π with respect to the singleton $\{p\}$.

Case 2. Let job J_v belong to set N_2 in the Johnson permutation π_r.

By an argument similar to the proof of Lemma 3.1, we can show that job J_w precedes job J_v in the Johnson permutation π_r, and so permutation π_r dominates any permutation from S^π. We conclude that for each vector $p \in T$ of the processing times, in both cases there exists a permutation $\pi_r = \pi = (s_1, J_v, s_2, J_w, s_3) \in S^\pi$ which dominates each permutation from set S^π.

$\diamond$

Lemma 3.3 *If the following condition holds:*

$$p_{w2}^U \le p_{w1}^L \text{ and } p_{w2}^U \le p_{v2}^L. \tag{3.19}$$

then for each vector $p \in T$ of the job processing times, there exists a permutation $\pi_k = (s_1, J_v, s_2, J_w, s_3) \in S^\pi$ that is a Johnson permutation for problem $\mathcal{F}2//\mathcal{C}_{max}$ with the vector $p \in T$ of job processing times.

PROOF. It is easy to see that inequalities (3.19) in Lemma 3.3 coincide with inequalities (3.18) in Lemma 3.2 when interchanging job J_v and job J_w. Thus, the proof of Lemma 3.3 is analogous to the proof of Lemma 3.2. ◇

Note that, if condition (3.17) holds, then job J_v belongs to set N_1 and job J_w belongs to the corresponding set N_2 for all possible realizations of the processing times. If condition (3.18) holds, then job J_v belongs to set N_1 for all possible realizations of the processing times, while job J_w may be either in set N_1 or in set N_2 for different realizations of the job processing times. If condition (3.19) holds, then job J_w belongs to set N_2 for all possible realizations of the processing times, while job J_v may be either in set N_1 or in set N_2 for different realizations of the job processing times.

Next, we prove that, if at least one of the conditions (3.17)–(3.19) holds, then there exists a J-solution $S^\pi(T)$ to problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$ with the fixed order $J_v \to J_w$, i.e., job $J_v \in J$ has to be located before job $J_w \in J$ in any permutation $\pi_i \in S^\pi(T)$. We shall also prove that, if both conditions (3.18) and (3.19) do not hold, then there is no J-solution $S^\pi(T)$ with the fixed order $J_v \to J_w$ in all permutations $\pi_i \in S^\pi(T)$. If in addition, no analogous condition holds for the opposite order $J_w \to J_v$, then at least one permutation with job J_v located before job J_w and that with job J_w located before job J_v must be included in any J-solution $S^\pi(T)$ to problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$. Summarizing, we prove the following claim.

Theorem 3.3 *There exists a J-solution $S^\pi(T)$ to problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$ with the fixed order $J_v \to J_w$ of jobs $J_v \in J$ and $J_w \in J$ if and only if at least one of the conditions (3.18) or (3.19) holds.*

PROOF. *Sufficiency.* From Lemmas 3.2 and 3.3, it follows that, if at least one of the conditions (3.18) or (3.19) holds, then for each vector $p \in T$ of the job processing times, there exists a Johnson permutation $\pi_i \in S^\pi$ with the same order $J_v \to J_w$ of jobs J_v and J_w. Let S' denote the subset of all such permutations of set S^π: $S' \subseteq S^\pi$. After deleting from set S' all the redundant permutations (see Definition 3.1), we obtain a minimal dominant set of permutations (i.e., a J-solution $S^\pi(T) \subseteq S'$) with the fixed order $J_v \to J_w$ to the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$.

Necessity. Let there exist a J-solution $S^\pi(T)$ with the fixed order $J_v \to J_w$ to the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$.

Due to Definition 3.1, for the deterministic problem $\mathcal{F}2//\mathcal{C}_{max}$ associated with any vector $p \in T$ of the job processing times, there exists a Johnson permutation of the form $\pi_i = (s_1, J_v, s_2, J_w, s_3) \in S^\pi$ (recall that a subsequence s_j, $j \in \{1,2,3\}$, may be empty). Due to the definition of a Johnson permutation, condition (3.2) must hold with $i_k = v$ and $i_m = w$:

$$\min\{p_{v1}, p_{w2}\} \le \min\{p_{w1}, p_{v2}\}. \tag{3.20}$$

Since inequality (3.20) must hold for any vector $p \in T$, we conclude that the following inequality holds:

$$\min\{p_{v1}^U, p_{w2}^U\} \le \min\{p_{w1}^L, p_{v2}^L\}. \tag{3.21}$$

Next, we show that inequality (3.21) implies at least one condition (3.18) or (3.19). To this end, four cases $(i) - (iv)$ have to be analyzed.

(i) $p_{v1}^U \le p_{w2}^U$, $p_{w1}^L \le p_{v2}^L$.

In case (i), inequality (3.21) implies $p_{v1}^U = \min\{p_{v1}^U, p_{w2}^U\} \le \min\{p_{w1}^L, p_{v2}^L\} = p_{w1}^L$. Thus, $p_{v1}^U \le p_{w1}^L \le p_{v2}^L$. We obtain condition (3.18).

(ii) $p_{v1}^U \le p_{w2}^U$, $p_{w1}^L > p_{v2}^L$.

In case (ii), inequality (3.21) implies $p_{v1}^U = \min\{p_{v1}^U, p_{w2}^U\} \le \min\{p_{w1}^L, p_{v2}^L\} = p_{v2}^L$. Thus, $p_{v1}^U \le p_{v2}^L < p_{w1}^L$. We obtain condition (3.18).

(iii) $p_{v1}^U > p_{w2}^U$, $p_{w1}^L \le p_{v2}^L$.

In case (iii), inequality (3.21) implies $p_{w2}^U = \min\{p_{v1}^U, p_{w2}^U\} \le \min\{p_{w1}^L, p_{v2}^L\} = p_{w1}^L$. Thus, $p_{w2}^U \le p_{w1}^L \le p_{v2}^L$. We obtain condition (3.19).

(iv) $p_{v1}^U > p_{w2}^U$, $p_{w1}^L > p_{v2}^L$.

In case (iv), inequality (3.21) implies $p_{w21}^U = \min\{p_{v1}^U, p_{w2}^U\} \le \min\{p_{w1}^L, p_{v2}^L\} = p_{v2}^L$. Thus, $p_{w2}^U \le p_{v2}^L < p_{w1}^L$. We obtain condition (3.19).

Theorem 3.3 has been proven.

◇

Arguing similarly as the above sufficiency proof, one can show that *condition (3.17) is sufficient for the existence of a J-solution $S^\pi(T)$ with the fixed order $J_v \to J_w$ of jobs J_v and J_w to the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$.*

Note also that *condition (3.17) implies at least one condition (3.18) or (3.19)* (while the opposite claim is not true). Indeed, if condition (3.17) holds, then there may exist one of only three possibilities:

- condition (3.18) holds and condition (3.19) does not hold;

- condition (3.19) holds and condition (3.18) does not hold;
- both conditions (3.18) and (3.19) hold.

Let $J \times J$ denote the Cartesian product of set J. Due to Theorem 3.3, via testing inequalities (3.18) and (3.19) for each pair of jobs $J_v \in J$ and $J_w \in J$, one can construct a binary relation $\mathcal{A}_{\preceq} \subseteq J \times J$ over the set J as follows: *inclusion* $(J_v, J_w) \in \mathcal{A}_{\preceq}$ *holds if and only if there exists a J-solution* $S^{\pi}(T)$ *to the uncertain problem* $\mathcal{F}2/p_{ij}^{L} \leq p_{ij} \leq p_{ij}^{U}/\mathcal{C}_{max}$ *such that job* $J_v \in J$ *is located before job* $J_w \in J$, $v \neq w$, *in all permutations* $\pi_k \in S^{\pi}(T)$. The binary relation $\mathcal{A}_{\preceq}$ defines a digraph $(J, \mathcal{A}_{\preceq})$ with the vertex set J and the arc set $\mathcal{A}_{\preceq}$. Relation $(J_v, J_w) \in \mathcal{A}_{\preceq}$ will be also represented as $J_v \preceq J_w$. Obviously, it takes $O(n^2)$ time to construct the digraph $(J, \mathcal{A}_{\preceq})$.

Let us consider the case when $\mathcal{J}_0 = \emptyset$, i.e., $J = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$. For a pair of jobs $J_v \in \mathcal{J}_1$ and $J_w \in \mathcal{J}_1$, it may happen that there exist both a J-solution $S^{\pi}(T) \subset S^{\pi}$ with job J_v located before job J_w in all permutations $\pi_k \in S^{\pi}(T)$ and a J-solution $S_*^{\pi}(T) \subset S^{\pi}$ with job J_w located before job J_v in all permutations $\pi_l \in S_*^{\pi}(T)$. In such a case, we can define a binary relation $\mathcal{A}_{\prec} \subset \mathcal{A}_{\preceq} \subseteq J \times J$ using the following agreements: *if* $J_v \preceq J_w$ *and* $J_w \not\preceq J_v$, *then* $J_v \prec J_w$; *if* $J_v \preceq J_w$ *and* $J_w \preceq J_v$ *with* $v < w$, *then* $J_v \prec J_w$ *and* $J_w \not\prec J_v$. Since set $\mathcal{J}_0$ is empty, we obtain an antireflective, antisymmetric, and transitive binary relation $\mathcal{A}_{\prec}$, i.e., a strict order. A strict order $\mathcal{A}_{\prec}$ defines a digraph $(J, \mathcal{A}_{\prec})$ with the arc set $\mathcal{A}_{\prec}$. Digraph $(J, \mathcal{A}_{\prec})$ has neither a circuit nor a loop. Using Theorem 3.3, one can represent the J-solution $S^{\pi}(T)$ in a more condense form via constructing a *reduction digraph* of the strict order relation $\mathcal{A}_{\prec}$. Let $(J, \mathcal{A}_{\prec}^*)$ denote the digraph with the vertex set J and the arc set $\mathcal{A}_{\prec}^* \subseteq \mathcal{A}_{\prec}$, where the arc (J_i, J_w) belongs to set $\mathcal{A}_{\prec}^*$ if and only if relation $J_i \prec J_w$ holds for jobs $J_i \in J$ and $J_w \in J$, and there is no job $J_j \in J$ such that both relations $J_i \prec J_j$ and $J_j \prec J_w$ hold. Digraph $(J, \mathcal{A}_{\prec}^*)$ has neither a transitive arc nor a circuit nor a loop.

Next, we consider the case when $\mathcal{J}_0 \neq \emptyset$. In this general case, the binary relation $\mathcal{A}_{\preceq}$ defined over set $\mathcal{J}$ is a pseudo-order relation since the binary relation $\mathcal{A}_{\preceq}$ definitely possesses only the transitivity property. The digraph $(\mathcal{J}, \mathcal{A}_{\preceq})$ defined by the binary relation $\mathcal{A}_{\preceq}$ may contain circuits and loops. However, a loop in the digraph $(\mathcal{J}, \mathcal{A}_{\preceq})$ has no sense while defining a J-solution $S^{\pi}(T)$ to the uncertain problem $\mathcal{F}2/p_{ij}^{L} \leq p_{ij} \leq p_{ij}^{U}/\mathcal{C}_{max}$.

Theorem 3.4 *A) If* $J_k \in \mathcal{J}_0$, $J_i \in \mathcal{J}_1$, *and*

$$p_{i1}^{L} < p_k, \tag{3.22}$$

then $J_i \preceq J_k$ *and* $J_k \not\preceq J_i$.

B) If $J_k \in \mathcal{J}_0$, $J_i \in \mathcal{J}_2$, *and*

$$p_{i2}^L < p_k, \tag{3.23}$$

then $J_k \preceq J_i$ *and* $J_i \not\preceq J_k$.

C) If $J_k \in \mathcal{J}_0$, $J_i \in \mathcal{J}^*$, *inequality (3.22) holds, and inequality (3.23) does not hold, then* $J_k \preceq J_i$ *and* $J_i \not\preceq J_k$.

D) If $J_k \in \mathcal{J}_0$, $J_i \in J^*$, *inequality (3.23) holds, and inequality (3.22) does not hold, then* $J_k \preceq J_i$ *and* $J_i \not\preceq J_k$.

PROOF. *Part A).* Let $J_k \in \mathcal{J}_0$, $J_i \in \mathcal{J}_1$, and inequality (3.22) hold.

Inclusion $J_k \in \mathcal{J}_0$ implies $p_{k2}^U \le p_{k1}^L$. Inclusion $J_i \in \mathcal{J}_1$ implies $p_{i1}^U \le p_{i2}^L$. Thus, condition (3.17) holds with $w = k$ and $v = i$. Therefore, there exists a J-solution $S^\pi(T)$ to the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$ with the fixed order $J_i \to J_k$ of jobs J_i and J_k, i.e., $J_i \preceq J_k$.

We prove $J_k \not\preceq J_i$ by contradiction. Let the opposite $J_k \preceq J_i$ hold, i.e., assume that there exists a J-solution $S_*^\pi(T)$ to the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$ with the fixed order $J_k \to J_i$. We consider vector $p' \in T$ with $p'_{i1} = p_{i1}^L$ and $p'_{i2} = p_{i2}^U$. The other components p'_{jm}, $j \ne i$, of vector p' may take any possible values: $p_{jm}^L \le p'_{jm} \le p_{jm}^U$, $M_m \in M$. Recall that inclusion $J_k \in \mathcal{J}_0$ implies $p'_{k1} = p'_{k2} = p_k$. For vector $p' \in T$, we obtain

$$\min\{p'_{i1}, p'_{k2}\} = \min\{p_{i1}^L, p_k\} = p_{i1}^L < \min\{p_k, p_{i2}^U\} = \min\{p'_{k1}, p'_{i2}\}. \tag{3.24}$$

The second equality in (3.24) follows from (3.22). The inequality in (3.24) follows from (3.22) and inclusion $J_i \in \mathcal{J}_1$. Thus, we obtain inequality $\min\{p'_{i1}, p'_{k2}\} < \min\{p'_{k1}, p'_{i2}\}$ which may be interpreted as a strong inequality from (3.2). Due to condition (3.2), in all Johnson permutations constructed for vector $p' \in T$ of the job processing times, the order of jobs J_k and J_i has to be as follows: $J_i \to J_k$. Therefore, any J-solution to the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$ has to include at least one permutation $\pi_s \in S^\pi$ such that job J_i is located before job J_k in permutation π_s. This contradicts to the above assumption that a J-solution $S_*^\pi(T)$ with the fixed order $J_k \to J_i$ exists. Thus, $J_k \not\preceq J_i$.

Part B) may be proven similarly as *part A).*

Part C). Let $J_k \in \mathcal{J}_0$, $J_i \in \mathcal{J}^*$, inequality (3.22) holds, and inequality (3.23) does not hold.

Inclusion $J_k \in \mathcal{J}_0$ implies $p_{k2}^U \le p_{k1}^L$. Since inequality (3.23) does not hold, we obtain $p_{i2}^L \ge p_k = p_{k2}^U$. Thus, condition (3.19) holds with $w = k$ and $v = i$. Due to Theorem 3.3, there exists a J-solution $S^\pi(T)$ to the

uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$ with the fixed order $J_i \to J_k$ of jobs J_i and J_k, i.e., $J_i \preceq J_k$.

We prove $J_k \not\preceq J_i$ by contradiction. Let the opposite $J_k \preceq J_i$ hold, i.e., assume that there exists a J-solution $S_0^\pi(T)$ to the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$ with the fixed order $J_k \to J_i$ of jobs J_i and J_k. We consider the same vector $p' \in T$ defined in the above proof of *part A)*. For vector $p' \in T$, we can again obtain (3.24), but in this case the inequality in (3.24) follows from (3.22) and inclusion $J_i \in \mathcal{J}^*$. Arguing in just the same way as in the proof of *part A)*, we obtain a contradiction to the assumption that a J-solution $S_0^\pi(T)$ with the fixed order $J_k \to J_i$ exists. Thus, $J_k \not\preceq J_i$.

For the proof of *part D)*, we can argue similarly as the previous *part C)* provided that condition (3.19) is replaced by (3.18). This completes the proof of Theorem 3.4.

◇

If inequality $p_k > \max\{p_{i1}^L,\ p_{i2}^L\}$ holds (i.e., both inequalities (3.22) and (3.23) hold), then similar to case *(b–4)* of the necessity proof of Theorem 3.1, one can prove that the order of these two jobs cannot be fixed in any J-solution $S^\pi(T)$ to the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$, i.e., both relations $J_i \not\preceq J_k$ and $J_k \not\preceq J_i$ hold.

Theorem 3.5 *If $J_k \in \mathcal{J}_0$, $J_i \in \mathcal{J}_1$ ($J_i \in \mathcal{J}_2$), and $p_{i1}^L \ge p_k$ ($p_{i2}^L \ge p_k$, respectively), then both relations $J_i \preceq J_k$ and $J_k \preceq J_i$ hold.*

Proof. Let $J_k \in \mathcal{J}_0$, $J_i \in \mathcal{J}_1$, $p_{i1}^L \ge p_k$. Then inclusion $J_k \in \mathcal{J}_0$ implies $p_{k1}^U = p_k = p_{k2}^L$. Inclusion $J_k \in \mathcal{J}_0$ and inequality $p_{i1}^L \ge p_k$ imply $p_{k1}^U = p_k \le p_{i1}^L$. Thus, condition (3.18) holds with $v = i$ and $w = k$. Due to Theorem 3.1, relation $J_i \preceq J_k$ must hold. On the other hand, inclusion $J_i \in \mathcal{J}_1$ implies $p_{i1}^U \le p_{i2}^L$. Inclusion $J_k \in \mathcal{J}_0$ implies $p_{k2}^U = p_k = p_{k1}^L$. Therefore, condition (3.17) holds with $v = k$ and $w = i$. Thus, relation $J_k \preceq J_i$ must hold.

The case when $J_k \in \mathcal{J}_0$, $J_i \in \mathcal{J}_2$, and $p_{i2}^L \ge p_k$, may be analyzed similarly provided that condition (3.19) is used instead of condition (3.18).

◇

Using Theorems 3.3, 3.4, and 3.5 for the case $\mathcal{J}_0 \ne \emptyset$, we can construct the digraph $(J, \mathcal{A}_\preceq)$ which may contain circuits. If the conditions of Theorem 3.5 do not hold for any job $J_i \in \mathcal{J}_1$ ($J_i \in \mathcal{J}_2$), then we can construct a digraph $(J, \mathcal{A}_\prec)$ without circuits. The former digraph uniquely defines a J-solution $S^\pi(T)$ to the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$. If there exists a job $J_i \in \mathcal{J}_1$ ($J_i \in \mathcal{J}_2$) such that inequality $p_{i1}^L \ge p_k$ (inequality

$p_{i2}^L \geq p_k$) holds, then we can construct a family of J-solutions to the problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$ via fixing different positions for job $J_k \in \mathcal{J}_0$ in the permutations from a J-solution. Such a family of J-solutions may be used by a decision-maker for selecting the best permutation to be realized using additional information obtained after processing some jobs from set J.

It should be noted that, since the cardinality of a J-solution $S^\pi(T)$ may vary for different uncertain problems $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$ in the range $[1, n!]$, there is no polynomial algorithm for a direct enumeration of all permutations of set $S^\pi(T)$. However, due to Theorems 3.3, 3.4, and 3.5, one can construct the digraph $(J, \mathcal{A}_{\preceq})$ or the digraph $(J, \mathcal{A}_{\prec})$ in $O(n^2)$ time. Digraph $(J, \mathcal{A}_{\prec})$ (digraph $(J, \mathcal{A}_{\preceq})$) defines a set $S^\pi(T)$ (a family of sets $\{S_j^\pi(T)\}$) and may be considered as a condense form of a J-solution (family of J-solutions) to the uncertain problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$. Of course, the more job pairs are involved in the binary relation $\mathcal{A}_{\preceq}$, the more redundant permutations will be deleted from set S^π while constructing a J-solution $S^\pi(T) \subseteq S^\pi$ to the uncertain problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$.

Computational Results and Resume

It is clear that it is necessary to choose one schedule for a practical realization of the process defined in problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$. That is why, if no additional information is available to a decision-maker (and if the set of feasible schedules is restricted by Johnson ones), only the condition of Theorem 3.1 allows one to obtain (without failure) a single dominant permutation before the realization of the process (in an *off-line* fashion). Such a dominant permutation (if any) is the best one for any possible realization of the job processing times. Next, we answer experimentally the question of how many randomly generated instances of the uncertain problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$ may be solved exactly due to Theorem 3.1 (in spite of the uncertainty of the job processing times).

The algorithm was coded in MATLAB and an AMD 3000 MHz processor was used for making our experiments. Table 3.2 presents the percentage of instances solved exactly due to Theorem 3.1 with small intervals of possible processing times. The lower bound for the job processing time was randomly generated from a uniform distribution of integers over the closed interval $[1, 1000]$, and an upper bound was calculated as follows: $p_{ij}^U = p_{ij}^L + L$, where $L \in \{1, 2, \ldots, 10\}$. For each combination of L and $n \in \{5, 10, \ldots, 70, 80, 90, 100\}$, a series of 1000 instances was randomly generated and tested. The CPU time for testing conditions *a)* and *b)* of Theorem

3.1 was less than 4 s for each series. Table 3.3 presents the percentage of instances solved exactly due to Theorem 3.1, where the upper bound for the job processing time was calculated as follows: $p_{ij}^U = p_{ij}^L \cdot (1 + l\%/100\%)$. As it follows from Tables 3.2 and 3.3, Theorem 3.1 is practically useful if the level of uncertainty is low and the number of jobs is small. In particular, our experiments found no instance satisfying conditions *a)* and *b)*, if $L \cdot n > 300$ or $l \cdot n > 200$.

Table 3.2: *Percentage of instances with $p_{ij}^U = p_{ij}^L + L$ solved due to Theorem 3.1*

	Number of jobs																
L	5	10	15	20	25	30	35	40	45	50	55	60	65	70	80	90	100
1	99.2	95.2	91.2	86.1	79.2	72.8	63.9	58.1	49.2	41.9	34.2	29.5	24	17.5	10.1	6.5	3.5
2	97.2	89.8	77.6	63.5	51	39.6	28.2	16.4	13.3	7.8	3.3	2.9	0.9	0.4	0.1	0	0.1
3	95	80.9	66.4	47.6	32.8	20.6	9.4	6.2	3.2	1.6	0.5	0.1	0	0	0	0	0
4	91.8	78.6	56	39.2	20.3	10.7	4.5	2.1	0.6	0.2	0	0	0	0	0	0	0
5	91	69.4	44.9	28.9	14.6	6	2	0.4	0.2	0	0	0	0	0	0	0	0
6	89.1	65	42.2	22.1	8.2	2.7	0.4	0.4	0.1	0.1	0	0	0	0	0	0	0
7	87.6	61	33.7	13.4	4.6	1.2	0.4	0.1	0	0	0	0	0	0	0	0	0
8	86.8	54.2	27.3	12.7	3.2	0.9	0.2	0	0	0	0	0	0	0	0	0	0
9	86.9	48.3	24.7	7.2	1.9	0.2	0.	0	0	0	0	0	0	0	0	0	0
10	84.8	50.3	19	5.5	2	0.5	0.1	0	0	0	0	0	0	0	0	0	0

If conditions *a)* and *b)* do not hold, it is useful to construct the digraph $(J, \mathcal{A}_{\preceq})$ due to Theorems 3.3, 3.4, and 3.5. Digraph $(J, \mathcal{A}_{\preceq})$ may be used for constructing a solution to the uncertain problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$.

The above results for solving the uncertain problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$ aim to complement general methods dealing with uncertain scheduling environments, e.g., a stochastic method [269], a fuzzy method [299], and a robust method [189]. The former method is practically useful when the scheduler has enough prior information to characterize the probability distributions of the random processing times and there is a large number of realizations of similar processes. The latter method allows one to determine the schedule with the best worst-case performance compared to the corresponding optimal schedule over all potential realizations of the job processing times. In this section, we focus on constructing a minimal set of schedules that dominates all the others. Due to the obtained results, it is sometimes possible to find an optimal schedule in spite of uncertain numerical data. In

Table 3.3: *Percentage of instances with* $p_{ij}^U = p_{ij}^L \cdot (1 + l\%/100\%)$ *solved due to Theorem 3.1*

	Number of jobs																
$l\%$	5	10	15	20	25	30	35	40	45	50	55	60	65	70	80	90	100
1	94.5	85.1	70	53.5	36.9	24.9	14.7	8.2	4.8	1.9	1.3	0.6	0.3	0	0	0	0
2	91.2	69.3	45.6	24.2	11.8	4.2	1.1	0.6	0.1	0	0	0	0	0	0	0	0
3	87.7	58.4	28.3	10.3	3.8	1.3	0.2	0	0	0	0	0	0	0	0	0	0
4	82.4	47.4	18.8	5.5	0.8	0.1	0	0	0	0	0	0	0	0	0	0	0
5	75.5	37.4	11.3	2.2	0.2	0	0	0	0	0	0	0	0	0	0	0	0
6	72.5	32.4	7.8	1.7	0	0	0	0	0	0	0	0	0	0	0	0	0
7	66.9	25	5.4	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0
8	65.8	19.9	2.1	0.4	0	0	0	0	0	0	0	0	0	0	0	0	0
9	63.4	18	2.4	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0
10	59.2	14.4	2	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0

particular, if the condition of Theorem 3.1 holds, then the scheduler can use the dominant schedule π_i, $\{\pi_i\} = S^\pi(T)$, which is necessarily optimal for any possible realization of the job processing times. If there is no possibility to construct a singleton $\{\pi_i\} = S^\pi(T)$, we propose to construct a J-solution $S^\pi(T)$, $|S^\pi(T)| > 1$, to the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$ and use it as a more general schedule form. In particular, using a J-solution, one can look for a *robust* schedule and reduce time-consuming computations [187] via treating a minimal set $S^\pi(T)$ of dominant permutations instead of the whole set S^π of job permutations. Of course, this reduction of the computations may be essential only if the cardinality of set $S^\pi(T)$ is significantly less than $|S^\pi| = n!$. We suggest that efforts to quantify these thresholds via a computer simulation is a useful direction for future work. To this end, it is essential to conduct well-designed computational experiments and analyze their results in an appropriate manner. Such a method is consistent with the hierarchical approach to scheduling and control adopted by many researches over the last several decades and corresponds to industrial practices [24]. In the *static phase*, when the level of uncertainty of the input data is high, the scheduler finds a J-solution $S^\pi(T)$ to the uncertain problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$. In order to find an optimal schedule, the subset of schedules from a J-solution to the uncertain subproblem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{max}$ obtained after each decision being made in the *dynamic phase* has to remain a J-solution for the remaining subset of

jobs. For future work, it is also useful to develop further the background for selecting the job to be processed next if there are no sufficient conditions for an optimal job ordering. For such a decision-making process different methods have been developed. The aim of future research may be to find other sufficient conditions for the existence of a single dominant permutation π_i. To this end, one can relax the above agreement for the desired permutation π_i to be a Johnson one. Instead of condition (3.2) for the optimality of permutation $\pi_i \in S^\pi$, one can consider more general conditions proven in [48, 232, 244, 246]. A larger subset of optimal permutations may provide more choices to the decision-maker to find an optimal schedule. The conditions proven in this section may be tested in polynomial time. However, for most generalizations of the two-machine flow shop problem, the existence of polynomial algorithms is unlikely (e.g., if the deterministic counterpart is NP-hard). So, for future more general and thus more practical investigations, it will be reasonable to look for methods of a branch-and-bound type, and it will also be useful to consider heuristic solutions instead of optimal ones.

3.2. Schedule Execution for a Flow Shop

This section addresses the issue of how to execute best a schedule in a two-phase scheduling decision framework by considering an uncertain two-machine flow shop problem. There are considered two phases in the scheduling process: the off-line phase (the schedule planning phase) and the on-line phase (the schedule execution phase). The information about the lower and upper bound for each uncertain processing time is available at the beginning of the off-line phase while the local information on the realization (the actual value) of each uncertain processing time is available once the corresponding operation is completed. In the off-line phase, the scheduler prepares a minimal set of dominant schedules, which is derived based on a set of sufficient conditions for schedule domination that we developed in Section 3.1 and in this section. This set of dominant schedules enables the scheduler to make quickly an on-line scheduling decision whenever additional local information on the realization of an uncertain processing time is available. Our approach enables the scheduler to execute a schedule best and may end up with executing the schedule optimally in many instances according to our extensive computational experiments which are based on randomly generated data up to 1000 jobs. The algorithm for testing the set of sufficient conditions of schedule domination is not only theoretically appealing (i.e., polynomial in

the number of jobs) but also empirically fast as our extensive computational experiments indicate.

We shall use the partition $J = \mathcal{J}_0 \cup \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}^*$ given in Section 3.1 on page 184. First, we consider the case of $\mathcal{J}_0 \neq \emptyset$. The jobs J_k of set $\mathcal{J}_0$ play a specific role in constructing a J-solution to problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$. If $J_i \preceq J_k$ and $J_k \preceq J_i$, then there exist J-solutions $S_l^\pi(T)$ and $S_j^\pi(T)$ such that for each permutation of set $S_l^\pi(T)$, job J_k precedes J_i while for each permutation of set $S_j^\pi(T)$, job J_k precedes J_i. Consequently, we can construct a family of J-solutions $\{S_j^\pi(T)\} = \{S_1^\pi(T), S_2^\pi(T), \ldots, S_l^\pi(T)\}$ to problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$ via fixing job $J_k \in \mathcal{J}_0$ at the candidate positions. Instead of using a single solution $S^\pi(T)$ as in the case of $\mathcal{J}_0 = \emptyset$, using a family of solutions $\{S_j^\pi(T)\}$ in the case of $\mathcal{J}_0 \neq \emptyset$ offers more flexibility in the on-line scheduling phase.

Remark 3.4 Let $J_i \preceq J_k$, $J_k \not\preceq J_i$, $J_k \preceq J_w$, and $J_w \not\preceq J_k$. In what follows, we shall consider only a family of solutions $\{S_j^\pi(T)\}$ in which for each set $S_j^\pi(T)$ of the family $\{S_j^\pi(T)\}$, each job $J_k \in \mathcal{J}_0$ is located at some position between job $J_i \in \mathcal{J}_1 \cup \mathcal{J}^*$ and job $J_w \in \mathcal{J}_2 \cup \mathcal{J}^*$ for all permutations of set $S_j^\pi(T)$.

We shall consider only the family of solutions $\{S_j^\pi(T)\}$ defined in Remark 3.4 since we shall take advantage of the *local* information to schedule the conflicting jobs that compete for the same machine at the same time. Based on Remark 3.4, for each job J_k of the set $\mathcal{J}_0 \neq \emptyset$, we can define the *candidate area* of job J_k for the J-solutions of the family $\{S_j^\pi(T)\}$ as follows. If $J_k \in \mathcal{J}_0$, then there exist jobs J_u and J_v such that the following equalities hold:

$$p_{u1}^L = \max\{p_{i1}^L \ : \ p_{i1}^L < p_k, \ J_i \in \mathcal{J}_1 \cup \mathcal{J}^*\}, \tag{3.25}$$

$$p_{v2}^L = \max\{p_{i2}^L \ : \ p_{i2}^L < p_k, \ J_i \in \mathcal{J}_2 \cup \mathcal{J}^*\}. \tag{3.26}$$

If job J_u is located at the r-th position and job J_v at the q-th position in a permutation $\pi_j \in S_j^\pi(T)$ $(r < q-1)$, then job J_k may be located at any position between the r-th and the q-th position. The set of positions $r+1, r+2, \ldots, q-1$ between job J_u and J_v will be called the *candidate area* of job J_k. There are $q-r+1$ positions in the candidate area of job J_k. It is clear that the following claim is correct.

Theorem 3.6 *Let $J_k \in \mathcal{J}_0$, $J_l \in \mathcal{J}_0$, and inequality $p_k \leq p_l$ hold. Then the candidate area of job J_k in the family of solutions $\{S_j^\pi(T)\}$ to problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$ contains the candidate area of job J_l.*

If $\mathcal{J}_0 \neq \emptyset$, then by using Theorem 3.3, one can construct a family of J-solutions $\{S_j^\pi(T)\}$ to problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$ (instead of a single J-solution). It is interesting to note that each job J_k of set $\mathcal{J}_0$ can serve as a buffer to absorb the uncertainties in the processing time of a job on a machine. To illustrate this idea, we consider Example 3.3.

Example 3.3 *We demonstrate how to execute best a schedule and possibly construct an actually optimal schedule for the uncertain problem* $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$ *with the intervals of the job processing times given in Table 3.4. We mean an actually optimal schedule in the sense that even though the processing times are a priori uncertain, the scheduler ends up with executing an optimal schedule as the scheduler has already known the realized values of all uncertain processing times beforehand.*

Table 3.4: *Intervals of the job processing times for Example 3.3*

i	1	2	3	4	5	6	7	8	9	10	11
p_{i1}^L	1	1	2	3	3	3	5	5	5	5	4
p_{i1}^U	1	1	2	3	4	5	5	11	7	8	4
p_{i2}^L	1	2	3	3	5	5	6	10	6	7	3
p_{i2}^U	1	3	3	3	8	8	6	11	7	9	3

There are two phases in the scheduling process: the off-line phase (the schedule planning phase) and the on-line phase (the schedule execution phase). The information about the lower and upper bounds for each uncertain processing time is available at the beginning of the off-line phase while the local information on the realization (the actual value) of each uncertain processing time is available once the corresponding operation is completed. In the off-line phase, a family of solutions $\{S_j^\pi(T)\}$ *is constructed first, which is useful in aiding the scheduler to execute best a schedule during the on-line phase. The subsets of set* J *in the partition* $J = \mathcal{J}_0 \cup \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}^*$ *are as follows:* $\mathcal{J}_0 = \{J_1, J_4\}$, $\mathcal{J}^* = \{J_8, J_9, J_{10}\}$, $\mathcal{J}_1 = \{J_2, J_3, J_5, J_6, J_7\}$, $\mathcal{J}_2 = \{J_{11}\}$. *Using Theorem 3.3, we obtain a partial strict order* $\mathcal{A}_\prec$ *over the set* $J \setminus \mathcal{J}_0$*:*

$$(J_2 \prec J_3 \prec \{J_5, J_6\} \prec J_7 \prec \{J_8, J_9, J_{10}\} \prec J_{11}). \tag{3.27}$$

The partial sequence of (3.27) means that neither the order of jobs J_5 *and* J_6 *is fixable nor the order of jobs* J_8, J_9 *and* J_{10} *is fixable for any J-solution* $S_i^\pi(T)$ *of the family* $\{S_j^\pi(T)\}$*. We now demonstrate how to execute best a schedule and possibly find an optimal schedule from set* $S_i^\pi(T)$ *of the family* $\{S_j^\pi(T)\}$*. Since the order of some jobs in set* $S_i^\pi(T)$ *is not fixable, there does not exist a dominant permutation that remains optimal for all feasible realizations of the job processing times. It is interesting that the scheduler may*

possibly find an actually optimal schedule by making a real-time scheduling decision at each decision-making time-point t_i of the completion time of job J_i on machine M_1 as soon as the exact processing times are available for those operations completed before or at time-point $t_i > t_0 = 0$.

At time-point $t_0 = 0$, either job J_1 or job J_2 may be started on machine M_1 in an optimal way due to the above family of solutions $\{S_j^{\pi}(T)\}$. (As it will be clear later, it is better to process job $J_2 \in \mathcal{J}_1$ first.) Figure 3.1 illustrates the part of the scheduling process, where the candidate set $\{J_1, J_2\}$ of jobs for processing next is indicated at the top.

Hereafter, let $c_1(i)$ and $c_2(i)$ denote the completion time of job $J_i \in J$ by machine M_1 and by machine M_2, respectively. We shall consider the decision-making time-point $t_i = c_1(i)$ at which job J_i is completed on machine M_1 and the scheduler has to decide which job will be processed next on machine M_1. In particular, at time-point $t_2 = c_1(2) = 1$, machine M_1 completes the processing of job J_2 and machine M_2 will start to process this job. Now, the scheduler has to select a job from set $\{J_1, J_3\}$ being processed next on machine M_1. (Again, it will be clear later that it is better to select job $J_3 \in \mathcal{J}_1$ as the next job.) At time-point $t_2 = 1$, machine M_1 starts to process job J_3 for $p_{3,1} = 2$ time units.

At time-point $t_3 = c_1(3) = 3$, machine M_1 completes the processing of job J_3 and the candidate set for processing next on machine M_1 is $\{J_1, J_4, J_5, J_6\}$. At this time-point $t_3 = 3$, machine M_2 still is processing job J_2. The relations $p_{5,1}^U = 4 > p_{3,2}^L = 3$ and $p_{6,1}^U = 5 > p_{3,2}^L = 3$ hold for jobs J_5 and J_6, and the selection of job J_5 or job J_6 for being processed next may cause an idle time on machine M_2. In such a case, the scheduler can select job J_1 from set $\{J_1, J_4, J_5, J_6\}$ for being processed immediately after job J_2. Such a selection of job $J_1 \in \mathcal{J}_0$ will allow the scheduler to delay the decision-making of sequencing jobs J_5 and J_6 until the time-point $t_1 = 3 + 1 = 4$ and thus to collect more realized values of the uncertain job processing times.

Let the realization (actual value) $p_{2,2}^*$ of the processing time $p_{2,2}$ of job J_2 turn out to be equal to $3 = p_{2,2}^*$. (Hereafter, we use the notation p_{ij}^* for the actual job processing time p_{ij}.) Then, at time-point $t_1 = c_1(1) = 4$, machine M_2 finishes the processing of job J_2, and 4 time units are needed to complete the processing of both jobs J_3 and J_1 on machine M_2 (3 time units for processing job J_3 and 1 time unit for processing job J_1). The following inequalities hold: $p_{5,1}^U = 4 \leq 4$, $p_{6,1}^U = 5 > 4$. For job J_5, the relation $p_{5,1}^U + p_{6,1}^U = 4 + 5 \leq p_{5,2}^L + 4 = 5 + 4$ holds. Therefore, jobs J_5 and J_6 can be optimally processed with job J_5 preceding J_6 (since such an order causes no idle time on machine M_2). Then, machine M_1 will process job J_7

immediately after job J_6 (since job $J_4 \in \mathcal{J}_0$ can be used as a buffer to absorb the uncertainties in the processing times later when necessary).

At time-point $t_6 = c_1(6) = 13$, when machine M_1 completes the processing of job J_6, the scheduler already knows all job processing times completing before and at time-point $t_6 = 13$. Let the realized values be as follows: $p^*_{5,1} = 4$, $p^*_{6,1} = 5$, $p^*_{5,2} = 5$.

At time-point $t_7 = c_1(7) = 18$, the scheduler has the choice for the next job to be processed on machine M_1 among the jobs J_4, J_8, J_9 and J_{10}. At time-point t_7, machine M_2 already processed job J_6 for 5 time units, and the scheduler has no sufficient information to optimally select a job from the set $\{J_8, J_9, J_{10}\}$ for processing next due to the fact that relations $p^U_{8,1} = 11 > p^L_{7,2} = 6$, $p^U_{9,1} = 7 > p^L_{7,2} = 6$, $p^U_{10,1} = 8 > p^L_{7,2} = 6$ hold (i.e., any such selection may cause an idle time on machine M_2). Now it is time for the scheduler to select job $J_4 \in \mathcal{J}_0$ for being processed immediately after job J_7 on machine M_1. The role of job $J_4 \in \mathcal{J}_0$ seems like a buffer to absorb the uncertainties of some uncertain job processing times.

At time-point $t_4 = c_1(4) = 18 + 3 = 21$, the scheduler has the choice for processing the next job among the jobs J_8, J_9 and J_{10}. Assuming that $p^*_{6,2} = 8$, we know that J_6 is still under processing at time-point t_7 and is finished just at time-point t_4 on machine M_2. Hence, we obtain equalities $p^*_{4,2} + p^*_{7,2} = 3 + 6 = 9$ and therefore, inequalities $p^U_{8,1} = 11 > 9$, $p^U_{9,1} = 7 < 9$, $p^U_{10,1} = 8 < 9$ hold. In the case when the scheduler selects job J_8 to be processed next, there will be an idle time on machine M_2. Thus, the scheduler can select a job from set $\{J_9, J_{10}\}$ to be processed next. Let us check the following relation for the order of the three jobs (J_i, J_{i+1}, J_{i+2}):

$$p^U_{i1} + p^U_{i+1,1} + p^U_{i+2,1} \leq 8 + p^L_{i2} + p^L_{i+1,2}. \tag{3.28}$$

The results of checking the four orders $\{(J_9, J_8, J_{10}), (J_9, J_{10}, J_8), (J_{10}, J_8, J_9), (J_{10}, J_9, J_8)\}$ are as follows:

$$p^U_{9,1} + p^U_{8,1} + p^U_{10,1} = 11 + 7 + 8 = 26 > 9 + p^L_{9,2} + p^L_{8,2} = 9 + 6 + 10 = 25,$$

$$p^U_{9,1} + p^U_{10,1} + p^U_{8,1} = 11 + 7 + 8 = 26 > 9 + p^L_{9,2} + p^L_{10,2} = 9 + 6 + 7 = 22,$$

$$p^U_{10,1} + p^U_{8,1} + p^U_{9,1} = 11 + 7 + 8 = 26 = 9 + p^L_{10,2} + p^L_{8,2} = 9 + 7 + 10 = 26,$$

$$p^U_{10,1} + p^U_{9,1} + p^U_{8,1} = 11 + 7 + 8 = 26 > 9 + p^L_{10,2} + p^L_{9,2} = 9 + 7 + 6 = 22.$$

Since relation (3.28) holds for the order (J_{10}, J_8, J_9), such an order will cause no idle time on machine M_2. Hence, the scheduler can optimally adopt the order (J_{10}, J_8, J_9) (since this order together with job J_{11} being the last one

will be optimal for any feasible realization of the processing times of the remaining jobs $\{J_8, J_9, J_{10}, J_{11}\}$*). Thus, we obtain the permutation:* $\pi_u = (J_2, J_3, J_1, J_5, J_6, J_7, J_4, J_{10}, J_8, J_9, J_{11})$*, which is necessarily optimal with the following partially realized values of the job processing times (i.e., those for the job set* $\{J_1, J_2, \ldots, J_7\}$*):*

$$p^*_{1,1} = 1,\ p^*_{1,2} = 1,\ p^*_{2,1} = 1,\ p^*_{2,2} = 3,\ p^*_{3,1} = 2,\ p^*_{3,2} = 3,\ p^*_{4,1} = 3, \quad (3.29)$$

$$p^*_{4,2} = 3,\ p^*_{5,1} = 4,\ p^*_{5,2} = 5,\ p^*_{6,1} = 5,\ p^*_{6,2} = 8,\ p^*_{7,1} = 5,\ p^*_{7,2} = 6.$$

The initial part of this schedule is represented in Figure 3.1. Note that the remaining part of this schedule cannot be shown exactly since at time-point $t_4 = 21$*, the processing times of jobs* J_8*,* J_9*,* J_{10} *and* J_{11} *are still unknown. However, what is important, any feasible values of the remaining four jobs will not invalidate the optimality of permutation* π_u*. Thus, in spite of the uncertain job processing times, the scheduler ends up with executing an actually optimal schedule from the family of sets* $\{S^\pi_j(T)\}$*.*

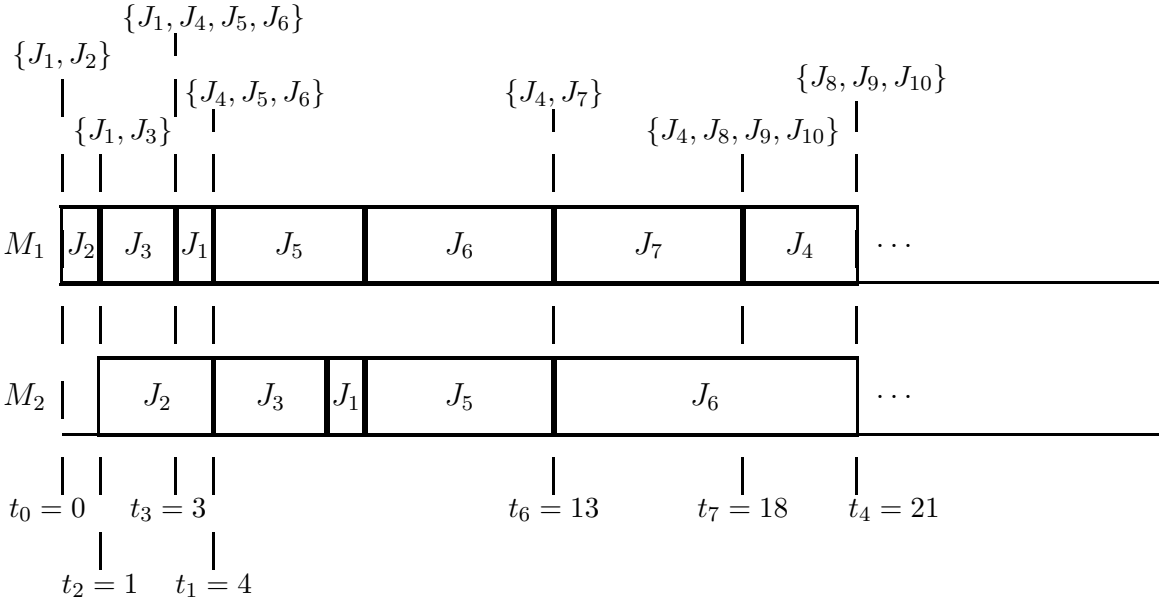


Figure 3.1: *Initial part of an optimal schedule with the processing times of jobs* $\{J_1, J_2, \ldots, J_7\}$ *given in (3.29)*

The above two-phase scheduling process consists of the off-line planning phase with the family of sets $\{S^\pi_i(T)\}$ being constructed using Theorem 3.3, and the on-line execution phase with the following decision-making time-points: $t_2 = 1$, $t_3 = 3$, $t_1 = 4$, $t_6 = 13$, $t_7 = 18$ and $t_4 = 21$. Next, we give the formal arguments of the above optimal scheduling process.

Conditions for Schedule Domination

We first recall Theorem 3.1 (given on page 184) presenting necessary and sufficient conditions for the existence of a permutation $\pi_j \in S^\pi$ that remains optimal for all vectors $p \in T$ of job processing times. We note that condition $(a) - (b)$ of Theorem 3.1 is rarely satisfied in real situations (see Tables 3.2 and 3.3 on page 199). Next, we provide sufficient conditions for the existence of a *dominant* set of permutations in the following sense.

Definition 3.2 *Permutation $\pi_u \in S^\pi$ dominates permutation $\pi_k \in S^\pi$ with respect to T if inequality $C_{max}(\pi_u, p) \leq C_{max}(\pi_k, p)$ holds for any vector $p \in T$ of the job processing times. The set of permutations $S^\pi[T] \subseteq S^\pi$ is called dominant with respect to T if for each permutation $\pi_k \in S^\pi$, there exists a permutation $\pi_u \in S^\pi[T]$ that dominates permutation π_k with respect to T.*

If conditions (a) – (b) of Theorem 3.1 hold, then there exists a singleton $\{\pi_j\}$ which is a J-solution $S^\pi(T)$ for problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$. It is clear that the singleton $\{\pi_j\} = S^\pi(T)$ is dominant with respect to T (we say that such a permutation π_j is dominant with respect to T). In general, the set of permutations $S^\pi(T)$ used in Definition 3.1 (given on page 183) is dominant with respect to T. It should be noted that Definition 3.2 does not exploit the notion of a Johnson permutation in contrast to Definition 3.1. In what follows, we shall relax (if it will be useful) the demand for a dominant permutation π_j to be a Johnson one (see Remark 3.2 given on page 182).

Next, we shall describe and justify sufficient conditions and formal algorithms for constructing a dominant permutation (if it will be possible) for the uncertain problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$. First, we consider the case of an empty set $\mathcal{J}_0$ and then the case of a non-empty set $\mathcal{J}_0$. As far as the on-line scheduling phase is concerned, two cases are distinguished:

(j) Both the actual values p_{i1}^* and p_{i2}^* of the processing times p_{i1} and p_{i2} are available at time-point $t_1 = c_1(i)$ when job J_i is completed by machine M_1;

(jj) The actual value p_{ij}^* of the processing time p_{ij} is available at time-point $t_i = c_j(i)$ when job J_i is completed by machine M_j.

We note that case (jj) is valid for most practical uncertain scheduling problems. Case (j) may occur in some real-world scheduling scenarios. One example of case (j) is that M_1 is a diagnostic machine and M_2 is a repairing machine. During the on-line scheduling phase, once a job J_i is completed on the diagnostic machine M_1, the scheduler usually knows the actual values

(realized values) p^*_{i1} and p^*_{i2} of the processing times p_{i1} and p_{i2} of job J_i on both machines M_1 and M_2. Another example of case (j) is that machine M_1 is used for a *rough processing* of a part $J_i \in \mathcal{J}$ and machine M_2 is used for its *perfect processing*.

On-line Scheduling in Case (j)

First, we consider the case of only two conflicting jobs: $|S^\pi(T)| = 2$.

Let $\mathcal{J}_0 = \emptyset$. Since there are only two permutations in a J-solution $S^\pi(T)$, i.e., $S^\pi(T) = \{\pi_u, \pi_v\}$, it is clear that there exist only two non-adjacent vertices in the digraph $(J, \mathcal{A}_\prec)$ representing the partial strict order defining the J-solution $S^\pi(T) = \{\pi_u, \pi_v\}$. Due to Definition 3.1, permutation π_u (permutation π_v) is an optimal Johnson permutation for at least one feasible vector of the job processing times but surely it is not a Johnson permutation for all feasible vectors $p \in T$ of job processing times (since the condition of Theorem 3.1 holds neither for permutation π_u nor for permutation π_v). W.l.o.g., we can assume that $\pi_u = (J_1,\ J_2, \ldots,\ J_{k-1},\ J_k, J_{k+1}, \ldots, J_n)$ and $\pi_v = (J_1,\ J_2, \ldots,\ J_{k-1},\ J_{k+1},\ J_k, \ldots, J_n)$, i.e., only the orders of jobs J_k and J_{k+1} are different in these two permutations. In what follows, if there is no path connecting vertex J_k with vertex J_{k+1} in digraph $(J, \mathcal{A}_\prec)$, we say that jobs J_k and J_{k+1} are *conflicting*. Since the order $(J_1,\ J_2, \ldots,\ J_{k-1})$ is the same in both permutations π_u and π_v defining the J-solution $S^\pi(T) = \{\pi_u, \pi_v\}$, it is justified to process these jobs just in this order. Let the actual processing of jobs $J_1,\ J_2, \ldots,\ J_{k-1}$ be started (and completed) in the order $J_1 \rightarrow J_2 \rightarrow \ldots \rightarrow J_{k-1}$ on both machines M_1 and M_2.

Since jobs J_k and J_{k+1} are conflicting, an additional decision has to be used at time-point $t_{k-1} = c_1(k-1)$. It is clear that at time-point t_{k-1}, the actual processing times of the jobs from set $\mathcal{J}(t_{k-1}, 1) = \{J_1,\ J_2, \ldots,\ J_{k-1}\}$ on machine M_1 are already known. Let these actual values of the processing times turn out to be as follows: $p_{1,1} = p^*_{1,1},\ p_{2,1} = p^*_{2,1}, \ldots, p_{k-1,1} = p^*_{k-1,1}$. In case (j), the following assumption has to be made.

Assumption 1 *The actual processing times of the jobs from set* $\mathcal{J}(t_{k-1}, 1)$ *on machine* M_2 *are available at time-point* $t_{k-1} = c_1(k-1)$*:* $p_{1,2} = p^*_{1,2},\ p_{2,2} = p^*_{2,2}, \ldots, p_{k-1,2} = p^*_{k-1,2}$.

Thus, at time-point $t_{k-1} = c_1(k-1)$, the following set of feasible vectors of the processing times

$$T(t_{k-1}) = \{p \in T\ :\ p_{ij} = p^*_{ij},\ J_i \in \mathcal{J}(t_{k-1}, 1),\ M_j \in M\} \tag{3.30}$$

will be utilized instead of set T defined by equality (3.1). Next, we consider

the following question. When will one of the permutations π_u or π_v be optimal for all vectors $p \in T(t_{k-1})$ of the job processing times? To answer this question, we have to consider all possible orders of the non-adjacent vertices J_k and J_{k+1} in the digraph $(J, \mathcal{A}_{\prec})$ representing the J-solution $S^{\pi}(T)$. (Recall that digraph $(J, \mathcal{A}_{\prec})$ may be constructed in $O(n^2)$ time.) At time-point t_{k-1}, the scheduler has the choice between jobs J_k and J_{k+1} (which are conflicting) for being processed next (immediately after job J_{k-1}) on machine M_1. Now, the scheduler can test the condition in the following claim.

Theorem 3.7 *If*

$$c_2(k-1) - c_1(k-1) \geq p_{k1}^U, \tag{3.31}$$

$$c_2(k-1) - c_1(k-1) + p_{k2}^L \geq p_{k1}^U + p_{k+1,1}^U, \tag{3.32}$$

then permutation $\pi_u \in S^{\pi}(T) = \{\pi_u, \pi_v\}$ is dominant with respect to $T(t_{k-1})$.

PROOF. For permutation π_u and any vector $p \in T(t_{k-1})$ of the job processing times, we can calculate the earliest starting time $s_2(k+2)$ of job J_{k+2} on machine M_2 as follows: $s_2(k+2) = \max\{c_1(k+2),\ c_2(k+1)\}$. As we consider only semiactive schedules, machine M_1 processes all the jobs without any idle time, so we obtain

$$c_1(k+2) = \sum_{i=1}^{k+2} p_{i1} = c_1(k-1) + p_{k1} + p_{k+1,1} + p_{k+2,1}.$$

For machine M_2, we obtain $c_2(k+1) = p_{k+1,2} + \max\{c_1(k-1) + p_{k1} + p_{k+1,1},\ p_{k,2} + \max\{c_1(k-1) + p_{k1},\ c_2(k-1)\}\}$. Inequality (3.31) implies equality $\max\{c_1(k-1)+p_{k1},\ c_2(k-1)\} = c_2(k-1)$ for any vector $p \in T(t_{k-1})$ of the job processing times. Therefore, we obtain

$$c_2(k+1) = p_{k+1,2} + \max\{c_1(k-1) + p_{k1} + p_{k+1,1},\ p_{k,2} + c_2(k-1)\}. \tag{3.33}$$

Equality (3.33) means that machine M_2 has no idle time while processing jobs J_{k-1} and J_k. Inequality (3.32) implies equality $\max\{c_1(k-1) + p_{k1} + p_{k+1,1},\ p_{k,2} + c_2(k-1)\} = p_{k,2} + c_2(k-1)$. Therefore, equality $c_2(k+1) = p_{k+1,2} + p_{k,2} + c_2(k-1)$ holds. This means that machine M_2 has no idle time while processing jobs J_k and J_{k+1}.

We obtain $s_2(k+2) = \max\{c_1(k-1) + p_{k1} + p_{k+1,1} + p_{k+2,1},\ p_{k+1,2} + p_{k,2} + c_2(k-1)\}$. Due to Assumption 1, the values $c_1(k-1)$ and $c_2(k-1)$ are available at time-point $c_1(k-1)$. As machine M_1 has no idle time while processing jobs from set $\{J_1, J_2, \ldots, J_{k+2}\}$, it is impossible to reduce the value $c_1(k-1) + p_{k1} + p_{k+1,1} + p_{k+2,1}$. Analogously, as machine M_2 has no

idle time while processing jobs from set $\{J_{k-1}, J_k, J_{k+1}\}$, it is impossible to reduce the value $p_{k+1,2}+p_{k,2}+c_2(k-1)$ by an alternative order of the jobs J_k and J_{k+1}. Therefore, permutation π_u dominates permutation π_v with respect to $T(t_{k-1})$ (regardless of the exact value $s_2(k+2)$). Since $S^\pi(T) = \{\pi_u, \pi_v\}$, permutation π_u is dominant with respect to $T(t_{k-1})$.

$\diamond$

Thus, if conditions (3.31) and (3.32) of Theorem 3.7 hold, then the order $J_k \rightarrow J_{k+1}$ of jobs J_k and J_{k+1} is the optimal order of these two jobs in the remaining part of an optimal permutation. Note that in Example 3.3, Theorem 3.7 was implicitly used for sequencing the order of jobs J_5 and J_6 at time-point $t_1 = 4$.

Theorem 3.8 *If* $c_2(k-1) - c_1(k-1) \geq p^U_{k1} + p^U_{k+1,1}$, *then each permutation from set* $S^\pi(T) = \{\pi_u, \pi_v\}$ *is dominant with respect to* $T(t_{k-1})$.

Proof. From condition $c_2(k-1) - c_1(k-1) \geq p^U_{k1} + p^U_{k+1,1}$, we obtain that both inequalities $c_2(k-1) - c_1(k-1) \geq p^U_{k1}$ and $c_2(k-1) - c_1(k-1) + p^L_{k2} \geq p^U_{k1} + p^U_{k+1,1}$ hold. Thus, conditions (3.31) and (3.32) of Theorem 3.7 hold for permutation $\pi_u \in S^\pi(T)$. Hence, permutation π_u is dominant with respect to $T(t_{k-1})$. On the other hand, condition $c_2(k-1) - c_1(k-1) \geq p^U_{k1} + p^U_{k+1,1}$ implies that both inequalities $c_2(k-1) - c_1(k-1) \geq p^U_{k+1,1}$ and $c_2(k-1) - c_1(k-1) + p^L_{k+1,2} \geq p^U_{k1} + p^U_{k+1,1}$ hold, i.e., the appropriate condition of Theorem 3.7 holds for permutation $\pi_v \in S^\pi(T)$ with an alternative order of jobs J_k and J_{k+1}. Hence, permutation π_v is dominant with respect to $T(t_{k-1})$ as well. This completes the proof.

$\diamond$

If the condition of Theorem 3.8 holds, then the order of jobs J_k and J_{k+1} may be arbitrary in the remaining part of an optimal permutation. Similarly, one can prove the following sufficient conditions for the domination of permutation π_u with respect to $T(t_{k-1})$. We omit the proof since it is similar to that of Theorem 3.7.

Theorem 3.9 *If*

$$c_2(k-1) - c_1(k-1) < p^L_{k1}, \tag{3.34}$$

$$p^U_{k+1,1} \leq p^L_{k2}, \tag{3.35}$$

$$p^L_{k+1,1} + p^L_{k+2,2} \geq p^U_{k2} + p^U_{k+1,1}, \tag{3.36}$$

then permutation $\pi_u \in S^\pi(T) = \{\pi_u, \pi_v\}$ *is dominant with respect to* $T(t_{k-1})$.

Thus, if at least one of the conditions of Theorems 3.7 – 3.9 holds, then the scheduler may fix the optimal order of jobs J_k and J_{k+1} regardless of the fact that the actual values of the processing times of the jobs J_k, $J_{k+1}, \ldots,$ J_n are still unavailable. In [237], five other sufficient conditions for the domination of permutation $\pi_u \in S^\pi(T) = \{\pi_u, \pi_v\}$ have been proven.

Next, we show how to generalize Theorems 3.7 – 3.9 for the case when three jobs are conflicting at time-point $t_i > 0$: $|S^\pi(T)| = 6$. Let the jobs from set $\{J_k,\ J_{k+1}, J_{k+2}\} \subset J$ be conflicting at time-point $t_{k-1} > 0$. So, there are six $(3! = 6)$ permutations in the J-solution $S^\pi(T)$. We can test the following conditions similar to those of Theorems 3.7 – 3.9 and find a dominant permutation with respect to $T(t_{k-1})$.

Theorem 3.10 *Let the strict order relation $\mathcal{A}_\prec$ over set $J = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ be as follows: $(J_1 \prec \ldots \prec J_{k-1} \prec \{J_k, J_{k+1},\ J_{k+2}\} \prec J_{k+3} \prec \ldots \prec J_n)$. If $c_2(k-1) - c_1(k-1) > p^U_{k1}$, $c_2(k-1) - c_1(k-1) + p^L_{k2} > p^U_{k1} + p^U_{k+1,1}$ and $c_2(k-1) - c_1(k-1) + p^L_{k2} + p^L_{k+1,2} > p^U_{k1} + p^U_{k+1,1} + p^U_{k+2,1}$, then permutation $(J_1, \ldots, J_k, J_{k+1}, J_{k+2}, \ldots, J_n)$ is dominant with respect to $T(t_{k-1})$.*

Proof. Arguing similarly as in the proof of Theorem 3.7, we conclude that machine M_1 has no idle time while processing the jobs from set $\{J_1, J_2, \ldots, J_{k+3}\}$. Thus, it is impossible to reduce the value $c_1(k+3)$ obtained for permutation $\pi_w = (J_1, \ldots, J_k, J_{k+1}, J_{k+2}, \ldots, J_n)$. Analogously, machine M_2 has no idle time while processing the jobs $\{J_{k-1}, J_k, J_{k+1}, J_{k+2}\}$ in the order defined by permutation π_w. Thus, it is impossible to reduce the value $c_2(k+2)$ defined for permutation π_w by an alternative order of the jobs J_k, J_{k+1} and J_{k+2}. Therefore, if the condition of Theorem 3.10 holds, then permutation $\pi_w \in S^\pi(T)$ is dominant with respect to $T(t_{k-1})$ (regardless of the unknown value $s_2(k+3) = \max\{c_1(k+3),\ c_2(k+2)\}$).

$\diamond$

If the condition of Theorem 3.10 holds, then in the remaining part of an optimal permutation, the order of the jobs J_k, J_{k+1} and J_{k+2} is as follows: $J_k \rightarrow J_{k+1} \rightarrow J_{k+2}$. We can test the six conditions analogous to those of Theorem 3.10 but for different orders of three conflicting jobs. In Example 3.3, Theorem 3.10 was implicitly used for sequencing the jobs J_8, J_9 and J_{10} at time-point $t_4 = 21$.

Similarly to the proof of Theorem 3.8, we can prove sufficient conditions for the existence of six dominant permutations as follows.

Theorem 3.11 *Let the strict order relation $\mathcal{A}_\prec$ over set $J = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ be as follows: $(J_1 \prec \ldots \prec J_{k-1} \prec \{J_k, J_{k+1},\ J_{k+2}\} \prec J_{k+3} \prec \ldots \prec J_n)$. If*

$c_2(k-1)-c_1(k-1) > p^U_{k1}+p^U_{k+1,1}+p^U_{k+2,1}$, *then each of the six permutations from set* $S^\pi(T)$ *is dominant with respect to* $T(t_{k-1})$.

If the condition of Theorem 3.11 holds, then the order of the three jobs J_k, J_{k+1} and J_{k+2} may be arbitrary in the remaining part of an optimal permutation. Theorem 3.9 may be also generalized, and we can obtain fourteen sufficient conditions for the existence of a dominant permutation when $|S^\pi(T)| = 6$. These conditions have been proven in [237]. Next, we present one of them.

Theorem 3.12 *Let the strict order relation* $\mathcal{A}_\prec$ *over set* $J = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ *be as follows:* $(J_1 \prec \ldots \prec J_{k-1} \prec \{J_k, J_{k+1}, J_{k+2}\} \prec J_{k+3} \prec \ldots \prec J_n)$. *If* $c_2(k-1) - c_1(k-1) < p^L_{k1}$, $p^L_{k2} \geq p^U_{k+1,1}$, $p^L_{k2} + p^L_{k+1,2} \geq p^U_{k+1,1} + p^U_{k+2,1}$ *and* $p^L_{k+1,1} + p^L_{k+2,1} + p^L_{k+3,1} \geq p^U_{k2} + p^U_{k+1,2} + p^U_{k+2,2}$, *then permutation* $\{J_1, \ldots, J_k, J_{k+1}, J_{k+2}, \ldots, J_n\}$ *is dominant with respect to* $T(t_{k-1})$.

Thus, if at least one of the sufficient conditions of Theorems 3.10 – 3.12 holds, then the order of the jobs J_k, J_{k+1} and J_{k+2} must be $J_k \rightarrow J_{k+1} \rightarrow J_{k+2}$ in the remaining part of an optimal permutation. We can also test the above conditions for the five other possible orders of the conflicting jobs J_k, J_{k+1} and J_{k+2}.

Next, we consider the general case of a J-solution $S^\pi(T)$. It is clear that Theorems 3.7 – 3.9 (Theorems 3.10 – 3.12, respectively) may be used if more than two (six) permutations are in the set $S^\pi(T)$ provided that, at each time-point of the schedule execution, no more than two (three) jobs from set J are conflicting. We demonstrate this on the following example which is appropriate for the use of Theorems 3.7 – 3.9.

Example 3.4 *Let* $|S^\pi(T)| = 8$ *and six jobs from set* $J = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ *be conflicting in a pairwise manner, e.g., the pair of jobs* J_k *and* J_{k+1} *are conflicting, the pair of jobs* J_l *and* J_{l+1}, *and the pair of jobs* J_m *and* J_{m+1}. *Then we can use Theorems 3.7 – 3.9 for each pair of jobs that are conflicting. W.l.o.g. we assume that the partial strict order* $\mathcal{A}_\prec$ *over set* $J = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ *is as follows:* $(J_1 \prec \ldots \prec J_{k-1} \prec \{J_k, J_{k+1}\} \prec J_{k+2} \prec \ldots \prec J_{l-1} \prec \{J_l, J_{l+1}\} \prec J_{l+2} \prec \ldots \prec J_{m-1} \prec \{J_m, J_{m+1}\} \prec J_{m+2} \prec \ldots \prec J_n)$.

First, we process the jobs $\{J_1, \ldots, J_{k-1}\}$ *in the optimal order* $J_1 \rightarrow \ldots \rightarrow J_{k-1}$. *At time-point* $t_{k-1} = c_1(k-1)$, *we test Theorems 3.7 – 3.9 for the pair of jobs* $\{J_k, J_{k+1}\}$ *that are conflicting. If at least one of the conditions of Theorems 3.7 – 3.9 holds for the order* $J_k \rightarrow J_{k+1}$, *then we process the jobs* $\{J_k, \ldots, J_{l-1}\}$ *in the order* $J_k \rightarrow \ldots \rightarrow J_{l-1}$. *At time-point* $t_{l-1} = c_1(l-1)$, *we test Theorems 3.7 – 3.9 for the pair of conflict jobs* $\{J_l, J_{l+1}\}$. *If at least*

one of the conditions of Theorems 3.7 – 3.9 holds for the order $J_l \rightarrow J_{l+1}$, *then we process the jobs* $\{J_l, \ldots, J_{m-1}\}$ *in the order* $J_l \rightarrow \ldots \rightarrow J_{m-1}$. *At time-point* $t_{m-1} = c_1(m-1)$, *we test Theorems 3.7 – 3.9 for the pair of jobs* $\{J_m, J_{m+1}\}$ *that are conflicting. If at least one of the conditions of Theorems 3.7 – 3.9 holds for the order* $J_m \rightarrow J_{m+1}$, *then we process the jobs* $\{J_m, \ldots, J_n\}$ *in the order* $J_m \rightarrow \ldots \rightarrow J_n$.

Therefore, if the condition of at least one of Theorems 3.7 – 3.9 holds for the pairs of jobs $\{J_k, J_{k+1}\}$, $\{J_l, J_{l+1}\}$, *and* $\{J_m, J_{m+1}\}$, *then we obtain a dominant permutation* $\pi_g = (J_1, \ldots, J_{k-1}, J_k, J_{k+1}, J_{k+2}, \ldots, J_{l-1}, J_l, J_{l+1}, J_{l+2}, \ldots, J_{m-1}, J_m, J_{m+1}, J_{m+2}, \ldots, J_n)$ *with respect to* $T(t_{m-1})$ *(this permutation will be optimal for the actual job processing times). Otherwise, if no condition of Theorem 3.7 – 3.9 holds for a pair of jobs* $\{J_r, J_{r+1}\}$, $r \in \{k, l, m\}$, *(say* $r = m$*), then we obtain a two-element dominant set of permutations* $\{\pi_g, \pi_h\}$, *where* $\pi_h = (J_1, \ldots, J_{k-1}, J_k, J_{k+1}, J_{k+2}, \ldots, J_{l-1}, J_l, J_{l+1}, J_{l+2}, \ldots, J_{m-1}, J_{m+1}, J_m, J_{m+2}, \ldots, J_n)$ *without a proof that one of the permutations* π_h *or* π_g *dominates the other one.*

Furthermore, we can generalize the above sufficient conditions for the case when an arbitrary number of jobs are conflicting at the same on-line decision-making time-points. Let the set of r jobs be conflicting at time-point $t_k = c_1(k) \geq 0$. W.l.o.g. we assume that the jobs from the set $\{J_{k_1}, J_{k_2}, \ldots, J_{k_r}\} \subset J = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ are conflicting. Then we need to test $r!$ possible orders of conflicting jobs. A generalization of Theorems 3.7 and 3.10 looks as follows.

Theorem 3.13 *Let the strict order* $\mathcal{A}_\prec$ *over the set* $J = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ *be as follows:* $(J_1 \prec \ldots \prec J_k \prec \{J_{k_1}, J_{k_2}, \ldots, J_{k_r}\} \prec J_{k+1} \prec \ldots \prec J_n)$. *If inequality*

$$\sum_{i=1}^{s+1} p^L_{k_i 1} \leq \sum_{j=0}^{s} p^U_{k_j 2}$$

holds for each $s = 0, 1, \ldots, r$, *where* $p^U_{k_0 2} = c_2(k) - c_1(k)$, *then permutation* $\{J_1, \ldots, J_k, J_{k_1}, J_{k_2}, \ldots, J_{k_r}, J_{k+1}, \ldots, J_n\}$ *is dominant with respect to* $T(t_k)$.

A generalization of Theorems 3.9 and 3.12 looks as follows.

Theorem 3.14 *Let the strict order* $\mathcal{J}_\prec$ *over the set* $J = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ *be as follows:* $(J_1 \prec \ldots \prec J_k \prec \{J_{k_1}, J_{k_2}, \ldots, J_{k_r}\} \prec J_{k+1} \prec \ldots \prec J_n)$. *If the condition*

$$\sum_{i=m}^{s} p^L_{k_i 1} > \sum_{j=m-1}^{s-1} p^U_{k_j 2},\ m = 1, 2, \ldots, s,$$

$$\sum_{i=s+1}^{m} p^U_{k_i 1} \le \sum_{j=s}^{m-1} p^L_{k_j 2},\ m = s+1, s+2, \ldots, r, \qquad \sum_{i=s+1}^{r+1} p^L_{k_i 1} \ge \sum_{j=s}^{r} p^U_{k_j 2}$$

holds, where $p^U_{k_0 2} = c_2(k) - c_1(k)$, *then permutation* $\{J_1, \ldots, J_k, J_{k_1}, J_{k_2}, \ldots, J_{k_r}, J_{k+1}, \ldots, J_n\}$ *is dominant with respect to* $T(t_k)$.

On-line Scheduling in Case (jj)

Let us consider the case (jj). Now, the actual value p^*_{j2} of processing time p_{j2} of job $J_j \in \mathcal{J}(t_{l-1}, 2) = \{J_1, J_2, \ldots, J_{l-1}\}$ is available at time-point $t_{k-1} = c_1(k-1) > c_2(l-1)$, i.e., $p_{j2} = p^*_{j2}$, while the actual value of processing time p_{k2} of job $J_k \in \{J_l, J_{l+1}, \ldots, J_n\}$ is unavailable at time-point $t_{k-1} = c_1(k-1) < c_2(l)$. Thus, at time-point $t_{k-1} = c_1(k-1)$, the following set of feasible vectors

$$T(t_{k-1}, l) = \{p \in T :\ p_{i1} = p^*_{i1},\ p_{j2} = p^*_{j2},\ J_i \in \mathcal{J}(t_{k-1}, 1),\ J_j \in \mathcal{J}(t_{l-1}, 2)\}$$

of job processing times will be used instead of set $T(t_{k-1})$ defined in (3.30).

Since Assumption 1 is not valid in case (jj), now we are forced to exploit the lower bounds $p^L_{l2},\ p^L_{l2}, \ldots, p^L_{k-1,2}$ instead of the actual values p^*_{l2}, $p^*_{l2}, \ldots, p^*_{k-1,2}$ since the latter are unavailable at time-point $t_{k-1} = c_1(k-1)$. As a result, we can calculate the lower bound $c^L_2(k-1)$ for the actual value $c_2(k-1)$ in the following way (see Figure 3.2):

$$c^L_2(k-1) = c_2(l-1) + \max\{p^L_{l2},\ c_1(k-1) - c_2(l-1)\} + \sum_{j=l}^{k-1} p^L_{j2}.$$

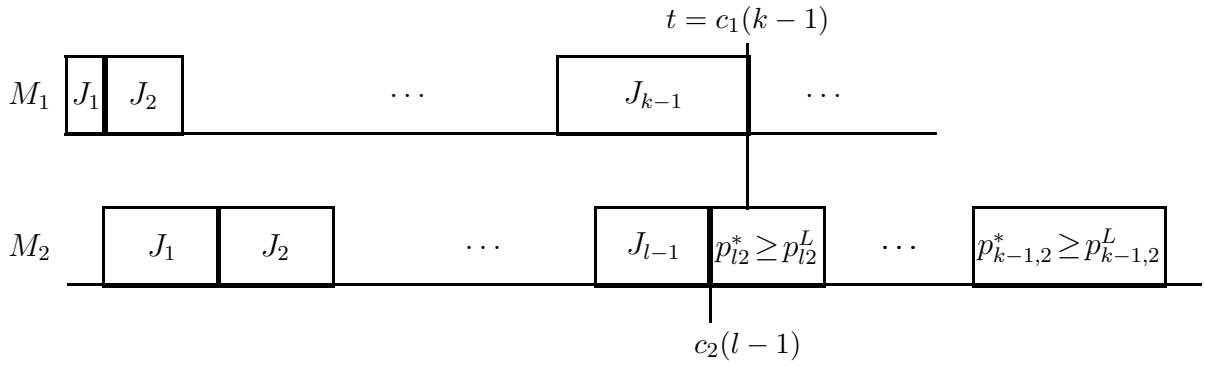


Figure 3.2: *Initial part of an optimal schedule for the jobs of set* $\{J_1, J_2, \ldots, J_{k-1}\}$

The analog of Theorem 3.7 is as follows.

Theorem 3.15 *If* $c_2^L(k-1)-c_1(k-1) \geq p_{k1}^U$ *and* $c_2^L(k-1)-c_1(k-1)+p_{k2}^L \geq p_{k1}^U+p_{k+1,1}^U$, *then permutation* $\pi_u \in S^*(T) = \{\pi_u, \pi_v\}$ *is dominant with respect to* $T(k,l)$.

We can calculate the following upper bound $c_2^U(k-1)$ for the actual value $c_2(k-1)$:

$$c_2^U(k-1) = c_2(l-1) + \sum_{j=l}^{k-1} p_{j2}^U.$$

Thus, the sufficient condition (3.34) – (3.36) from Theorem 3.9 can be reformulated as follows.

Theorem 3.16 *If* $c_2^U(k-1) - c_1(k-1) < p_{k1}^L$, $p_{k+1,1}^U \leq p_{k2}^L$ *and* $p_{k+1,1}^L + p_{k+2,2}^L \geq p_{k2}^U + p_{k+1,1}^U$, *then permutation* $\pi_u \in S^*(T) = \{\pi_u, \pi_v\}$ *is dominant with respect to* $T(k,l)$.

Theorems 3.10 – 3.14 can be reformulated for case (jj) in a similar way.

Dominant Permutation in Off-Line Scheduling

We show that in the off-line scheduling phase, claims similar to Theorems 3.7 – 3.16 can also be applied along with Theorem 3.1 (given on page 184). Recall that Theorem 3.1 provides a necessary and sufficient condition for the existence of a Johnson permutation that is dominant with respect to T. Due to a relaxation in the requirement that the permutation π_u is a Johnson one, we can obtain another sufficient condition for the existence of a dominant permutation. To this end, it is necessary to substitute the exact difference $c_2(k-1) - c_1(k-1)$ (which is unavailable before time-point $t_0 = 0$) by its lower bound. It is clear that for the off-line scheduling phase, there is no difference between case (j) and case (jj).

Let the strict order $\mathcal{A}_{\prec}$ defining the J-solution $S^{\pi}(T)$ look as follows:

$$(J_1 \prec \ldots \prec J_{k-1} \prec \{J_k, J_{k+1}\} \prec \ldots). \tag{3.37}$$

Then jobs J_k and J_{k+1} can be started on machine M_1 at time-point $t_{k-1} = c_1(k-1)$, and machine M_2 is available to process one of the jobs J_k or J_{k+1} from time-point $c_2(k-1)$. If at time-point $t \leq t_0 = 0$, the scheduler can calculate a lower bound Δ_{k-1} for the exact difference $c_2(k-1) - c_1(k-1)$, then before beginning the execution of a schedule, the scheduler can test the conditions of Theorems 3.7 – 3.16 using the value Δ_{k-1} instead of the difference $c_2(k-1) - c_1(k-1)$ unavailable at time-point t.

Next, we show how to calculate a tight lower bound Δ_{k-1}. If inclusion $J_i \in \mathcal{J}_1$ holds for $i = 1, 2, \ldots, k-1$, then for each index $i \in \{1, 2, \ldots, k-1\}$,

inequality $p_{i1}^U \leq p_{i2}^L$ must hold and so $p_{i2} - p_{i1} \geq p_{i2}^L - p_{i1}^U \geq 0$. Thus, the following inequalities give a tight lower bound Δ_{k-1} for the difference $c_2(k-1) - c_1(k-1)$:

$$c_2(k-1) - c_1(k-1) \geq p_{11} + \sum_{i=1}^{k-1} p_{i2} - \sum_{i=1}^{k-1} p_{i1} \geq p_{11}^L + \sum_{i=1}^{k-1} (p_{i2}^L - p_{i1}^U) = \Delta_{k-1}. \tag{3.38}$$

In the opposite case (when $J_i \notin \mathcal{J}_1$), a lower bound Δ_{k-1} for the difference $c_2(k-1) - c_1(k-1)$ may be calculated recursively as follows. If $|\mathcal{J}_1| = m$, we obtain

$$\Delta_m = \sum_{i=1}^{m} (p_{i2}^L - p_{i1}^U) + p_{11}^L$$

due to the last equality in (3.38) with $k-1 = m$. Furthermore, for each index $l \in \{m+1, m+1, \ldots, k-1\}$, one can use the following recursive formula: $\Delta_l = \max\{0, \ \Delta_{l-1} - p_{l,1}^U\} + p_{l,2}^L$. As a result, we obtain the following claim similar to Theorem 3.7.

Theorem 3.17 *If $\Delta_{k-1} \geq p_{k1}^U$ and $\Delta_{k-1} + p_{k2}^L \geq p_{k1}^U + p_{k+1,1}^U$, then permutation $\pi_u \in S^*(T) = \{\pi_u, \pi_v\}$ is dominant with respect to T.*

Furthermore, all the theorems presented in the previous part of this section can be reformulated for the case of off-line scheduling provided that the exact difference $c_2(k-1) - c_1(k-1)$ is substituted by the lower bound Δ_{k-1}. Note that Theorems 3.7 – 3.17 may be only used if $k > 1$ in the partial strict order (3.37). Let $k = 1$ and jobs J_1 and J_2 be conflicting, i.e., the partial strict order (3.37) is as follows: $(\{J_1, J_2\} \prec J_3 \prec \ldots)$. We shall try to sequence two conflicting jobs in an optimal way before time-point $t_0 = 0$. Let us consider the case when machine M_2 has an idle time before processing job J_3. In this case, machine M_2 can process job J_3 from the time-point when machine M_1 completes the processing of this job (i.e., from time-point $t_3 = c_1(3)$). It is easy to prove the following sufficient condition.

Theorem 3.18 *If $p_{3,1}^L \geq p_{2,2}^U + \max\{0, \ p_{1,2}^U - p_{2,1}^L\}$, then the order of jobs J_1 and J_2 in an optimal permutation is $J_1 \rightarrow J_2$.*

Obviously, $c_2(2) - c_1(2) \leq p_{2,2}^U + \max\{0, \ p_{1,2}^U - p_{2,1}^L\}$. In the latter inequality, the difference $p_{1,2}^U - p_{2,1}^L$ is equal to the maximal addition for the case when machine M_2 cannot finish job J_1 before machine M_1 has finished job J_2. Hence, we obtain inequality $c_1(3) > c_2(2)$, and machine M_2 has an idle time before processing job J_3. Therefore, in the opposite case (when the optimal order of jobs J_1 and J_2 cannot be defined by Theorem

3.18), we cannot decrease the makespan. Of course, if $p^L_{3,1} > p^U_{2,2} + p^U_{1,2}$, then both permutations $\pi_u = (J_1, J_2, J_3, \ldots)$ and $\pi_v = (J_2, J_1, J_3, \ldots)$ are dominant and the optimal order of jobs J_1 and J_2 may be arbitrary. More precisely, if $p^L_{3,1} > \max\{p^U_{1,2} + p^U_{2,2} - \min\{p^L_{1,1}, p^L_{1,2}\},\ \max\{p^U_{1,2}, p^U_{2,2}\}\}$, then both permutations π_u and π_v are dominant. In other words, if $p^L_{3,1} > \max\{0,\ p^U_{1,2} - p^L_{2,1},\ p^U_{2,2} - p^L_{1,1}\} + \max\{p^U_{1,2}, p^U_{2,2}\}$, then both permutations π_u and π_v are dominant (an arbitrary order of jobs J_1 and J_2 is optimal).

It is easy to see that the above theorems can be generalized to the case when more than two jobs are conflicting at time-point $t_0 = 0$. Next, we demonstrate such a generalization with two examples of possible claims. In particular, for three conflicting jobs, we obtain the following claim.

Theorem 3.19 *Let the strict order $\mathcal{A}_\prec$ look as follows: $(\{J_1, J_2, J_3\} \prec J_4 \prec \ldots)$. If $p^L_{4,1} \geq p^U_{3,2} + \max\{0,\ p^U_{2,2} - p^L_{3,1} + \max\{0,\ p^U_{1,2} - p^L_{2,1}\}\}$, then the order of jobs J_1, J_2 and J_3 in an optimal permutation is $J_1 \rightarrow J_2 \rightarrow J_3$.*

Let δ_m be defined recursively as follows: $\delta_m = \max\{0,\ p^U_{m,2} - p^L_{m+1,1} + \delta_{m-1}\}$, where $\delta_1 = \max\{0,\ p^U_{1,2} - p^L_{2,1}\}$. Using this notation, we can generalize Theorems 3.18 and 3.19 to the case of r conflicting jobs at time-point $t_0 = 0$.

Theorem 3.20 *Let the strict order $\mathcal{A}_\prec$ look as follows: $(\{J_1, J_2, \ldots, J_r\} \prec J_{r+1} \prec \ldots)$. If $p^L_{r+1,1} \geq p^U_{r,2} + \delta_{r-1}$, then the order of jobs J_1, J_2, $\ldots$, J_r in an optimal permutation is $J_1 \rightarrow J_2 \rightarrow \ldots \rightarrow J_r$.*

Algorithms and Computational Results

Our computational study of the two-phase scheduling was performed on a large number of randomly generated problems $\mathcal{F}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}/\mathcal{C}_{max}$. The following algorithms were coded in C: Algorithm J-*SOL* for the off-line scheduling problem and Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$) (Algorithm *ON-LINE*, respectively) for the on-line scheduling problem provided that set $\mathcal{J}_0$ is empty (non-empty).

Algorithm J-*SOL*

Input: Lower and upper bounds p^L_{ij} and p^U_{ij} for the processing times p_{ij} of jobs $J_i \in J$ on machines $M_j \in M$.

Output: a J-solution $S^\pi(T)$ to problem $\mathcal{F}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}/\mathcal{C}_{max}$; a binary relation $\mathcal{A}_\preceq$ defining a J-solution $S^\pi(T)$, if $|S^\pi(T)| > 1$.

Step 1: Test conditions $a) - b)$ of Theorem 3.1.
Step 2: **IF** conditions $a) - b)$ hold **GOTO** step 8.
Step 3: Using Theorem 3.3 construct the digraph $(J', \mathcal{A}_{\preceq})$ with the vertex set $J' = J \setminus \mathcal{J}_0$.
Step 4: Construct the binary relation $\mathcal{A}_{\preceq}$ by adding set $\mathcal{J}_0$ to digraph $(J', \mathcal{A}_{\preceq})$.
Step 5: Test the conditions of Theorems 3.17 – 3.20.
Step 6: **IF** there are no conflicting jobs **GOTO** step 8.
Step 7: **STOP**
Step 8: **STOP:** a dominant permutation with respect to T.

In Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$), integer $k,\ k \leq n$, denotes the number of decision-making time-points $t_i = c_1(i),\ J_i \in J$, in the on-line scheduling phase. The integer $m,\ m \leq k$, denotes the number of decision-making time-points for which the optimal orders of the conflicting jobs were found using the sufficient conditions from Theorems 3.7 – 3.17.

Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$)

Input: Lower and upper bounds p_{ij}^L and p_{ij}^U for the processing times p_{ij} of the jobs $J_i \in J$ on the machines $M_j \in M$; a J-solution $S^{\pi}(T), |S^{\pi}(T)| > 1$, to problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$ defined by the strict order $\mathcal{A}_{\prec}$.
Output: Either a dominant permutation π_u with respect to $T(i)$, or a permutation $\pi_u \in S^{\pi}(T)$ without its optimality proof.

Step 1: Set $k = 0$, $m = 0$.
Step 2: **IF** the first jobs in the strict order $\mathcal{A}_{\prec}$ are conflicting **THEN BEGIN**
check the condition of Theorem 3.20 for all orders of conflicting jobs
IF the condition of Theorem 3.20 holds for at least one order of conflicting jobs
THEN $k := k + 1$, $m := m + 1$, choose an optimal order of the conflicting jobs **ELSE** $k := k + 1$, choose an arbitrary order of the conflicting jobs and process the conflicting jobs in the chosen order.
END
Step 3: **UNTIL** the last job in the actual schedule is finished.
Step 4: Process the linear part of the partial strict order $\mathcal{A}_{\prec}$.

Step 5: Check the conditions of Theorems 3.13 – 3.14 for all orders of conflicting jobs. **IF** there are two conflicting jobs **THEN** check the conditions of Theorem 3.9.

Step 6: **IF** at least one sufficient condition of Theorems 3.7 – 3.16 holds for at least one order of conflicting jobs **THEN** $k := k + 1$, $m := m + 1$, choose an optimal order of the conflicting jobs **ELSE** $k := k + 1$, choose an arbitrary order of the conflicting jobs.

Step 7: Process the conflicting jobs in the chosen order.

Step 8: **RETURN**

Step 9: **IF** $k = m$ **THEN GOTO** step 14.

Step 10: Calculate the length $\mathcal{C}_{\max}$ of the schedule that was constructed via steps 1–9 and the length $\mathcal{C}^*_{\max}$ of the optimal schedule constructed for the actual processing times.

Step 11: **IF** $\mathcal{C}_{\max} = \mathcal{C}^*_{\max}$ **THEN GOTO** step 13.

Step 12: **STOP:** The constructed schedule is not optimal for the actual processing times.

Step 13: **STOP:** The optimality of the actual permutation is defined after the schedule execution.

Step 14: **STOP:** The optimality of the actual permutation is proven before the schedule execution.

If $\mathcal{J}_0 \neq \emptyset$, then Algorithm *ON-LINE* has to be used instead of Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$) at the on-line scheduling phase. The former differs from the latter by the following part which has to be used instead of the above steps 5–7. Moreover, in Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$), the set $S^\pi(T)$ will be substituted by $S^\pi(T)$. Let N_j denote the subset of set $\mathcal{J}_0$ of the jobs that can be processed at time-point $t_j = c_1(j)$.

Part of Algorithm *ON-LINE*

Step 5: Check the conditions of Theorems 3.13 – 3.14 for all orders of conflicting jobs. **IF** there are only two conflicting jobs at time point t_i **THEN** check the condition of Theorem 3.9.

Step 5a: **IF** no sufficient condition holds **THEN** calculate the corresponding subset N_j.

Step 5b: **UNTIL** $N_j = \emptyset$ **OR** at least one sufficient condition from Theorems 3.7 – 3.16 holds for at least one order of conflicting jobs.

Step 5c: Process job $J_i \in N_j$ with the largest p_i and delete job J_i from set N_j.
Step 6: **IF** at least one of the sufficient conditions holds for at least one order of the conflicting jobs **THEN** $k := k + 1$, $m := m + 1$, choose an optimal order of the conflicting jobs **ELSE** $k := k + 1$, choose an arbitrary order of the conflicting jobs.
Step 7: Process the conflicting jobs in the chosen order.
Step 7a: **RETURN**

For the experiments, we used a Celeron 1200 MHz processor with 384 MB main memory. We made 100 tests in each series, i.e., for each combination of n and L, where L defines the percentage of the relative error of the input data (job processing times) known before scheduling. The lower bound p_{ij}^L and the upper bound p_{ij}^U for the job processing times are uniformly distributed in the range $[10, 1000]$ in such a way that the following equality holds:

$$L = ((p_{ij}^U - p_{ij}^L) : (p_{ij}^U + p_{ij}^L)/2) \cdot 100.$$

The bounds p_{ij}^L and p_{ij}^U and the actual processing times p_{ij} were decimal fractions with two digits after the decimal point. The generator from [351] has been used for (pseudo)randomly generating instances of problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$. It is easy to see that all sufficient conditions proven in this section may be tested in polynomial time in the number n of jobs. Moreover, to minimize the running time of the above three algorithms, these sufficient conditions were tested in an increasing order of their complexity up to the first positive answer (if any) to the following question. Does a dominant permutation exist at time-point $t_i = c_1(i)$?

In the experiments, the CPU-time was rather small. Even for $n = 1000$ jobs, Algorithm J-*SOL*, Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$) and Algorithm *ON-LINE* take less than 0.05 seconds for solving one instance of problem $\mathcal{F}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\mathcal{C}_{max}$. Therefore, we do not present the CPU-time in Tables 3.5 – 3.9. Note that the results presented in Tables 3.5 – 3.9 have been only obtained for the case (jj) of on-line scheduling, i.e., Assumption 1 was not used and so the actual value of processing time p_{ij} became only known at time point $t_i = c_j(i)$ when job J_i was completed by machine M_j.

Tables 3.5 – 3.7 present the percentage of small problem instances which were solved exactly or approximately in the off-line phase (by Algorithm J-*SOL*) and in the on-line phase (by Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$) or Algorithm *ON-LINE*) in spite of the uncertain numerical input data. Column 1 (column

Table 3.5: *Percentage of solved instances with an empty set* $\mathcal{J}_0$

n	L (%)	Number of decision points	Percentage of proved decisions	Off-line optimal (%)	On-line optimal (%)	Optimal without proof (%)	Max error of $\mathcal{C}_{max}$ (%)	Average error of $\mathcal{C}_{max}$ (%)
	1	19	73.68	82	13	3	6.230213	0.092163
	2	39	69.23	65	24	7	14.894381	0.302439
	3	54	64.81	52	31	15	7.330572	0.129230
	4	71	73.24	41	43	10	11.425118	0.186663
10	5	71	71.83	42	39	14	18.172153	0.332733
	6	93	70.97	27	49	18	12.438417	0.275810
	7	100	74.00	29	47	15	17.770338	0.274215
	8	121	66.94	19	51	21	24.294342	0.379896
	9	125	56.80	10	44	32	16.657515	0.952808
	10	130	67.69	16	50	24	18.044373	0.673897
	1	71	85.92	47	43	9	1.857141	0.018571
	2	126	91.27	17	72	9	14.779399	0.163159
	3	155	87.74	14	70	15	0.022465	0.000225
	4	211	90.05	4	76	18	7.927369	0.079337
20	5	238	84.87	1	70	22	10.647840	0.370658
	6	237	81.43	1	65	25	7.414827	0.277665
	7	288	84.03	0	66	28	7.479012	0.114603
	8	250	78.80	0	64	32	12.671661	0.314745
	9	294	82.31	0	60	25	10.750363	0.537575
	10	303	81.85	0	59	28	8.804494	0.366674
	1	134	97.76	26	71	3	0.000000	0.000000
	2	241	89.63	10	71	18	0.004085	0.000041
	3	319	93.73	0	83	13	3.760090	0.067121
	4	347	95.10	0	86	13	1.603097	0.016031
30	5	413	94.43	0	80	19	11.283378	0.112834
	6	390	88.46	0	69	23	6.422380	0.222811
	7	448	90.18	0	70	27	10.415929	0.198046
	8	450	90.67	0	69	26	0.192515	0.002738
	9	440	87.73	0	64	25	15.253723	0.399358
	10	446	89.01	0	67	26	11.338615	0.119900
	1	236	96.19	4	88	8	0.000000	0.000000
	2	421	94.54	0	84	14	0.269543	0.002735
40	3	484	96.69	0	88	11	9.348538	0.093485
	4	559	93.74	0	72	23	6.632380	0.101148
	5	581	95.87	0	83	16	1.854262	0.018543
	10	526	90.68	0	68	30	7.492048	0.074967
50	5	764	94.24	0	74	23	4.768900	0.047914
	10	616	89.29	0	60	31	3.341783	0.125370
60	5	889	93.70	0	63	32	8.621157	0.136614
	10	704	92.33	0	64	29	8.556119	0.250279
70	5	954	96.02	0	76	21	1.219268	0.012251
	10	716	92.18	0	63	31	14.920141	0.230028
80	5	1086	95.67	0	68	32	0.000000	0.000000
	10	784	91.20	0	63	30	6.311226	0.078879
90	5	1201	95.50	0	66	29	3.027027	0.048897
	10	776	90.08	0	51	37	15.321626	0.302109
100	5	1259	96.43	0	74	25	0.001865	0.000019
	10	750	89.73	0	50	37	5.903523	0.169706

2) presents the number of jobs n, $10 \leq n \leq 100$ (the relative error of the input data L, $1 \leq L \leq 10$, in percentage).

Column 3 presents the number of sets of conflicting jobs in the strict order $\mathcal{A}_{\prec}$ for Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$) with $\mathcal{J}_0 = \emptyset$ (in the binary relation $\mathcal{A}_{\preceq}$ for Algorithm *ON-LINE* with $\mathcal{J}_0 \neq \emptyset$) for which the decisions were made in the decision-making time-points $t_i = c_1(i)$. (In the above description

of Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$) and Algorithm *ON-LINE*, this number is denoted as k.) The percentage of the correct decisions made due to Theorem 3.1 and Theorems 3.17 – 3.20 in the off-line scheduling phase and the correct decisions made due to Theorems 3.7 – 3.16 in the on-line scheduling phase are given in column 4. This number is equal to $m/k \cdot 100\%$, where m and k are those used in Algorithms *ON-LINE*($\mathcal{J}_0 = \emptyset$) and *ON-LINE*.

Table 3.6: *Percentage of solved instances with* 10% *of jobs from set* $\mathcal{J}_0$

n	L (%)	Number of decision points	Percentage of proved decisions	Off-line optimal (%)	On-line optimal (%)	Optimal without proof (%)	Max error of $\mathcal{C}_{max}$ (%)	Average error of $\mathcal{C}_{max}$ (%)
	1	18	72.22	79	18	2	17.966948	0.179669
	2	26	61.54	65	25	1	21.889175	1.026670
	3	49	69.39	49	37	8	12.544724	0.411497
	4	50	68.00	49	35	7	17.959299	0.656619
10	5	54	55.56	49	30	10	14.644439	0.911094
	6	73	64.38	26	49	14	14.280224	0.763463
	7	82	57.32	20	47	18	23.059456	1.461247
	8	92	59.78	21	46	17	16.038074	1.009285
	9	96	57.29	20	45	15	17.468323	1.431748
	10	114	56.14	10	49	19	19.623438	1.667662
	1	52	88.46	49	46	4	5.821766	0.058218
	2	99	92.93	18	75	3	6.804946	0.165536
	3	150	86.67	5	78	11	7.669424	0.257041
	4	154	85.71	5	77	6	8.306249	0.544108
20	5	190	85.26	6	71	12	15.721927	0.588305
	6	202	89.60	0	83	6	8.690571	0.590736
	7	233	86.70	1	75	13	9.809823	0.518078
	8	269	81.78	0	71	16	19.979408	0.823335
	9	247	79.76	0	64	10	10.719061	1.378448
	10	286	83.92	0	68	14	11.104570	1.122154
	1	133	93.98	14	80	5	3.743102	0.037431
	2	214	92.06	7	81	7	9.082943	0.171559
	3	275	93.82	2	84	11	6.714648	0.129161
	4	328	91.46	1	76	12	10.427673	0.408122
30	5	346	92.20	0	78	10	4.925068	0.523251
	6	340	88.53	0	68	15	13.092766	0.667315
	7	365	89.59	0	71	12	20.051733	0.891334
	8	424	90.80	0	71	10	13.626771	0.935853
	9	388	88.14	0	70	12	16.552177	0.836974
	10	416	88.70	0	68	19	9.795372	0.620623
	1	202	94.06	5	83	10	4.489825	0.085888
	2	354	94.35	0	85	13	6.027034	0.093683
40	3	428	95.33	0	87	8	4.454952	0.138229
	4	475	93.26	0	77	9	14.140268	0.689480
	5	513	96.69	0	88	6	17.602324	0.317764
	10	519	90.17	0	62	15	6.950419	0.735055
50	5	652	96.63	0	83	11	3.451158	0.161636
	10	580	89.83	0	58	12	6.377758	0.672041
60	5	739	95.81	0	75	12	3.175847	0.337660
	10	634	91.01	0	62	7	9.123387	0.838601
70	5	933	95.71	0	73	11	2.618088	0.303773
	10	729	93.14	0	66	13	8.430060	0.540960
80	5	992	94.76	0	72	8	9.327187	0.501414
	10	735	90.88	0	62	6	8.642805	0.703802
90	5	1113	96.14	0	76	10	11.927623	0.440951
	10	761	93.04	0	66	9	14.317971	0.661120
100	5	1083	95.57	0	68	7	4.340650	0.430289
	10	829	93.97	0	64	4	3.391186	0.517160

The percentage of problem instances which were optimally solved in the off-line scheduling phase is given in column 5. For such instances, Algorithm J-*SOL* terminates in step 8. The percentage of problem instances which were optimally solved in the on-line scheduling phase (and the optimality of the adopted permutation became only known after the schedule execution) is given in column 6. For such instances, Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$) (Algorithm *ON-LINE*) terminates in step 14. Note that both columns 5 and 6 define the percentage of problem instances for which optimal permutations were defined *before execution* of the whole schedule, i.e., each decision in the on-line phase (resolution of conflicting jobs) was made correctly due to one of the sufficient conditions proven in this section.

Column 7 presents the percentage of problem instances which were optimally solved occasionally (without a preliminary proof of permutation optimality). Namely, the value $\mathcal{C}_{max}$ obtained for the actual schedule turns out to be equal to the optimal value $\mathcal{C}^*_{max}$ calculated for an optimal schedule with the actual job processing times. (Note that the value $\mathcal{C}^*_{max}$ can be calculated after completing the last job from set J when all actual job processing times p^*_{ij}, $J_i \in J$, $M_j \in M$, and all actual job completion times become known). For such instances, Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$) (Algorithm *ON-LINE*) terminates in step 13. Subtracting the sum of the numbers given in columns 5, 6 and 7 from 100% gives the percentage of instances for which optimal permutations were not found both in the off-line scheduling phase and in the on-line scheduling phase. For such instances, Algorithm *ON-LINE* ($\mathcal{J}_0 = \emptyset$) and Algorithm *ON-LINE* terminate in step 12. The maximal (average) relative error of the makespan $[(\mathcal{C}_{max} - \mathcal{C}^*_{max})/\mathcal{C}^*_{max}] \cdot 100\%$ obtained for the actual schedule constructed by Algorithm J-*SOL* and Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$) (Algorithm *ON-LINE*) is given in column 8 (column 9).

Table 3.5 presents computational results for the case $\mathcal{J}_0 = \emptyset$ obtained by Algorithm J-*SOL* and Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$). Table 3.6 (Table 3.7) presents computational results for instances with 10% (30%) of the jobs from set $\mathcal{J}_0$ obtained by Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$) and Algorithm *ON-LINE*.

Table 3.8 (Table 3.9) presents the percentage of large instances ($200 \leq n \leq 1000$) solved exactly or approximately in the off-line phase by Algorithm J-*SOL* and in the on-line phase by Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$) for $\mathcal{J}_0 = \emptyset$ (by Algorithm *ON-LINE* for $\mathcal{J}_0 \neq \emptyset$). In Tables 3.8 and 3.9, we use the same columns as in Tables 3.5, 3.6 and 3.7 except column 5 since no large instance with $n > 100$ has been optimally solved in the off-line phase of scheduling.

The computational results seems to be rather promising, especially for the on-line scheduling phase. Tables 3.5, 3.6 and 3.7 show that off-line

Table 3.7: *Percentage of solved instances with* 30% *of jobs from set* $\mathcal{J}_0$

n	L (%)	Number of decision points	Percentage of proved decisions	Off-line optimal (%)	On-line optimal (%)	Optimal without proof (%)	Max error of $\mathcal{C}_{max}$ (%)	Average error of $\mathcal{C}_{max}$ (%)
	1	9	66.67	85	12	0	12.769878	0.191195
	2	14	78.57	70	27	0	13.634037	0.347974
	3	31	61.29	59	31	2	13.529054	0.735699
	4	40	32.50	44	37	5	23.480253	1.282487
10	5	43	37.21	42	35	6	21.291142	1.924775
	6	47	53.19	36	42	8	17.709812	1.613568
	7	62	48.39	27	44	7	18.857832	2.174505
	8	54	46.30	38	35	5	26.330475	2.890409
	9	71	42.25	24	40	9	26.084694	2.628773
	10	67	50.75	26	47	5	21.843046	2.253550
	1	26	80.77	54	41	2	9.495034	0.189950
	2	66	72.73	25	59	6	9.860765	0.496118
	3	92	78.26	19	65	7	8.274606	0.526225
	4	126	79.37	8	73	10	8.872502	0.651189
20	5	123	82.11	4	76	6	9.250347	0.918365
	6	145	83.45	1	77	5	8.582347	1.017275
	7	172	74.42	1	67	9	13.947658	1.272475
	8	169	78.11	0	72	8	22.539068	1.723347
	9	203	78.33	0	73	12	10.772242	1.085768
	10	190	81.05	0	73	5	11.410321	1.304585
	1	74	97.30	26	72	1	5.184722	0.051847
	2	132	92.42	6	84	4	5.993108	0.310904
	3	195	92.31	1	87	8	5.672687	0.189419
	4	223	89.24	0	78	10	7.023677	0.547064
30	5	260	91.54	0	81	3	7.348124	0.853035
	6	274	87.96	0	75	5	8.557904	0.987270
	7	281	92.53	0	83	8	13.237219	0.531717
	8	310	87.42	0	71	10	6.351249	0.769261
	9	320	81.88	0	67	10	14.983966	1.227586
	10	330	85.45	0	62	14	15.381570	1.045122
	1	133	93.23	7	86	2	4.371631	0.119839
	2	209	91.87	0	84	11	4.267471	0.200966
40	3	264	93.94	0	87	8	6.235053	0.216928
	4	324	91.36	0	81	8	4.509482	0.434682
	5	389	93.83	0	84	5	4.814591	0.373192
	10	413	83.29	0	51	7	12.213601	1.728035
50	5	490	93.06	0	78	9	3.951199	0.374512
	10	461	87.42	0	65	5	20.524624	1.057270
60	5	569	95.61	0	79	5	7.773736	0.510159
	10	554	91.34	0	66	6	10.940594	0.846808
70	5	715	95.66	0	77	4	3.125759	0.413271
	10	650	94.00	0	73	3	2.815311	0.485601
80	5	806	97.15	0	81	8	9.777588	0.338340
	10	654	91.44	0	61	5	5.004590	0.742792
90	5	821	96.22	0	79	4	4.352552	0.338179
	10	771	92.74	0	64	3	3.976014	0.647212
100	5	984	96.24	0	73	6	2.246622	0.350034
	10	714	92.86	0	63	7	3.588353	0.444596

scheduling allowed us to find optimal schedules only for small numbers of jobs and small errors of the input data, e.g., for $n = 40$ and $L = 1\%$ dominant permutations have been obtained only for 4% of the randomly generated instances. For $n > 40$, there were no such instances at all. Fortunately, on-line scheduling allowed us to find optimal schedules (with optimality proofs before the schedule execution) for most instances with $n \leq 100$ (Tables 3.5

Table 3.8: *Percentage of solved instances with an empty set* $\mathcal{J}_0$

n	L (%)	Number of decision points	Percentage of proved decisions	On-line optimal (%)	Optimal without proof (%)	Max error of $\mathcal{C}_{max}$ (%)	Average error of $\mathcal{C}_{max}$ (%)
200	5	1665	96.04	66	28	0.000819	0.000032
	10	909	88.67	42	45	1.687293	0.019191
	15	563	77.09	12	60	1.810943	0.054092
	20	424	64.62	0	66	5.189355	0.219553
300	5	1617	96.23	65	29	0.403977	0.004053
	10	891	85.52	25	59	13.129663	0.131382
	15	548	69.16	1	65	2.639358	0.096203
	20	419	56.09	0	59	5.012075	0.096354
400	5	1605	93.21	47	45	0.000692	0.000014
	10	840	80.48	10	62	3.807984	0.064929
	15	484	65.70	0	68	1.623536	0.025619
	20	361	48.20	0	55	2.821229	0.079312
500	5	1834	95.58	57	31	0.000462	0.000024
	10	840	79.29	4	73	11.960889	0.123434
	15	468	63.03	0	58	5.103044	0.069323
	20	309	33.98	0	44	1.205535	0.025511
600	5	1659	93.31	45	45	2.989783	0.054760
	10	783	77.01	1	60	1.114273	0.026779
	15	417	57.31	0	55	0.212790	0.003442
	20	273	28.21	0	54	0.607854	0.011057
700	5	1766	94.11	48	35	1.544116	0.024961
	10	706	77.48	1	61	1.575914	0.028462
	15	392	51.79	0	55	1.496273	0.027993
	20	244	24.59	0	30	1.170049	0.020081
800	5	1665	92.19	45	39	1.034108	0.011902
	10	691	75.69	0	59	3.810499	0.050601
	15	333	40.24	0	42	2.311629	0.088997
	20	209	20.10	0	36	0.263040	0.002976
900	5	1599	90.43	35	46	6.364723	0.075417
	10	628	67.83	0	54	4.828169	0.068645
	15	323	42.41	0	40	0.438542	0.005272
	20	193	11.92	0	27	2.273839	0.067102
1000	5	1621	92.23	32	49	0.000402	0.000020
	10	593	66.78	1	57	0.000558	0.000062
	15	297	33.33	0	42	3.157275	0.031717
	20	171	14.04	0	27	0.889870	0.023861

– 3.7) and for many instances with $200 \leq n \leq 1000$ (Tables 3.8 and 3.9).

The following computational results are even more impressive. The average relative error of the makespan $[(\mathcal{C}_{max} - \mathcal{C}^*_{max})/\mathcal{C}^*_{max}] \cdot 100\%$ obtained for all actual schedules is less than 2.9% for all randomly generated instances with $n = 10$ jobs (column 9 in Tables 3.5 – 3.7). The average relative error of the makespan obtained for all actual schedules is less than 1.67% for all randomly generated instances with n jobs with $20 \leq n \leq 1000$ (column 9 in Tables 3.5 – 3.7, column 8 in Tables 3.8 and 3.9). These results are obtained since the percentage of the correct decisions made in the on-line scheduling phase is rather large (column 4). Thus, the sufficient conditions for the existence of a dominant permutation given in Theorems 3.8 – 3.20 may be very effective for on-line scheduling.

It should be also noted that the number of decision-making time-points

Table 3.9: *Percentage of solved instances with* 30% *of jobs from set* $\mathcal{J}_0$

n	L (%)	Number of decision points	Percentage of proved decisions	On-line optimal (%)	Optimal without proof (%)	Max error of $\mathcal{C}_{max}$ (%)	Average error of $\mathcal{C}_{max}$ (%)
200	5	1307	95.87	68	7	1.493803	0.228527
	10	836	92.94	59	10	4.797616	0.268801
	15	592	81.25	23	14	3.531275	0.550724
	20	424	69.58	1	32	8.917567	0.603160
300	5	1636	96.45	67	1	6.174240	0.230672
	10	880	89.20	40	11	2.548966	0.272686
	15	578	77.34	5	27	0.684393	0.327328
	20	448	66.74	0	23	2.098592	0.402790
400	5	1612	96.09	60	4	4.655282	0.200383
	10	873	87.06	28	13	2.560096	0.274370
	15	550	76.18	3	28	3.529257	0.306747
	20	411	61.07	0	21	4.233299	0.330161
500	5	1742	95.24	58	2	1.803466	0.155213
	10	797	84.32	17	27	5.349672	0.271104
	15	526	70.53	0	26	4.218193	0.215071
	20	384	63.28	0	24	3.550418	0.235060
600	5	1710	95.50	60	5	1.594194	0.115055
	10	861	85.25	13	20	1.047556	0.172209
	15	479	68.68	0	32	1.667378	0.192229
	20	335	55.22	0	23	2.664976	0.199377
700	5	1787	96.59	62	3	0.283844	0.085372
	10	821	82.58	8	25	0.974754	0.136061
	15	458	68.12	0	21	3.935263	0.207185
	20	294	46.26	0	30	1.589900	0.154975
800	5	1789	95.03	52	8	0.251936	0.081683
	10	832	83.17	5	24	0.247568	0.125456
	15	443	64.79	0	22	0.766673	0.121888
	20	293	49.49	0	15	1.834536	0.151659
900	5	1737	95.28	57	7	0.934157	0.074773
	10	744	77.42	2	30	0.522401	0.117334
	15	397	59.19	0	28	1.861207	0.141729
	20	263	49.43	0	20	3.329657	0.142571
1000	5	1724	95.01	53	7	1.111841	0.072543
	10	728	80.91	2	28	0.266260	0.098637
	15	404	61.39	0	27	2.376391	0.107110
	20	229	43.23	0	21	1.266256	0.124885

$t_i = c_1(i)$ when the order of conflicting jobs has to be decided is rather large for some instances with $n \geq 50$ (column 3). However, these decisions made in Algorithm *ON-LINE*($\mathcal{J}_0 = \emptyset$) (Algorithm *ON-LINE*) are very fast: There were no randomly generated instance which takes a running time more than 0.05 seconds for a processor with 1200 MHz.

3.3. Job Shop with Interval Processing Times

In this section, we address the two-machine job shop problem for the case when it is hard to obtain exact probability distributions for the random processing times, and when assuming a specific probability distribution is not realistic. Usually, the schedules obtained after assuming a certain probability distribution may be not close to an optimal schedule. It has been

observed that, although the exact probability distribution of the job processing times may not be known in advance (before scheduling), upper and lower bounds for the job processing times are easy to obtain in many practical cases. We show that the information on the bounds of the job processing times is important and should be utilized in finding a solution to the scheduling problem with interval processing times. In Section 3.1, necessary and sufficient conditions are given when a transposition of two jobs may be used to minimize the makespan for the uncertain flow shop problem with two machines. In this section, we show how to use this result for solving the uncertain two-machine job shop problem. We consider the following nonpreemptive job shop problem with interval processing times. Two machines $M = \{M_1, M_2\}$ have to process n jobs J with different two-stage routes. Machine repetition in a route is not allowed. Let $J(12) \subseteq J$ denote the subset of jobs with the route (M_1, M_2), i.e., job $J_i \in J(12)$ has to be processed on machine M_1 and then on machine M_2. Let $J(21) \subseteq J$ denote the subset of jobs with the opposite route (M_2, M_1), and $J(k) \subseteq J$ denote the subset of jobs that have to be processed only on machine $M_k, k \in \{1, 2\}$. Thus, we have $J = J(1) \cup J(2) \cup J(12) \cup J(21)$. We denote $q_k = |J(k)|$, where $k \in \{1, 2, 12, 21\}$. In contrast to the deterministic job shop problem, it is assumed that the processing time p_{ij} of job J_i on machine M_j is not fixed before scheduling. In the realization of the process, p_{ij} may take any real value between a lower bound p_{ij}^L and an upper bound p_{ij}^U being given before scheduling. The probability distribution of the random processing time is unknown. In such a case, there may not exist a unique schedule that remains optimal for all possible realizations of the job processing times. Therefore, we consider a set of schedules that dominate all feasible schedules. We consider the criterion $\mathcal{C}_{\max}$, i.e., the minimization of the schedule length.

Let $T^J \subseteq R_+^q$ define the given set of feasible vectors of the job processing times. For the case of an uncertain job shop problem, it is useful to consider T^J as the Cartesian product of four sets: $T^J = T_{12} \times T_1 \times T_2 \times T_{21}$, where T_w is the set of feasible vectors defining the processing times of the jobs from set $J(k)$, $k \in \{1, 2, 12, 21\}$. In what follows, inclusion $p \in T^J$ means that vector p has the dimension $q = q_{12} + q_1 + q_2 + q_{21}$ and the set of its components consists of the following four ordered subsets $\{p_{ij} : J_i \in J(12), j \in \{1, 2\}\}$, $\{p_{ij} : J_i \in J(1), j = 1\}$, $\{p_{ij} : J_i \in J(2), j = 2\}$ and $\{p_{ij} : J_i \in J(21), j \in \{1, 2\}\}$. We also assume that the jobs from set J are numbered with respect to the above order of these four subsets of the components of vector p. Thus, vector $p \in T^J$ may be represented as follows: $p = (p_{1,1}, p_{1,2}, \ldots, p_{q_{12},1}, p_{q_{12},2}; p_{q_{12}+1,1}, p_{q_{12}+2,1}, \ldots, p_{q_{12}+q_1,1}; p_{q_{12}+q_1+1,2}, p_{q_{12}+q_1+2,2}, \ldots, p_{q_{12}+q_1+q_2,2};$

$p_{q_{12}+q_1+q_2+1,1}, p_{q_{12}+q_1+q_2+1,2}, \ldots, p_{q,1}, p_{q,2})$ or in a more short way: $p = (p(12); p(1); p(2); p(21))$, where $p(12) \in T_{12}$, $p(1) \in T_1$, $p(2) \in T_2$ and $p(21) \in T_{21}$. Using the three-field notation, this uncertain job shop problem is denoted as $\mathcal{J}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U, n_i \leq 2/\mathcal{C}_{\max}$. If equality $p_{ij}^L = p_{ij}^U$ holds for each job $J_i \in J$ and machine $M_j \in M$, then problem $\mathcal{J}2/p_{ij}^L \leq p_{ij} \leq p_{ij}^U, n_i \leq 2/\mathcal{C}_{\max}$ turns into a deterministic job shop problem $\mathcal{J}2/n_i \leq 2/\mathcal{C}_{\max}$ that is polynomially solvable due to the following theorem proven by Jackson [168].

Theorem 3.21 *For problem $\mathcal{J}2/n_i \leq 2/\mathcal{C}_{\max}$, a semiactive schedule for processing the jobs $J = J(1) \cup J(2) \cup J(12) \cup J(21)$ has the minimal length, if machine M_1 processes the jobs in the order $(J(12), J(1), J(21))$, machine M_2 in the order $(J(21), J(2), J(12))$ provided that the jobs $J(12)$ are ordered for processing on both machines using the condition*

$$\min\{p_{i_k1}, p_{i_{k+1}2}\} \leq \min\{p_{i_{k+1}1}, p_{i_k2}\},\ k = 1, 2, \ldots, q_{12} - 1, \tag{3.39}$$

and the jobs $J(21)$ are ordered for processing on both machines using the condition

$$\min\{p_{i_k2}, p_{i_{k+1}1}\} \leq \min\{p_{i_{k+1}2}, p_{i_k1}\},\ k = 1, 2, \ldots, q_{21} - 1, \tag{3.40}$$

jobs $J(1)$ and jobs $J(2)$ may be arbitrarily ordered for processing.

It is easy to see that, due to Theorem 3.21, the jobs $J(12)$ can be optimally ordered via Johnson's algorithm similarly as in a flow shop problem $\mathcal{F}2//\mathcal{C}_{\max}$ since condition (3.39) coincides with condition (3.2) (see Section 3.1, page 181) provided that $J = J(12)$ and $T = T_{12}$. The jobs $J(21)$ can be optimally ordered via Johnson's algorithm in a flow shop problem $\mathcal{F}2//\mathcal{C}_{\max}$ with the opposite route (M_2, M_1) of the jobs since condition (3.40) may be obtained from condition (3.2) after interchanging machine M_1 with machine M_2 and setting $J = J(21)$ and $T = T_{21}$. Thus, an optimal schedule for problem $\mathcal{J}2/n_i \leq 2/\mathcal{C}_{\max}$ may be defined by the pair of permutations π' and π'' (we call it Jackson pair of permutations), where $\pi' = (\pi_i^{(12)}, \pi_u^{(1)}, \pi_j^{(21)})$ is a permutation of the jobs on machine M_1, and $\pi'' = (\pi_j^{(21)}, \pi_v^{(2)}, \pi_i^{(12)})$ is a permutation of the jobs on machine M_1. Job J_r belongs to permutation $\pi_k^{(w)}$ if and only if $J_r \in J(w)$ with $w \in \{1, 2, 12, 21\}$. To be more precise, permutation π' of the jobs $J(12) \cup J(1) \cup J(21)$ on machine M_1, and permutation π'' of the jobs $J(21) \cup J(2) \cup J(12)$ on machine M_2 may be defined as follows.

The jobs within permutation $\pi_i^{(12)}$ (permutation $\pi_j^{(21)}$, respectively) have to be ordered by the SPT (LPT) rule on machine M_1 (on machine M_2). Rule

SPT (shortest processing time) means that the jobs have to be sorted in a non-decreasing order of the processing times. Rule LPT (longest processing time) means that the jobs have to be sorted in a non-increasing order of the processing times. Since the order of the jobs in permutation $\pi_u^{(1)}$ and in permutation $\pi_v^{(2)}$ may be arbitrary, we can fix these permutations. Let the jobs in permutations $\pi_u^{(1)}$ and $\pi_v^{(2)}$ be sorted with respect to the job numbers:

$$\pi_u^{(1)} = (J_{q_{12}+1}, J_{q_{12}+2}, \ldots, J_{q_{12}+q_1}),\ \pi_v^{(2)} = (J_{q_{12}+q_1+1}, J_{q_{12}+q_1+2}, \ldots, J_{q_{12}+q_1+q_2}).$$

We denote the set of all permutations of the jobs $J(12)$ as

$$S_{12}^{\pi} = \{\pi_1^{(12)}, \pi_2^{(12)}, \ldots, \pi_{q_{12}!}^{(12)}\},$$

and the set of all permutations of the jobs $J(21)$ as

$$S_{21}^{\pi} = \{\pi_1^{(21)}, \pi_2^{(21)}, \ldots, \pi_{q_{21}!}^{(21)}\}.$$

Let $\langle S_{12}^{\pi}, S_{21}^{\pi}\rangle$ be a subset of the Cartesian product $(S_{12}^{\pi}, \pi_u^{(1)}, S_{21}^{\pi}) \times (S_{21}^{\pi}, \pi_v^{(2)}, S_{12}^{\pi})$ such that the elements of set $\langle S_{12}^{\pi}, S_{21}^{\pi}\rangle$ are ordered pairs of the two permutations π' and π'', where $\pi' = (\pi_i^{(12)}, \pi_u^{(1)}, \pi_j^{(21)})$ and $\pi'' = (\pi_j^{(21)}, \pi_v^{(2)}, \pi_i^{(12)})$, $1 \le i \le q_{12}!$, $1 \le j \le q_{21}!$. Since both permutations $\pi_u^{(1)}$ and $\pi_v^{(2)}$ are fixed, and index i (index j) is the same in each permutation from the pair π', π'', we obtain $|\langle S_{12}^{\pi}, S_{21}^{\pi}\rangle| = q_{12}!q_{21}!$ pairs of permutations. We use the following definition of a J-solution to problem $\mathcal{J}2/p_{ij}^L \le p_{ij} \le p_{ij}^U, n_i \le 2/\mathcal{C}_{\max}$.

Definition 3.3 *The set of pairs of permutations* $\langle S_{12}^{\pi}(T_{12}), S_{21}^{\pi}(T_{21})\rangle \subseteq \langle S_{12}^{\pi}, S_{21}^{\pi}\rangle$ *is called a J-solution to problem* $\mathcal{J}2/p_{ij}^L \le p_{ij} \le p_{ij}^U, n_i \le 2/\mathcal{C}_{\max}$, *if for each vector* $p \in T^J$, *the set* $\langle S_{12}^{\pi}(T_{12}), S_{21}^{\pi}(T_{21})\rangle$ *contains at least one optimal Jackson pair of permutations for problem* $\mathcal{J}2/n_i \le 2/\mathcal{C}_{\max}$ *with the vector* p *of job processing times. If any proper subset of the J-solution* $\langle S_{12}^{\pi}(T_{12}), S_{21}^{\pi}(T_{21})\rangle$ *is not a J-solution to problem* $\mathcal{J}2/p_{ij}^L \le p_{ij} \le p_{ij}^U, n_i \le 2/\mathcal{C}_{\max}$, *then* $\langle S_{12}^{\pi}(T_{12}), S_{21}^{\pi}(T_{21})\rangle$ *is called a minimal J-solution.*

Next, we show that the results for the flow shop problem obtained in Section 3.1 may be used for solving the job shop problem with interval processing times. In particular, we prove the following lemma.

Lemma 3.4 *Let* $S_{12}^{\pi}(T_{12})$ *be a J-solution to problem* $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{\max}$ *with the set of jobs* $J = J(12)$ *and the processing times defined by the set of vectors* $T = T_{12}$. *Then* $\langle S_{12}^{\pi}(T_{12}), S_{21}^{\pi}\rangle$ *is a J-solution to problem* $\mathcal{J}2/p_{ij}^L \le p_{ij} \le p_{ij}^U, n_i \le 2/\mathcal{C}_{\max}$ *with the set of jobs* $J = J(1) \cup J(2) \cup J(12) \cup J(21)$ *and the processing times defined by the set of vectors* $T^J = T_{12} \times T_1 \times T_2 \times T_{21}$.

PROOF. By contradiction, we assume that there exists a vector $p = (p(12); p(1); p(2); p(21)) \in T^J$ of job processing times, for which no optimal Jackson pair of permutations for problem $\mathcal{J}2/n_i \leq 2/\mathcal{C}_{\max}$ with the vector p of job processing times belongs to set $\langle S^{\pi}_{12}(T_{12}), S^{\pi}_{21}\rangle$.

Due to Definition 3.3 and taking into account that set S^{π}_{21} contains all permutations of jobs $J(21)$, a permutation pair from set $\langle S^{\pi}_{12}(T_{12}), S^{\pi}_{21}\rangle$ may be not a J-solution to problem $\mathcal{J}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}, n_i \leq 2/\mathcal{C}_{\max}$ with the set of jobs $J = J(1) \cup J(2) \cup J(12) \cup J(21)$ and the processing times defined by the set of vectors $T^J = T_{12} \times T_1 \times T_2 \times T_{21}$ only if set $S^{\pi}_{12}(T_{12})$ does not include a Johnson permutation to problem $\mathcal{F}2//\mathcal{C}_{\max}$ with the job set $J = J(12)$ and the vector $p(12) \in T = T_{12}$ of job processing times. Thus, due to Definition 3.1, set $S^{\pi}_{12}(T_{12})$ is not a J-solution to problem $\mathcal{F}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}/\mathcal{C}_{\max}$ with the set of jobs $J = J(12)$ and the set $T = T_{12}$ of feasible vectors of the job processing times.

We obtain a contradiction to the condition of Lemma 3.4. Hence, our assumption was wrong. In fact, for each vector $p \in T^J$, set $\langle S^{\pi}_{12}(T_{12}), S^{\pi}_{21}\rangle$ contains an optimal Jackson pair of permutations for problem $\mathcal{J}2/n_i \leq 2/\mathcal{C}_{\max}$ with the vector p of job processing times. Consequently, set $\langle S^{\pi}_{12}(T_{12}), S^{\pi}_{21}\rangle$ is a J-solution to problem $\mathcal{J}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}, n_i \leq 2/\mathcal{C}_{\max}$ with the set of jobs $J = J(1) \cup J(2) \cup J(12) \cup J(21)$ and the processing times defined by the set of vectors $T^J = T_{12} \times T_1 \times T_2 \times T_{21}$.

$\diamond$

Similarly we can prove the following claim.

Lemma 3.5 *Let $S^{\pi}_{21}(T_{21})$ be a J-solution to problem $\mathcal{F}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}/\mathcal{C}_{\max}$ with the set of jobs $J = J(21)$ and the processing times defined by the set of vectors $T = T_{21}$. Then $\langle S^{\pi}_{12}, S^{\pi}_{21}(T_{21})\rangle$ is a J-solution to problem $\mathcal{J}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}, n_i \leq 2/\mathcal{C}_{\max}$ with the set of jobs $J = J(1) \cup J(2) \cup J(12) \cup J(21)$ and the processing times defined by the set of vectors $T^J = T_{12} \times T_1 \times T_2 \times T_{21}$.*

Using Lemma 3.4 and Lemma 3.5, we prove the following claim.

Theorem 3.22 *If $S^{\pi}_{12}(T_{12})$ is a J-solution to the flow shop problem $\mathcal{F}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}/\mathcal{C}_{\max}$ with the set of jobs $J(12)$ and the processing times defined by the set of vectors $T = T_{12}$, and $S^{\pi}_{21}(T_{21})$ is a J-solution to the flow shop problem $\mathcal{F}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}/\mathcal{C}_{\max}$ with the set of jobs $J(21)$ and the processing times defined by the set of vectors $T = T_{21}$, then $\langle S^{\pi}_{12}(T_{12}), S^{\pi}_{21}(T_{21})\rangle$ is a J-solution to problem $\mathcal{J}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}, n_i \leq 2/\mathcal{C}_{\max}$ with the set of jobs $J = J(1) \cup J(2) \cup J(12) \cup J(21)$ and the processing times defined by the set of vectors $T^J = T_{12} \times T_1 \times T_2 \times T_{21}$.*

PROOF. By contradiction, we assume that there exists a vector $p = (p(12); p(1); p(2); p(21)) \in T^J$ of job processing times, for which no optimal Jackson pair of permutations for problem $\mathcal{J}2/n_i \leq 2/\mathcal{C}_{\max}$ with the vector p of processing times belongs to set $\langle S^{\pi}_{12}(T_{12}), S^{\pi}_{21}(T_{21})\rangle$.

Due to Definition 3.3, set $\langle S^{\pi}_{12}(T_{12}), S^{\pi}_{21}(T_{21})\rangle$ is not a J-solution for problem $\mathcal{J}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}, n_i \leq 2/\mathcal{C}_{\max}$ with the set of jobs $J = J(1) \cup J(2) \cup J(12) \cup J(21)$ and the processing times defined by the set of vectors $T^J = T_{12} \times T_1 \times T_2 \times T_{21}$ either if set $S^{\pi}_{12}(T_{12})$ does not include a Johnson permutation for problem $\mathcal{F}2//\mathcal{C}_{\max}$ with job set $J(12)$, route (M_1, M_2) and processing times $p(12)$) or if set $S^{\pi}_{21}(T_{21})$ does not include a Johnson permutation for problem $\mathcal{F}2//\mathcal{C}_{\max}$ with job set $J(21)$, route (M_2, M_1) and processing times $p(21)$). We denote the former case as case (a), and the latter case as case (b). In case (a), set $\langle S^{\pi}_{12}(T), S^{\pi}_{21}\rangle$ cannot be a J-solution to problem $\mathcal{F}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}/\mathcal{C}_{\max}$ with the set of jobs $J = J(12)$ and the set $T = T_{12}$ of feasible vectors of job processing times. However, this is not possible due to Lemma 3.4 (since the condition of Lemma 3.4 holds). Thus, we obtain a contradiction. Hence, in case (a), our assumption was wrong.

Using Lemma 3.5 and arguing similarly, we can obtain a contradiction in case (b) as well.

$\diamond$

Due to Theorem 3.22, solving problem $\mathcal{J}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}, n_i \leq 2/\mathcal{C}_{\max}$ is reduced to solving two problems $\mathcal{F}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}/\mathcal{C}_{\max}$. To illustrate this, we consider the following two examples of an uncertain job shop problem.

Example 3.5 *Let the bounds for the job processing times of problem $\mathcal{J}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}, n_i \leq 2/\mathcal{C}_{\max}$ be given in Table 3.10. These bounds define the set T^J of feasible vectors of the job processing times. The set of jobs J consists of four subsets $J(12) = \{J_1, J_2, J_3\}$, $J(1) = \{J_4\}$, $J(2) = \{J_5\}$ and $J(21) = \{J_6, J_7, J_8\}$. It is easy to see that Theorem 3.21 used for problem*

Table 3.10: *Lower and upper bounds for the job processing times in Example 3.5*

i	1	2	3	4	5	6	7	8
a_{i1}	2	3	7	2	-	4	5	7
b_{i1}	4	4	8	3	-	5	6	8
a_{i2}	5	5	7	-	3	1	2	7
b_{i2}	6	6	8	-	5	3	3	8

$\mathcal{F}2/p^L_{ij} \leq p_{ij} \leq p^U_{ij}/\mathcal{C}_{\max}$ with the job set $J(12)$ implies the following binary relation $\mathcal{A}_{\preceq}$ (see page 195 in Section 3.1) on the job set $J(12) = \{J_1, J_2, J_3\}$:

$J_1 \preceq J_3$ and $J_2 \preceq J_3$. Therefore, instead of considering $3! = 6$ permutations of the jobs $J(12)$, it is sufficient to consider two permutations: (J_1, J_2, J_3) and (J_2, J_1, J_3).

Next, we consider a flow shop problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{\max}$ with the job set $J(21) = \{J_6, J_7, J_8\}$ and the route (M_2, M_1). Due to Theorem 3.21, we find the binary relation $\mathcal{A}_{\preceq}$ on the job set $J(21)$: $J_6 \preceq J_8$ and $J_7 \preceq J_8$. Therefore, instead of considering $3! = 6$ permutations of these three jobs, it is sufficient to consider two permutations: (J_6, J_7, J_8) and (J_7, J_6, J_8).

Due to Theorem 3.22, for the job shop problem $\mathcal{J}2/p_{ij}^L \le p_{ij} \le p_{ij}^U, n_i \le 2/\mathcal{C}_{\max}$ with the job set $J = J(12) \cup J(1) \cup J(2) \cup J(21) = \{J_1, J_2, \ldots, J_8\}$ and the intervals of the processing times given in Table 3.10, we obtain a J-solution which consists of the following four pairs of permutations:

$$\pi_1' = (J_1, J_2, J_3, J_4, J_6, J_7, J_8), \pi_1'' = (J_6, J_7, J_8, J_5, J_1, J_2, J_3);$$

$$\pi_2' = (J_1, J_2, J_3, J_4, J_7, J_6, J_8), \pi_2'' = (J_7, J_6, J_8, J_5, J_1, J_2, J_3);$$

$$\pi_3' = (J_2, J_1, J_3, J_4, J_6, J_7, J_8), \pi_3'' = (J_6, J_7, J_8, J_5, J_2, J_1, J_3);$$

$$\pi_4' = (J_2, J_1, J_3, J_4, J_7, J_6, J_8), \pi_4'' = (J_7, J_6, J_8, J_5, J_2, J_1, J_3).$$

Thus, instead of considering $q_{12}! \cdot q_{21}! = 3! \cdot 3! = 36$ permutations, it is sufficient to consider four pairs of permutations which certainly includes an optimal Jackson pair of permutations for each vector $p \in T^J$ of the job processing times.

Example 3.6 *Let us define a minimal J-solution to problem $\mathcal{J}2/p_{ij}^L \le p_{ij} \le p_{ij}^U, n_i \le 2/\mathcal{C}_{\max}$ for processing the set of jobs $J = J(12) \cup J(1) \cup J(2) \cup J(21) = \{J_1, J_2, \ldots, J_{14}\}$, where the jobs $J_1, J_2, \ldots, J_5$ from set $J(12)$ have the route (M_1, M_2). Job $J_6 \in J(1)$ and job $J_7 \in J(1)$ have the route (M_1). Jobs $J_8 \in J(2)$ and $J_9 \in J(2)$ have the route (M_2). Jobs $J_{10}, J_{11}, \ldots, J_{14}$ from set $J(21)$ have the route (M_2, M_1). The intervals of feasible job processing times are given in Table 3.11. First, we consider the uncertain flow shop*

Table 3.11: *Lower and upper bounds for the job processing times in Example 3.6*

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a_{i1}	5	2	8	7	10	4	2	-	-	2	7	8	10	2
b_{i1}	7	4	11	10	13	5	3	-	-	3	8	9	11	4
a_{i2}	8	6	7	7	5	-	-	4	5	5	3	4	9	6
b_{i2}	10	9	10	9	7	-	-	6	7	7	5	5	12	6

problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{\max}$ with the job set $J(12) = \{J_1, J_2, J_3, J_4, J_5\}$

and the route (M_1, M_2). *Due to Theorem 3.21, we find the following binary relation* $\mathcal{A}_{\preceq}$ *on the job set* $J(12)$*:* $J_2 \preceq J_1$, $J_1 \preceq J_3$, $J_1 \preceq J_4$, $J_3 \preceq J_5$, $J_4 \preceq J_5$. *Therefore, instead of considering* $5! = 120$ *permutations of these five jobs, it is sufficient to consider two permutations:* $(J_2, J_1, J_3, J_4, J_5)$ *and* $(J_2, J_1, J_4, J_3, J_5)$.

Next, we consider the uncertain flow shop problem $\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\mathcal{C}_{\max}$ *with the job set* $J(21) = \{J_{10}, J_{11}, J_{12}, J_{13}, J_{14}\}$ *and the route* (M_2, M_1). *Due to Theorem 3.21, we find the following binary relation* $\mathcal{A}_{\preceq}$ *on the job set* $J(21)$*:* $J_{11} \preceq J_{13}$, $J_{12} \preceq J_{13}$, $J_{13} \preceq J_{10}$, $J_{13} \preceq J_{14}$. *Therefore, instead of considering* $5! = 120$ *permutations of these five jobs, it is sufficient to consider four permutations:* $(J_{11}, J_{12}, J_{13}, J_{10}, J_{14})$, $(J_{11}, J_{12}, J_{13}, J_{14}, J_{10})$, $(J_{12}, J_{11}, J_{13}, J_{10}, J_{14})$ *and* $(J_{12}, J_{11}, J_{13}, J_{14}, J_{10})$.

Using Theorem 3.22, we obtain that a minimal J-solution to problem $\mathcal{J}2/p_{ij}^L \le p_{ij} \le p_{ij}^U, n_i \le 2/\mathcal{C}_{\max}$ *with the set of jobs* $J = \{J_1, J_2, \ldots, J_{14}\}$ *includes the following pairs of permutations (the sequences for processing the jobs of set* J_1 *and the jobs of set* J_2 *are fixed with respect to the job numbers):*

$$\pi_1' = (J_2, J_1, J_3, J_4, J_5, J_6, J_7, J_{11}, J_{12}, J_{13}, J_{10}, J_{14}),$$
$$\pi_1'' = (J_{11}, J_{12}, J_{13}, J_{10}, J_{14}, J_8, J_9, J_2, J_1, J_3, J_4, J_5);$$
$$\pi_2' = (J_2, J_1, J_3, J_4, J_5, J_6, J_7, J_{11}, J_{12}, J_{13}, J_{14}, J_{10}),$$
$$\pi_2'' = (J_{11}, J_{12}, J_{13}, J_{14}, J_{10}, J_8, J_9, J_2, J_1, J_3, J_4, J_5);$$
$$\pi_3' = (J_2, J_1, J_3, J_4, J_5, J_6, J_7, J_{12}, J_{11}, J_{13}, J_{10}, J_{14}),$$
$$\pi_3'' = (J_{12}, J_{11}, J_{13}, J_{10}, J_{14}, J_8, J_9, J_2, J_1, J_3, J_4, J_5);$$
$$\pi_4' = (J_2, J_1, J_3, J_4, J_5, J_6, J_7, J_{12}, J_{11}, J_{13}, J_{14}, J_{10}),$$
$$\pi_4'' = (J_{12}, J_{11}, J_{13}, J_{14}, J_{10}, J_8, J_9, J_2, J_1, J_3, J_4, J_5);$$
$$\pi_5' = (J_2, J_1, J_4, J_3, J_5, J_6, J_7, J_{11}, J_{12}, J_{13}, J_{10}, J_{14}),$$
$$\pi_5'' = (J_{11}, J_{12}, J_{13}, J_{10}, J_{14}, J_8, J_9, J_2, J_1, J_4, J_3, J_5);$$
$$\pi_6' = (J_2, J_1, J_4, J_3, J_5, J_6, J_7, J_{11}, J_{12}, J_{13}, J_{14}, J_{10}),$$
$$\pi_6'' = (J_{11}, J_{12}, J_{13}, J_{14}, J_{10}, J_8, J_9, J_2, J_1, J_4, J_3, J_5);$$
$$\pi_7' = (J_2, J_1, J_4, J_3, J_5, J_6, J_7, J_{12}, J_{11}, J_{13}, J_{10}, J_{14}),$$
$$\pi_7'' = (J_{12}, J_{11}, J_{13}, J_{10}, J_{14}, J_8, J_9, J_2, J_1, J_4, J_3, J_5);$$
$$\pi_8' = (J_2, J_1, J_4, J_3, J_5, J_6, J_7, J_{12}, J_{11}, J_{13}, J_{14}, J_{10}),$$
$$\pi_8'' = (J_{12}, J_{11}, J_{13}, J_{14}, J_{10}, J_8, J_9, J_2, J_1, J_4, J_3, J_5).$$

A digraph (without transitive arcs), defining the above minimal J-solution to problem $\mathcal{J}2/p_{ij}^L \le p_{ij} \le p_{ij}^U, n_i \le 2/\mathcal{C}_{\max}$ *is presented in Figure 3.3.*

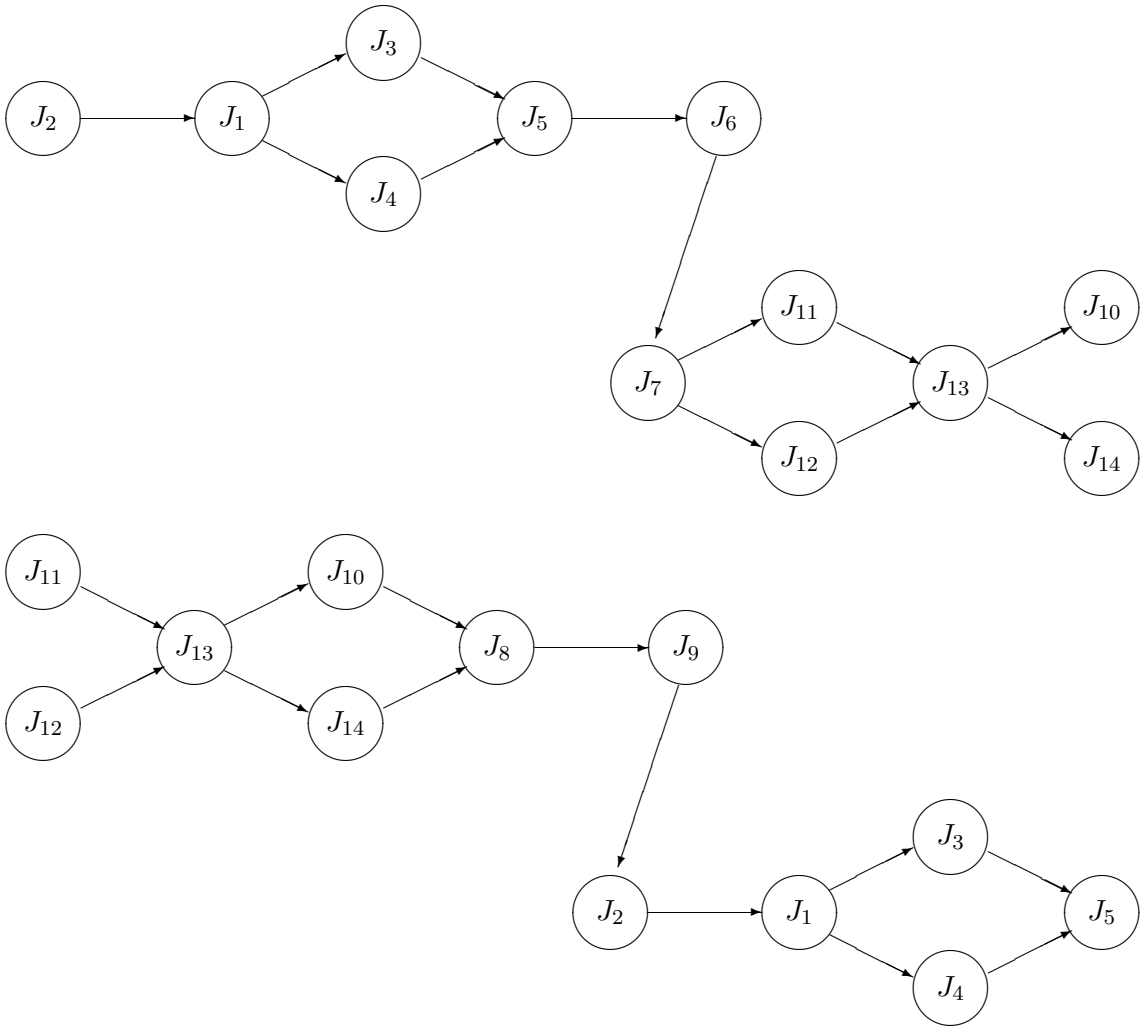


Figure 3.3: *Digraph representing the binary relations $\mathcal{A}_{\preceq}$ for Example 3.6*

3.4. Flow Shop with Limited Machine Availability

This section deals with the problem of minimizing the makespan in the two-machine flow shop with w non-availability intervals given on machines M_1 and M_2. First, we show that this problem is binary NP-hard even if there is only one non-availability interval ($w = 1$) either on machine M_1 or on machine M_2. If there are no non-availability intervals on any machine ($w = 0$), the two-machine flow shop problem with the makespan criterion may be solved in $O(n \log_2 n)$ time using Johnson's algorithm given on page 181. We derive sufficient conditions for the optimality of a Johnson permutation in the case of the $w \geq 1$ given non-availability intervals. The instrument we use is a stability analysis which answers the question how stable an optimal schedule is if there are independent changes in the processing times of the jobs. The influence of possible variations of the processing times on the

optimality of a schedule was investigated in Chapter 1, where the stability radius of an optimal schedule was studied for a general shop with $m \geq 2$ machines. In contrast to the stability analysis developed in Chapter 1, the variation of the processing time studied in this section may have only one direction, since the non-availability intervals of a machine may increase the interval used for processing a job but they cannot decrease this interval. Therefore, we have to modify the definition of the stability radius for the purposes of this section.

We assume that all w non-availability intervals are known offline (before scheduling). To indicate this, we shall use the notation NC^{off}. Any job $J_i \in J = \{J_1, J_2, \ldots, J_n\}$ that cannot be completed before the non-availability interval of machine $M_k \in M = \{M_1, M_2\}$ processes this job can be continued just after machine M_k will be available again. To indicate this, we use the notation *pmtn* since the latter assumption is close to the allowance of operation preemptions. Thus, in this and the next sections, it is allowed to violate Condition 4 (see Introduction, page 12). Using the three-field notation, the problem under consideration is denoted as $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$.

In this section, we use the following notations, where $j \in \{1, 2\}$ denotes the machine index and $i \in \{1, 2, \ldots, n\}$ denotes the job index: w_j denotes the number of non-availability intervals on machine M_j; N_{kj} denotes the k^{th} non-availability interval on machine M_j; $s(N_{kj})$ denotes the starting point of the k^{th} non-availability interval; $f(N_{kj})$ denotes the endpoint of the k^{th} non-availability interval; $h(N_{kj})$ denotes length of the k^{th} non-availability interval provided that $h(N_{kj}) = f(N_{kj}) - s(N_{kj})$.

It is assumed that w non-availability intervals are known before scheduling. Machine $M_j \in \{M_1, M_2\}$ is not available for processing jobs of set J from the starting point $s(N_{kj})$ until the endpoint $f(N_{kj}) = s(N_{kj}) + h(N_{kj})$, $k \in \{1, 2, \ldots, w_j\}$. The operation Q_{ij} started before point $s(N_{kj})$ but not finished until point $s(N_{kj})$ is suspended during the time interval of length $h(N_{kj})$ starting from point $s(N_{kj})$, then the processing of operation Q_{ij} is resumed from time $f(N_{kj}) = s(N_{kj}) + h(N_{kj})$. Let s_j (c_j, respectively) denote the earliest (latest) possible starting time of a job on machine M_j in a semiactive schedule. Let $s_{ij}(\pi_k)$ ($c_{ij}(\pi_k)$) denote the starting (completion) time of operation Q_{ij} in the semiactive schedule defined by a permutation π_k of the n jobs from set J.

Since the minimization of $C_{max}(\pi_k) = \max\{c_{ij}(\pi_k) : J_i \in J, j \in \{1, 2\}\}$ is a *regular* criterion, we consider only semiactive schedules. Each semiactive schedule is uniquely defined by a permutation of the n jobs on each machine. For the two-machine flow shop problem, it is sufficient to consider the same

permutation $\pi_k = (J_{k_1}, J_{k_2}, \ldots, J_{k_n})$ of the jobs on each of the two machines, since the set of such schedules is dominant for minimizing the makespan.

NP-hardness of Problem $\mathcal{F}2,NC^{off}/pmtn, w = 1/\mathcal{C}_{max}$

Next, we prove that problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ is binary NP-hard even if there exists only one non-availability interval ($w = 1$) either on machine M_1 or on machine M_2.

Theorem 3.23 *If $w_1 = 1$ and $w_2 = 0$, then problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ is binary NP-hard.*

Proof. We prove the theorem by transforming polynomially the NP-hard Partition problem [125] into problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ with $w_1 = 1$ and $w_2 = 0$. The Partition problem can be described as follows.

Given a finite set $N^0 = \{1, 2, \ldots, n_0\}$ of indices and positive integers $e_1, e_2, \ldots, e_{n_0}$ which sum up to $2E = \sum_{i\in N^0} e_i$, does there exist a partition of set N^0 into two subsets N_1^0 and N_2^0 (i.e., $N^0 = N_1^0 \cup N_2^0$, $N_1^0 \cap N_2^0 = \emptyset$) such that $\sum_{i\in N_1^0} e_i = \sum_{i\in N_2^0} e_i = E$?

Let $e_{max} = \max\{e_i \mid i \in N^0\}$. We can assume that

$$e_{max} < E + 1. \tag{3.41}$$

Indeed, if $e_{max} \geq E+1$, then the Partition problem definitely has no solution.

Given an instance of the Partition problem, we generate a set of data for the instance of problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ as follows. Let $p_{i1} = e_i$ and $p_{i2} = e_i(E + 1)$ for each $i \in N^0$. In addition, let $w_1 = 1$, $w_2 = 0$, $n = n^0 + 1$, $p_{n1} = 0$, $p_{n2} = e_{max}$, $s(N_{1,1}) = E$, and $f(N_{1,1}) = E^2 + E$.

Does there exist a schedule for the above instance of problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ with a makespan of no more than $2E^2 + 2E + e_{max}$?

Sufficiency. Let there exist a solution $N^0 = N_1^0 \cup N_2^0$, $N_1^0 \cap N_2^0 = \emptyset$, of the above instance of the Partition problem: $\sum_{i\in N_1^0} e_i = \sum_{i\in N_2^0} e_i = E$. Then we can construct a desired schedule as follows. First schedule job J_n, then the jobs J_i with $i \in N_1^0$ in an arbitrary order, and finally the jobs J_j with $j \in N_2^0$ in an arbitrary order as well. This schedule is represented in Figure 3.4. Let N' be any subset of set N_1^0 and let J_k be the last job of the ordered subset N'. Due to equality $p_{i2} = p_{i1}(E + 1)$, we obtain

$$\sum_{i\in N'} p_{i1} \leq e_{max} + \sum_{i\in N'} p_{i1} - p_{k1} \leq p_{n2} + (E + 1)(\sum_{i\in N'} p_{i1} - p_{k1})$$

$$= p_{n2} + \sum_{i\in N'} (E + 1)p_{i1} - (E + 1)p_{k1} = p_{n2} + \sum_{i\in N'} p_{i2} - p_{k2}.$$

Furthermore, the last job in the ordered set N_1^0 finishes at time $e_{max} + (E + 1)E$ on machine M_2. Hence, there is no idle time on machine M_2 from time $t_0 = 0$ up to time $e_{max} + (E + 1)E = E^2 + E + e_{max}$.

Similar arguments can be applied to show that there is no idle time on machine M_2 for processing the set of jobs $\{J_i \mid i \in N_2^0\}$. Hence, the makespan of this schedule is equal to $e_{max} + (E + 1)E + (E + 1)E = 2E^2 + 2E + e_{max}$. Sufficiency has been proven.

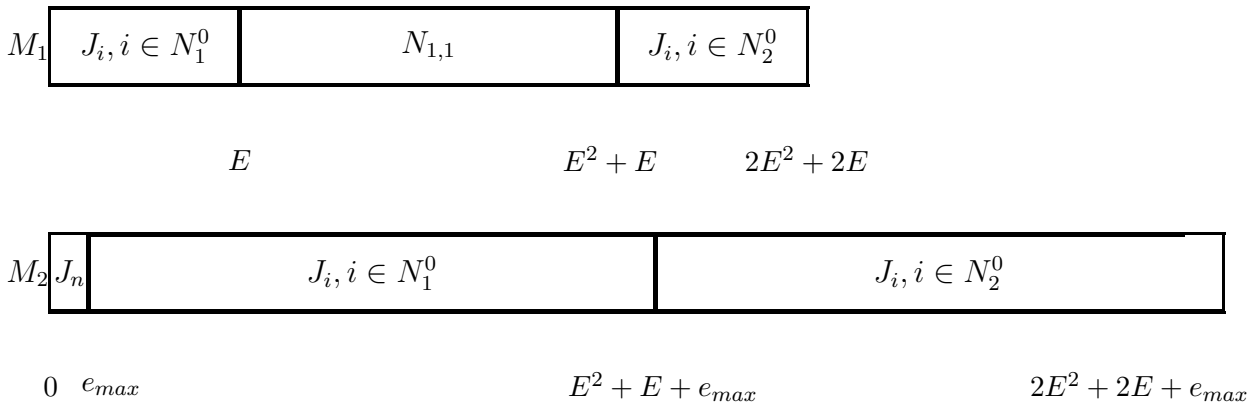


Figure 3.4: *Optimal schedule for problem* $\mathcal{F}2,NC^{off}/pmtn, w = 1/\mathcal{C}_{max}$

Necessity. Suppose that there exists a schedule with the makespan $2E^2 + 2E + e_{max}$ for the above instance of problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$.

Since $\sum_{i \in J} p_{i2} = 2E^2 + 2E + e_{max}$, there is no idle time in this schedule for machine M_2. Hence, job J_n must be processed first in this schedule.

Aiming for contradiction, we suppose that there exists no solution for the above instance of the Partition problem. Then there is no subset of jobs J that can have a finishing time on machine M_1 at time point $s(N_{1,1}) = E$. Moreover, since all the numbers e_i are integers, for any subset of jobs $\{J_i \mid i \in N' \subseteq N^0\}$ that finish before time point E on machine M_1, we obtain

$$\sum_{i \in N'} p_{i1} \leq E - 1.$$

Therefore, equalities $p_{i2} = p_{i1}(E + 1)$, $i \in N' \subseteq N^0$, imply

$$\sum_{i \in N'} p_{i2} = (E + 1) \sum_{i \in N'} p_{i1} \leq (E + 1)(E - 1) = E^2 - 1.$$

Due to inequality (3.41), we obtain

$$e_{max} + \sum_{i \in N'} p_{i2} \leq e_{max} + E^2 - 1 < E^2 + E = f(N_{1,1}).$$

Therefore, there must exist an idle time on machine M_2. This contradiction completes the necessity proof.

Thus, we constructed a polynomial transformation of the binary NP-hard Partition problem to problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ with $w_1 = 1$ and $w_2 = 0$. Therefore, the latter problem is binary NP-hard as well.

$\diamond$

The following claim can be proven similarly.

Theorem 3.24 *If $w_1 = 0$ and $w_2 = 1$, then problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ is binary NP-hard.*

Motivation and Example

Let $\pi_i \in S^\pi$ be a Johnson permutation of the n jobs of set J, i.e., permutation π_i satisfies condition (3.2) given on page 181. Such a permutation is optimal for the makespan criterion when all machines are continuously available ($w = 0$) during the planning horizon (see Section 3.1). For the purposes of this and the next sections, we can assume that the Johnson permutation π_i is constructed by Johnson's algorithm given on page 181 (see Remark 3.1 on page 182). We shall study the question whether this permutation remains optimal for the case of $w \geq 1$ non-availability intervals of each of the two machines known offline. In a concrete schedule, enlargements of time intervals used for processing jobs may be caused by non-availability intervals of machines. The main idea realized in this section is to consider the non-availability interval on a machine as an additional part of the job processing time. In order to take into account such an increase of the total time interval used for processing a job, we compute the *stability polytope* and the *stability radius* ρ_j of a Johnson permutation which is the minimum of the maximal possible enlargements r_{ij} of the jobs $J_i \in J$ on machine $M_j \in \{M_1, M_2\}$ such that the Johnson permutation is not changed. Such a stability radius ρ_j can be computed in $O(n)$ time.

We compute also the *enlargement polytope* and the *enlargement radius* δ_j of the processing times of the operations on machine M_j which denotes the maximum of the possible enlargements d_{ij} of the time intervals used for processing job J_i on machine M_j caused by non-availability intervals. It is shown that a Johnson permutation $\pi_k \in S^\pi = \{\pi_1, \pi_2, \ldots, \pi_{n!}\}$ for problem $\mathcal{F}2//\mathcal{C}_{max}$ remains *optimal* for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ if inequality $d_{ij} \leq r_{ij}$ holds for each $i \in \{1, 2, \ldots, n\}$ and each $j \in \{1, 2\}$. The enlargement polytopes and the radii δ_j can be computed in $O(w^2 + n \log_2 n)$ time.

Further, we present sufficient conditions for the stability of any optimal permutation $\pi_l \in S^{\pi}$. At the end of this section, the results obtained for a stability analysis are demonstrated on a huge number of randomly generated two-machine flow shop problems with $5 \leq n \leq 10000$ and $1 \leq w \leq 1000$.

In [177], it has been shown that a semiactive schedule defined by a Johnson permutation is optimal for the case when all machines are continuously available throughout the planning horizon, i.e., when $w = 0$ (see Section 3.1). Moreover, preemptions of operations cannot decrease the objective function value $\mathcal{C}_{max}$, and so the instance of problem $\mathcal{F}2//\mathcal{C}_{max}$ and the instance of problem $\mathcal{F}2/pmtn/\mathcal{C}_{max}$ with the same input data have the same optimal schedule defined by a Johnson permutation. The following example is used to illustrate the calculations.

Example 3.7 *We consider a three-job two-machine flow shop problem with $w = 9$ non-availability intervals on the two machines defined in Table 3.12. Table 3.13 presents the given processing times p_{ij} of the jobs $J_i \in J = \{J_1, J_2, J_3\}$ on the machines $M_j \in M = \{M_1, M_2\}$. Next, we*

Table 3.12: *Non-availability intervals of the machines in Example 3.7*

Machine M_1	
$s(N_{1,1}) = 2$	$f(N_{1,1}) = 3$
$s(N_{2,1}) = 11$	$f(N_{2,1}) = 13$
$s(N_{3,1}) = 16$	$f(N_{3,1}) = 17$
$s(N_{4,1}) = 19$	$f(N_{4,1}) = 20$
$s(N_{5,1}) = 24$	$f(N_{5,1}) = 26$
$w_1 = 5$	

Machine M_2	
$s(N_{1,2}) = 1$	$f(N_{1,2}) = 4$
$s(N_{2,2}) = 9$	$f(N_{2,2}) = 10$
$s(N_{3,2}) = 14$	$f(N_{3,2}) = 15$
$s(N_{4,2}) = 21$	$f(N_{4,2}) = 22$
$w_2 = 4$	

Table 3.13: *Job processing times for Example 3.7*

Machine M_1	Machine M_2
$p_{1,1} = 3$	$p_{1,2} = 5$
$p_{2,1} = 4$	$p_{2,2} = 1$
$p_{3,1} = 7$	$p_{3,2} = 2$

show how to compute the values $r_{ij}, \rho_j, s_j, c_j, d_{ij}$ and δ_j for the jobs $J_i \in J$ and the machines $M_j \in \{M_1, M_2\}$. For this example, it will be shown that the schedule with the jobs following the Johnson permutation $\pi_k = (1, 3, 2)$ applied to the situation when the machines are not continuously available remains optimal.

Let π_k be a Johnson permutation (which is optimal when both machines are continuously available during the planning horizon, i.e., for problem $\mathcal{F}2//\mathcal{C}_{max}$). Our aim is to answer the question whether this permutation remains optimal for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ as well, i.e., when $w \geq 1$ non-availability intervals are given before scheduling and the operation processing times are the same as in problem $\mathcal{F}2//\mathcal{C}_{max}$. A non-availability interval N_{kj} on machine M_j may be interpreted as an additional part of the processing time of the operation Q_{ij} the processing of which on machine M_j is preempted due to interval N_{kj}. Indeed, machine M_j cannot process any job during the interval N_{kj} and job J_i has to wait until the endpoint of interval N_{kj}. In a concrete schedule, an enlargement equal to $h(N_{kj})$ of the completion time of job J_i is caused by the non-availability interval N_{kj}.

Next, we show how to compute the *stability radius* ρ_j for machine M_j which is the minimum of all maximal possible enlargements r_{ij} of the processing times of the jobs $J_i \in J$ on machine M_j such that the Johnson permutation is not changed. Then we show how to compute the *enlargement radius* δ_j of the processing times of the operations on machine M_j which is the maximum of all maximal possible enlargements d_{ij} of the jobs $J_i \in J$ on machine M_j caused by the non-availability intervals on machine M_j. The Johnson permutation π_k remains optimal if $d_{ij} \leq r_{ij}$ for all jobs $J_i \in J$ and machines $M_j \in M$.

Stability Polytopes for a Johnson Permutation

The general definition of the stability radius of an optimal schedule was given in Chapter 1. Here we need simplified versions of the stability radius since non-availability intervals can increase the time interval used for processing the operation but they cannot decrease this interval. Moreover, the stability radius considered in Chapter 1 defines the maximal independent simultaneous variations of the processing times such that the given schedule (digraph G_s) remains *optimal*. In Definition 3.4 which follows, we do not take care directly about the *optimality* of a permutation but we are concerned with the property of the given permutation to be a Johnson one after increasing the processing times of all or a portion of the jobs. It should be noted that the optimality of a permutation of n jobs does not imply that this permutation is a Johnson one even for problem $\mathcal{F}2//\mathcal{C}_{max}$ with $w = 0$ (see Remark 3.2 on page 182).

Definition 3.4 *The stability radius ρ_j of a Johnson permutation π_s (constructed by Johnson's algorithm) on machine M_j is defined as the minimum*

value among all maximal possible enlargements r_{ij} of the operations Q_{ij}, $J_i \in J$, on machine $M_j \in \{M_1, M_2\}$ such that permutation π_s necessarily remains a Johnson permutation for the modified processing times. The closed polytopes $P_j = \{x = (x_{1,j}, x_{2,j}, \ldots, x_{nj}) : p_{ij} \leq x_{ij} \leq p_{ij} + r_{ij},\ J_i \in J\}$, $M_j \in \{M_1, M_2\}$, in the space R^n_+ of non-negative n-dimensional real vectors are called stability polytopes of a Johnson permutation.

A stability polytope P_j defines all possible enlargements of the processing times of the operations Q_{ij} on machine M_j such that the given permutation π_s of the jobs J remains a Johnson permutation. As a result, permutation π_s is optimal for problem $\mathcal{F}2//\mathcal{C}_{max}$ with the modified processing times. Note that in this section, we investigate the stability properties of a fixed permutation π_s from the set of all permutations $S^\pi = \{\pi_1, \pi_2, \ldots, \pi_{n!}\}$ of the n jobs J. That is why for simplicity, we do not indicate the permutation π_s or the index s in the notations ρ_j, P_j etc. Next, we prove the following lemma.

Lemma 3.6 *The stability radii ρ_j and the stability polytopes P_j, $j \in \{1, 2\}$, can be computed in $O(n)$ time.*

PROOF. Let π_s be a Johnson permutation constructed by Johnson's algorithm. W.l.o.g. we assume that $\pi_s = (J_1, J_2, \ldots, J_n)$. Let the first k jobs belong to set N_1 and the remaining $n - k$ jobs belong to set N_2, i.e., $N_1 = \{J_1, J_2, \ldots, J_k\}$ and $N_2 = \{J_{k+1}, J_{k+2}, \ldots, J_n\}$. (For the definition of the sets N_1 and N_2, see Johnson's algorithm given on page 181 in Section 3.1.) To compute the numbers r_{ij}, we distinguish two cases: $J_i \in N_1$ or $J_i \in N_2$. In the case when $J_i \in N_1$, inequality $p_{i1} \leq p_{i2}$ holds, and the maximal possible enlargements r_{i1} are the minimum of the two values $a^{(i1)}$ and $b^{(i1)}$, where $a^{(i1)} = p_{i2} - p_{i1}$ represents the maximum amount of the processing time one can add to value p_{i1} such that job J_i will remain in set N_1 due to Johnson's algorithm. The value

$$b^{(i1)} = \begin{cases} p_{i+1,1} - p_{i1}, & i = 1, 2, \ldots, k-1, \\ \infty, & i = k, \end{cases}$$

represents the maximum amount of the processing time (or infinity if $i = k$) one can add to value p_{i1} such that the SPT (shortest processing time) ordering within set N_1 is still preserved.

Similar arguments are valid for the case $J_i \in N_2$, i.e., when inequality $p_{i1} \geq p_{i2}$ holds. Every computation of an enlargement r_{ij} can be done

in a constant time, and the stability polytopes P_j and the stability radii $\rho_j,\ j \in \{1,2\}$, can be computed in $O(n)$ time.

$\diamond$

Example 3.7 (continued). *In Table 3.14, the results of the calculation of the values r_{ij} and ρ_j are shown for Example 3.7. If no processing time on*

Table 3.14: *Stability radii $\rho_j, M_j \in \{M_1, M_2\}$, for Example 3.7*

Machine M_1	
$p_{1,1} = 3$	$r_{1,1} = \min\{2, \infty\} = 2$
$p_{2,1} = 4$	$r_{2,1} = \infty$
$p_{3,1} = 7$	$r_{3,1} = \infty$
	$\rho_1 = 2$

Machine M_2	
$p_{1,2} = 5$	$r_{1,2} = \infty$
$p_{2,2} = 1$	$r_{2,2} = \min\{3, 1\} = 1$
$p_{3,2} = 2$	$r_{3,2} = \min\{5, \infty\} = 5$
	$\rho_2 = 1$

machine M_1 is enlarged by more than two units and if no processing time on machine M_2 is enlarged by more than one unit, then permutation $\pi_k = (1,3,2)$ remains a Johnson permutation for the non-availability intervals given in Table 3.12.

Enlargement Polytopes

If $w_j = 0$, operation Q_{ij} of job J_i on machine M_j may be processed without preemptions and so the length of the interval for scheduling operation Q_{ij} is equal to p_{ij}. However, if $w_j \geq 1$, the *scheduling time* (i.e., the difference between the completion time and the starting time) of operation Q_{ij} may be increased by the lengths of some non-availability intervals. Next, we determine the maximal possible enlargements of the scheduling times of the operations Q_{ij} on machine $M_j \in \{M_1, M_2\}$ caused by non-availability intervals independently of a concrete schedule under consideration.

Definition 3.5 *The enlargement radius δ_j of the operations on machine M_j is defined as the maximum of the possible enlargements d_{ij} of the scheduling times of operations $Q_{ij},\ J_i \in J$, caused by non-availability intervals on machine M_j. The closed polytopes $\Delta_j = \{x = (x_{1j}, x_{2j}, \ldots, x_{nj}) : p_{ij} \leq x_{ij} \leq p_{ij} + d_{ij},\ J_i \in J\}$, $M_j \in \{M_1, M_2\}$, in the space R^n_+ are called the enlargement polytopes.*

The enlargement radius gives the maximum amount by which the completion time of an operation on a machine may be delayed by one or more

consecutive non-availability intervals which may be included in the scheduling interval for processing the operation. We prove the following lemma.

Lemma 3.7 *The enlargement radii δ_j and the enlargement polytopes Δ_j, $M_j \in \{M_1, M_2\}$, can be computed in $O(w^2 + n \log_2 n)$ time.*

PROOF. For machine M_1, we do the following. If operation Q_{i1} starts or is resumed immediately after a non-availability interval on machine M_1, we have to add the length of this non-availability interval to the scheduling time of operation Q_{i1}. We compute the maximal sum D_i of the lengths of i consecutive non-availability intervals of machine M_1 for $i = 1, 2, \ldots, w_1$:

$$D_i = \max_{a=1}^{w_1+1-i} \left\{ \sum_{k=a}^{a+i-1} h(N_{k1}) \right\}.$$

Using the given non-availability intervals, we can determine the *availability intervals* of machine $M_j \in \{M_1, M_2\}$.

Let A_{kj} denote the k^{th} availability interval, i.e., the k^{th} maximal interval when machine M_j is available to process a job. Let $s(A_{kj})$, $f(A_{kj})$ and $h(A_{kj})$ be the starting point, the endpoint and the length of the interval A_{kj}, respectively. W.l.o.g. we assume that $s(N_{1,1}) > 0$. Indeed, if $s(N_{1,1}) = 0$, we can change the earliest possible starting time $s_1 = 0$ on machine M_j by $s_1 = f(N_{1,1})$ and an optimal permutation remains the same. Thus, we obtain $s(A_{1,1}) = 0$, $f(A_{1,1}) = s(N_{1,1})$ and $h(A_{1,1}) = f(A_{1,1}) - s(A_{1,1}) > 0$.

For $k = 2, 3, \ldots, w_1$, we obtain $s(A_{k1}) = f(N_{k-1,1})$, $f(A_{k1}) = s(N_{k1})$ and $h(A_{k1}) = f(A_{k1}) - s(A_{k1}) > 0$.

We set $E_0 = 0$ and compute the sum E_i of the lengths of i consecutive availability intervals A_{k1} of machine M_1 for $i = 1, 2, \ldots, w_1 + 1$:

$$E_i = \min_{a=1}^{w_1+1-i} \left\{ \sum_{k=a}^{a+i-1} h(A_{k1}) \right\}.$$

These computations need $O(w^2)$ time. Using the numbers D_k and E_k for $k = 1, 2, \ldots, w_1$, we can calculate the maximal possible enlargement of the scheduling time for processing the operation Q_{i1} of job J_i on machine M_1. Indeed, to process operation Q_{i1}, at most k consecutive availability intervals of machine M_1 have to be used if $E_{k-1} \leq p_{i1} < E_k$. These k availability intervals are alternated with k non-availability intervals of machine M_1. Therefore, the scheduling time for processing operation Q_{i1} may be increased by at most D_k. Hence, the scheduling time for processing each operation Q_{i1} on machine M_1 with $E_{k-1} \leq p_{i1} < E_k$ may have the maximal

enlargement caused by non-availability intervals with a total length equal to D_k. Similar arguments are valid for machine M_2.

Thus, the calculation of the enlargement of each job from set J takes $O(w + n\log_2 n)$ time, so we need $O(w^2 + n\log_2 n)$ time for the calculation of the enlargement radii δ_j and the enlargement polytope $\Delta_j,\ j \in \{1, 2\}$.

$\diamond$

Example 3.7 (continued). *In Table 3.15, the enlargement radii for machines M_1 and M_2 are presented for Example 3.7. Thus, any scheduling time of an operation on machine M_1 (machine M_2) cannot be enlarged by more than three units (by more than two units, respectively) if one sticks non-availability intervals with jobs J. Similarly, any scheduling time on machine M_2 cannot be enlarged by more than two units.*

Table 3.15: *Enlargement radii $\delta_j, M_j \in \{M_1, M_2\}$, for Example 3.7*

Machine M_1	
$p_{1,1} = 3$	$d_{1,1} = 2$
$p_{2,1} = 4$	$d_{2,1} = 3$
$p_{3,1} = 7$	$d_{3,1} = 3$
	$\delta_1 = 3$

Machine M_2	
$p_{1,2} = 5$	$d_{1,2} = 2$
$p_{2,2} = 1$	$d_{2,2} = 1$
$p_{3,2} = 2$	$d_{3,2} = 1$
	$\delta_2 = 2$

Using Lemma 3.6 and Lemma 3.7, we prove the following theorem.

Theorem 3.25 *A Johnson permutation π_s defined for problem $\mathcal{F}2//\mathcal{C}_{max}$ remains optimal for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ with the same processing times if*

$$d_{ij} \le r_{ij},\ i \in \{1, 2, \ldots, n\},\ j \in \{1, 2\}. \tag{3.42}$$

Testing condition (3.42) takes $O(w^2 + n\log_2 n)$ time.

Proof. We prove this claim by contradiction. Let there exist a permutation $\pi_k \in S^\pi$ of the n jobs J such that

$$C_{max}(\pi_k) < C_{max}(\pi_s) \tag{3.43}$$

for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ mentioned in Theorem 3.25. We define a modification (we call it Problem 1) of the above problem $\mathcal{F}2//\mathcal{C}_{max}$ which has the following processing times p'_{ij} of the operations Q_{ij}:

$$p'_{ij} = \begin{cases} h(N_{kj}) + c_{ij}(\pi_k) - s_{ij}(\pi_k), & \text{if there exists an } N_{kj} \\ & \text{with } f(N_{kj}) = s_{ij}(\pi_k), \\ c_{ij}(\pi_k) - s_{ij}(\pi_k) & \text{otherwise.} \end{cases}$$

Due to Definition 3.5, we have $p'_{ij} \leq p_{ij} + d_{ij}$. Due to Definition 3.4 and inequalities (3.42), permutation π_s remains a Johnson permutation for Problem 1. As far as $w = 0$ for Problem 1, the Johnson permutation π_s is optimal. Therefore, $C_{max}(\pi_s) \leq C_{max}(\pi_k)$ which contradicts to (3.43).

The complexity of the calculation of the values r_{ij} and d_{ij} given in Lemma 3.6 and Lemma 3.7 defines the complexity of testing inequalities (3.42).

$\diamond$

Next, we present more simple sufficient conditions for the optimality of a permutation using the information about the location of some operations Q_{ij} and non-availability intervals N_{kj} in the concrete schedule.

Sufficient Conditions for the Stability of an Optimal Permutation

Let $\pi_v = (J_{i_1}, J_{i_2}, \ldots, J_{i_n})$ be an optimal permutation for problem $\mathcal{F}2//\mathcal{C}_{max}$, $s(\pi_v)$ be the schedule for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ defined by the permutation π_v, and $c_{i_kj}(s(\pi_v))$ denotes the completion time of operation Q_{i_kj} in the schedule $s(\pi_v)$ for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ with $1 \leq k \leq n$.

Theorem 3.26 *Permutation π_v which is optimal for problem $\mathcal{F}2//\mathcal{C}_{max}$ remains optimal for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ with the same processing times if there exists a time point $t = c_{i_k1}(s(\pi_v))$, $1 \leq k \leq n$, for schedule $s(\pi_v)$ such that*

(i) the shortest k operations on machine M_1 and the non-availability intervals of machine M_1 completely fill the time interval $[0, t]$,

(ii) the shortest $n + 1 - k$ operations on machine M_2 and the non-availability intervals of machine M_2 completely fill the time interval $[t, c_{i_n2}(s(\pi_v))]$.

Proof. Since in schedule $s(\pi_v)$, there are no idle times on machine M_1 in the interval $[0, t]$ and on machine M_2 in the interval $[t, c_{i_n2}(s(\pi_v))]$, any transposition of jobs within the set $\{i_1, i_2, \ldots, i_k\}$ cannot decrease the value $C_{\max}(s(\pi_v))$. Similar arguments are valid for any transposition of jobs within the set $\{i_k, i_{k+1}, \ldots, i_n\}$. Let s be any semiactive schedule constructed for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ under consideration. In any semiactive schedule, machine M_1 has no idle time which belongs to availability intervals within the interval $[1, c_{i_n1}(s)]$. Thus, the operations Q_{t1}, $k + 1 \leq t \leq n$, and the non-availability intervals of machine M_1 completely fill the interval $[t, c_{i_n1}(s(\pi_v))]$. From condition (ii), it follows that

for schedule $s(\pi_v)$, only the interval $[0,t]$ may include an idle time of machine M_2 which belongs to availability intervals. Let $l(s(\pi_v))$ denote the total length of such idle times. If $l(s(\pi_v)) = 0$, then schedule $s(\pi_v)$ is optimal for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$. If $l(s(\pi_v)) > 0$, then decreasing the value $C_{\max}(s(\pi_v)) = c_{i_n 2}(s(\pi_v))$ is equivalent to decreasing the total length $l(s(\pi_v))$ of idle times.

To finish the proof, we have to consider each semiactive schedule s obtained from schedule $s(\pi_v)$ after a transposition of job J_{i_l}, $1 \leq l \leq k$, and job J_{i_m}, $k+1 \leq m \leq n$. From condition (i), it follows that machine M_1 cannot completely process more than k operations within the interval $[0,t]$, and machine M_2 cannot completely process more than $k-1$ operations within the interval $[0,t]$. Consequently, from condition (ii), it follows that the total idle time of machine M_2 which belongs to availability intervals within the interval $[0,t]$ in schedule s cannot be less than $l(s(\pi_v))$, and thus $C_{\max}(s(\pi_v)) \leq C_{\max}(s)$.

$\diamond$

Note that in the above proof of Theorem 3.26, we do not use that π_v is a Johnson permutation for problem $\mathcal{F}2//\mathcal{C}_{max}$ indicated in Theorem 3.26. Indeed, if conditions (i) and (ii) are valid for any permutation π_v which is optimal for problem $\mathcal{F}2//\mathcal{C}_{max}$ (but π_v may not satisfy condition (3.2) given on page 181), then permutation π_v remains optimal for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ as well. The following corollaries are also valid for any permutation π_v which is optimal for problem $\mathcal{F}2//\mathcal{C}_{max}$.

Corollary 3.2 *If machine M_2 is filled in the interval $[c_{i_n 1}(s(\pi_v)), c_{i_n 2}(s(\pi_v))]$ only with the shortest operation on machine M_2 and non-availability intervals, then schedule $s(\pi_v)$ is optimal for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$.*

Corollary 3.3 *If $p_{i_1 1} = \min\{p_{i_k 1} \mid 1 \leq k \leq n\}$ and machine M_2 has no idle time in the interval $[c_{i_1 1}(s(\pi_v)), c_{i_n 2}(s(\pi_v))]$, then schedule $s(\pi_v)$ is optimal for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$.*

For the proof of Corollary 3.2 (Corollary 3.3), it is sufficient to note that conditions (i) and (ii) from Theorem 3.26 necessarily hold for $k = n$ (for $k = 1$, respectively). Again, it is not assumed that permutation π_v used in Corollary 3.2 and in Corollary 3.3 satisfies condition (3.2), however, these corollaries are often realized just for a Johnson permutation. It is easy to see that testing Corollary 3.2 takes $O(w_2)$ time, while testing Corollary 3.3 takes $O(w_2 + n)$ time. In order to reduce the number of non-availability intervals

that have to be considered for testing Theorem 3.25 and Theorem 3.26, one can use the following lemma.

Lemma 3.8 *The earliest possible starting times s_j and the latest possible completion times c_j of any job from set J on machine $M_j, j \in \{1,2\}$, in a semiactive schedule constructed for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ can be computed in $O(w+n)$ time.*

PROOF. On machine M_1, the earliest starting time of a job is equal to

$$s_1 = \max \begin{cases} f(N_{1,1}), & \text{if } s(N_{1,1}) = 0, \\ 0 & \text{otherwise.} \end{cases}$$

The latest completion time of a job from set J on machine M_1 is equal to $c_1 = p_1 + h + g$ with $p_1 = \sum p_{i,1}$ and $h = \sum_{i=1}^{k} h(N_{i,1})$ with $s(N_{k,1}) < p_1$ for the maximal k. Let g initially be 0, and k be taken from the former computation. The value g is increased by $h(N_{k+1,1})$ and k is increased by 1 as long as $s(N_{k+1,1}) < p_1 + h + g$. The computation of p_1 takes $O(n)$ time, and the computation of h and g takes $O(w)$ time. Hence, we need $O(w+n)$ time to compute the latest completion time of a job from set J on machine M_1.

Let $p_{min}(1)$ be the smallest processing time of an operation on machine M_1. On machine M_2, the earliest possible starting time of any job is the completion time of the first operation (which is assumed to be the shortest one) on machine M_1. Let $d = p_{min}(1) + c$. The values c and k are initially equal to 0. The value c is increased by $h(N_{k+1,1})$ and k is increased by 1 as long as $s(N_{k+1,1}) < p_{min}(1) + c$. Then we have

$$s_2 = \max \begin{cases} f(N_{is}), & \text{if there is a non-availability interval} \\ & N_{i,2} \text{ with } s_{i2} \le d \text{ and } c_{i2} \ge d, \\ d & \text{otherwise.} \end{cases}$$

The above computation takes $O(w+n)$ time. Let $p_{max}(2)$ be the largest processing time of an operation on machine M_2. The latest possible completion time of any job on machine M_2 is estimated by the upper bound $c_2 = c_1 + \sum p_{i2} + d$. The value k is the maximum index with $s(N_{k2}) \le c_1$. The value d is initially equal to $\max\{f(N_{k2}) - c_1, 0\}$. The value d is increased by $h(N_{k+1,2})$ and k is increased by 1 as long as $s(N_{k+1,2}) < c_1 + \sum p_{i2} + d$. This computation takes $O(w+n)$ time.

◇

Example 3.7 (continued). *Table 3.14 and Table 3.15 show the results for the computation of the stability and enlargement radii for machine M_1 and machine M_2. One can see that due to Theorem 3.25, a Johnson permutation $(1, 3, 2)$ remains optimal for the non-availability case under consideration since $d_{ij} \leq r_{ij}$ for all $J_i \in J$ and $j \in \{1, 2\}$. In addition, Corollary 3.2 and Theorem 3.26 are valid for this example of problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$.*

In Example 3.7 and in the computational experiments, non-negative integer numbers are used as processing times and the lengths of the non-availability intervals. However, it is obvious that all the above results are valid for non-negative real input data as well.

Computational Experiments

A stability analysis was performed on a huge number of randomly generated problems $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$. For each instance with $w \geq 1$, we constructed at least one Johnson permutation for the corresponding flow shop problem $\mathcal{F}2//\mathcal{C}_{max}$ with the same processing times but without non-availability intervals ($w = 0$), and answered the question: Is this permutation optimal for the original problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ with $w \geq 1$ given non-availability intervals? To this end, we tested the sufficient conditions proven in this section. To minimize the running time of the algorithm, these conditions were tested in an increasing order of their complexity up to the first positive answer (if any) to the above question. More formally, the following algorithm has been realized.

Algorithm for $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$

Input: Processing times of the jobs J and
non-availability intervals of the machines M.
Output: An optimal schedule for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ or
a feasible schedule if algorithm terminates in *Step 6*.

Step 1: Construct a Johnson permutation π for problem $\mathcal{F}2//\mathcal{C}_{max}$.
Step 2: **IF** π satisfies Corollary 3.2 **THEN GOTO** *Step 7*.
Step 3: **IF** π satisfies Corollary 3.3 **THEN GOTO** *Step 7*.
Step 4: Construct $\lambda = \min\{\lambda^*, 2^k\}$ Johnson permutations
$\pi_1, \pi_2, \ldots, \pi_\lambda$.
Step 5: **FOR** $i = 1, 2, \ldots, \lambda$ **DO**
BEGIN

Set $\pi := \pi_i$.
IF π satisfies Corollary 3.2 **THEN GOTO** *Step 7.*
IF π satisfies Corollary 3.3 **THEN GOTO** *Step 7.*
IF π satisfies Theorem 3.26 **THEN GOTO** *Step 7.*
IF π satisfies Theorem 3.25 **THEN GOTO** *Step 7.*
END

Step 6: Optimality of permutations π_i, $i = 1, 2, \ldots, \lambda$, for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ is not proven **STOP**.

Step 7: Permutation π is optimal for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ **STOP**.

Steps 4 and 5 in the above algorithm are realized since there may be more than one Johnson permutation constructed by Johnson's algorithm. Indeed, if there are k equalities $p_{ij} = p_{kj}$ for different jobs $J_i \in J$ and $J_k \in J$, then there may be 2^k possible Johnson permutations. So, there were tested up to $\min\{\lambda^*, 2^k\}$ Johnson permutations for each instance. In the experiments, we set $\lambda^* = 1024$.

Table 3.16: *Percentage of solved instances with $w_1 > 0$ and $w_2 > 0$*

Number of jobs	Number of non-availability intervals									
	1	2	3	4	5	6	7	8	9	10
5	74.2%	72.1%	72.3%	72.7%	72.3%	73.4%	74.1%	74.6%	75.1%	76.2%
10	87.5%	85.0%	84.6%	83.8%	83.5%	83.4%	83.5%	83.1%	84.4%	83.5%
15	91.1%	90.7%	90.0%	89.9%	89.6%	89.0%	88.7%	87.9%	88.5%	88.5%
20	93.4%	93.2%	92.5%	92.3%	92.1%	91.9%	91.7%	91.3%	90.8%	91.3%
25	94.5%	94.1%	94.3%	93.6%	93.3%	93.7%	93.8%	93.0%	93.2%	92.4%
30	95.1%	94.9%	95.0%	95.1%	94.7%	94.5%	93.9%	94.2%	94.0%	94.4%
35	95.5%	95.7%	95.5%	95.9%	95.4%	95.3%	95.5%	95.4%	95.1%	94.9%
40	96.0%	96.2%	96.1%	96.3%	96.1%	96.1%	95.9%	95.8%	95.5%	95.6%
45	96.6%	96.9%	96.6%	96.4%	96.2%	96.5%	96.3%	96.2%	96.1%	96.1%
50	97.2%	97.0%	96.8%	96.6%	96.7%	96.7%	96.9%	96.6%	96.9%	96.4%
55	97.2%	97.2%	97.2%	97.5%	96.8%	97.1%	97.2%	97.1%	96.8%	96.8%
60	97.7%	97.4%	97.6%	97.2%	97.3%	97.3%	97.2%	97.1%	97.2%	97.1%
65	97.6%	97.4%	97.7%	97.7%	97.3%	97.3%	97.3%	97.2%	97.3%	97.4%
70	97.8%	97.5%	97.9%	98.0%	97.5%	97.9%	97.7%	97.2%	97.8%	97.5%
75	97.7%	98.1%	97.9%	98.2%	97.8%	97.7%	97.8%	97.6%	97.8%	97.9%
80	98.1%	98.0%	98.1%	97.9%	98.1%	98.2%	98.0%	97.9%	97.6%	97.9%
85	98.1%	98.3%	98.2%	98.2%	98.3%	98.0%	98.2%	98.1%	98.0%	98.0%
90	98.5%	98.5%	98.3%	98.2%	98.3%	97.9%	98.1%	98.3%	98.2%	98.0%
95	98.3%	98.3%	98.4%	98.2%	98.1%	98.3%	98.2%	98.3%	98.2%	98.2%
100	98.5%	98.5%	98.5%	98.5%	98.4%	98.4%	98.3%	98.3%	98.2%	98.3%

For each combination of n and w, Tables 3.16 – 3.25 present the percent-

age of instances in the series for which the calculation based on the above algorithm found a Johnson permutation which remains optimal in spite of the w given non-availability intervals.

The above algorithm was coded in C++. For the computational experiments, we used an AMD 1200 MHz processor with 1024 MB main memory. For the small and moderate instances similar to those considered in [195], we made 10000 tests in each series, i.e., for each combination of n and w. These results are presented in Table 3.16 – Table 3.20. For the large instances presented in Tables 3.21 – 3.25, we made 1000 tests in each series.

Lemma 3.8 was used during the generation of random instances: The non-availability intervals were chosen in such a way that every non-availability interval was counted. This means that, if c_1 and c_2 are the completion times of the last job on machine M_1 and M_2 (in the schedule with non-availability intervals), then every non-availability interval was located either in the segment $[0, c_1]$ for machine M_1 or in the segment $[0, c_2]$ for machine M_2.

Problems of Small and Moderate Sizes

We tested problems similar to those considered in [195], i.e., with

- $5, 10, 15, \ldots, 100$ jobs with integer processing times uniformly distributed in the range $[1, 1000]$ and with
- $1, 2, 3, \ldots, 10$ non-availability intervals (on both machines) with integer lengths uniformly distributed in the range $[1, 1000]$.

In [195], a branch-and-bound algorithm was developed and used for constructing optimal schedules for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$. Due to the limit t of the running time used for each instance (the limit t was equal to 1000 seconds), not all instances were optimally solved within this limit t. For similar instances, we answer (by experiments) the question of how often a Johnson permutation (constructed for problem $\mathcal{F}2//\mathcal{C}_{max}$) remains optimal in spite of non-availability intervals on the machines (i.e., for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$). We tested only the sufficient conditions and our algorithm does not guarantee to find an optimal schedule for some problems $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$. However, our experiments have shown that such instances arise rather seldom for most randomly generated problems. We considered 10000 instances in each series of small problems (in [195], 10 instances were tested in each series).

Table 3.17: *Percentage of solved instances with $w = w_2$*

Number of jobs	Number of non-availability intervals									
	1	2	3	4	5	6	7	8	9	10
5	76.7%	75.0%	74.5%	75.2%	76.6%	77.3%	77.9%	77.6%	79.6%	80.3%
10	87.9%	86.4%	85.2%	84.4%	83.1%	83.7%	85.0%	84.5%	85.4%	85.6%
15	91.7%	91.1%	90.4%	88.9%	89.5%	88.9%	89.2%	88.9%	89.2%	88.9%
20	93.6%	92.6%	92.9%	92.2%	92.5%	91.5%	91.7%	91.0%	90.4%	90.5%
25	94.5%	94.5%	94.3%	93.5%	93.7%	92.9%	92.6%	93.2%	92.7%	92.6%
30	95.3%	95.0%	94.6%	94.5%	94.0%	94.1%	94.4%	94.0%	94.0%	93.7%
35	96.0%	95.8%	95.9%	95.7%	95.3%	95.4%	94.8%	94.5%	94.7%	94.8%
40	96.3%	96.4%	95.9%	95.8%	95.8%	96.2%	95.5%	96.0%	95.8%	95.4%
45	96.5%	96.8%	96.5%	96.5%	96.0%	96.4%	96.2%	96.0%	95.7%	96.0%
50	97.2%	96.8%	96.8%	96.7%	96.6%	96.8%	96.8%	96.5%	96.4%	96.0%
55	97.1%	97.4%	97.3%	97.2%	97.0%	96.9%	96.5%	96.8%	96.6%	96.8%
60	97.4%	97.4%	97.5%	97.5%	96.9%	97.1%	97.0%	96.9%	97.0%	96.8%
65	97.5%	97.7%	97.5%	97.5%	97.4%	97.3%	97.2%	97.4%	97.5%	96.8%
70	98.0%	98.0%	97.9%	97.7%	98.0%	97.7%	97.6%	97.4%	97.8%	97.5%
75	98.1%	98.0%	97.8%	97.7%	97.7%	97.8%	97.8%	97.8%	97.8%	97.9%
80	98.2%	98.2%	98.2%	98.0%	98.1%	97.9%	97.9%	97.8%	97.6%	97.6%
85	98.1%	98.3%	98.2%	98.2%	98.0%	97.8%	98.1%	98.1%	97.9%	98.1%
90	98.4%	98.2%	98.3%	98.1%	98.3%	98.2%	98.0%	98.1%	98.2%	98.0%
95	98.6%	98.4%	98.3%	98.4%	98.1%	98.2%	98.3%	98.2%	98.0%	97.8%
100	98.3%	98.3%	98.7%	98.5%	98.4%	98.2%	98.2%	98.2%	98.3%	98.2%

Tables 3.16 – 3.20 present the percentage of problem instances which were (optimally) solved due to the stability analysis via the algorithm described on page 248. We do not present the running times for series of small and moderate instances, since the running times were very close for different instances and very small: Among all instances in the series presented in Tables 3.16, 3.17 and 3.18, the maximum running time for an instance was 0.000125 seconds, and among the series presented in Tables 3.19 and 3.20, it was 0.000731 seconds.

Table 3.16 presents small problems. We can compare Table 3.16 with Table 1 in [195]. The percentage of solved instances in our experiments was less for $n = 5, n = 10$ and $n = 15$, but almost the same for $15 \leq n \leq 100$. Moreover, the branch-and-bound algorithm used essentially more running time (1000 seconds were not sufficient to solve some small instances).

Due to an NP-hardness proof and experiments, in [195], it was shown that instances in which the values p_{i2} were a double of the values p_{i1} (i.e., $p_{i2} = 2p_{i1}$) were much harder than instances with processing times uniformly generated for both machines. Our computations confirmed this property for a small problem size (compare Table 3.16 with Table 3.19, and Table 3.18

Table 3.18: *Percentage of solved instances with* $w = w_1$

Number of jobs	Number of non-availability intervals									
	1	2	3	4	5	6	7	8	9	10
5	71.4%	67.4%	67.0%	69.0%	69.6%	71.0%	72.5%	73.7%	73.8%	76.4%
10	86.2%	83.3%	82.2%	80.9%	81.2%	81.0%	81.2%	82.6%	81.6%	82.2%
15	91.2%	89.7%	89.5%	88.1%	87.7%	87.6%	87.1%	87.1%	88.0%	88.1%
20	93.0%	92.9%	92.0%	91.7%	90.6%	90.7%	90.5%	90.0%	90.5%	90.2%
25	94.1%	93.8%	93.8%	93.1%	92.8%	92.8%	92.5%	92.2%	92.4%	92.1%
30	95.1%	94.9%	94.2%	95.2%	94.7%	94.1%	94.2%	93.8%	93.9%	93.5%
35	96.1%	95.8%	95.5%	95.3%	94.9%	94.4%	95.1%	94.7%	94.3%	94.3%
40	96.5%	96.1%	95.8%	95.8%	95.9%	95.5%	95.2%	95.6%	95.1%	95.2%
45	97.0%	96.6%	96.4%	96.5%	96.1%	96.2%	95.7%	95.9%	95.9%	95.9%
50	96.9%	96.6%	96.4%	96.7%	96.9%	96.7%	96.5%	96.5%	96.5%	96.2%
55	97.4%	97.2%	96.7%	96.9%	96.9%	97.1%	97.0%	97.0%	96.8%	96.6%
60	97.5%	97.7%	97.2%	97.0%	97.3%	97.0%	97.1%	96.8%	97.1%	97.0%
65	97.9%	97.5%	97.5%	97.6%	97.6%	97.3%	97.4%	97.3%	97.2%	97.2%
70	97.5%	97.7%	98.0%	97.9%	97.8%	97.5%	97.6%	97.1%	97.5%	97.8%
75	97.9%	98.0%	97.9%	97.8%	97.7%	97.8%	97.8%	97.6%	97.5%	97.7%
80	98.2%	98.4%	98.0%	98.0%	98.0%	97.6%	97.8%	97.5%	97.8%	97.7%
85	98.2%	98.2%	98.3%	97.8%	98.0%	98.1%	98.1%	97.7%	97.9%	97.8%
90	98.3%	98.1%	98.4%	98.0%	98.2%	98.3%	98.2%	98.0%	98.2%	97.7%
95	98.5%	98.4%	98.4%	98.2%	98.3%	98.3%	98.1%	97.9%	98.3%	98.1%
100	98.4%	98.3%	98.3%	98.4%	98.5%	98.4%	98.4%	98.2%	98.1%	98.3%

with Table 3.20). From comparing Table 3.19 with Table 3 in [195], it follows that our approach has a close percentage of solved problems (since the limit t was used for the branch-and-bound algorithm) and essentially outperforms the branch-and-bound algorithm in the running time.

Along with the class of problems when non-availability intervals are on both machines: $w_1 > 0$ and $w_2 > 0$, we tested the classes of problems when either $w_1 = 0$ or $w_2 = 0$. Theoretically, the case with $w_1 = 0$ seems to be harder than the case with $w_2 = 0$ since the non-availability interval on machine M_1 may cause an idle time on machine M_2. So, we tested the above two cases (see Table 3.17 for $w = w_1$, and Table 3.18 for $w = w_2$). From Tables 3.17 and 3.18, it follows that there are computational differences between the cases $w = w_1$ and $w = w_2$ (problems with $w = w_1$ are often harder than problems with $w = w_2$), however, these differences are not so significant.

The hardest problems for our stability analysis were obtained due to the union of the above two difficulties. Table 3.20 presents the computational results for the problems with both equalities $w = w_1$ and $p_{i2} = 2p_{i1}$. The worst results were obtained for series with a small number of jobs ($n = 5, n =$

Table 3.19: *Percentage of solved instances with* $p_{i2} = 2p_{i1}$, $w_1 > 0$ *and* $w_2 > 0$

Number of jobs	Number of non-availability intervals									
	1	2	3	4	5	6	7	8	9	10
5	64.5%	50.6%	44.4%	42.1%	40.0%	38.2%	37.7%	37.8%	36.5%	36.0%
10	66.6%	54.2%	47.4%	42.5%	40.3%	36.8%	35.6%	34.6%	33.9%	33.1%
15	68.0%	55.0%	49.2%	45.0%	42.2%	39.1%	37.5%	35.9%	35.6%	34.1%
20	69.4%	56.7%	50.8%	46.7%	43.9%	41.6%	39.9%	38.6%	36.6%	35.2%
25	68.4%	58.2%	52.2%	47.8%	45.8%	42.2%	40.2%	39.0%	37.4%	37.5%
30	68.8%	58.5%	52.7%	48.9%	45.4%	44.2%	42.4%	40.9%	39.8%	38.0%
35	68.6%	58.7%	53.4%	50.0%	46.2%	44.9%	43.0%	41.7%	40.3%	39.4%
40	69.7%	58.2%	52.6%	49.1%	47.2%	45.1%	43.0%	41.1%	41.8%	39.6%
45	69.9%	59.2%	53.2%	50.0%	47.6%	44.9%	43.7%	42.3%	41.2%	41.1%
50	70.3%	60.1%	54.5%	49.5%	47.6%	47.0%	44.5%	44.8%	43.3%	41.0%
55	68.9%	59.1%	54.1%	50.5%	48.0%	46.5%	46.8%	43.6%	42.1%	42.1%
60	70.0%	59.5%	53.8%	50.6%	49.2%	47.4%	46.0%	43.8%	43.7%	42.2%
65	70.4%	60.0%	53.7%	51.5%	49.1%	47.7%	46.5%	44.8%	43.4%	42.7%
70	70.1%	60.4%	54.7%	50.9%	49.5%	48.2%	47.7%	45.5%	43.6%	42.6%
75	68.8%	60.7%	54.9%	51.5%	49.5%	48.4%	46.1%	45.5%	44.3%	43.7%
80	70.0%	60.3%	54.2%	51.5%	50.3%	48.3%	47.0%	45.0%	44.4%	44.6%
85	70.8%	60.5%	55.1%	52.1%	50.4%	48.2%	46.6%	45.5%	44.7%	45.0%
90	70.2%	60.8%	55.4%	52.3%	50.9%	48.0%	47.2%	46.7%	45.5%	43.9%
95	71.2%	60.0%	55.5%	52.5%	51.0%	47.7%	47.5%	46.5%	45.9%	44.6%
100	70.4%	60.8%	57.2%	53.1%	50.8%	48.6%	47.1%	46.9%	45.9%	45.7%

$10, n = 15$ and $n = 20$) and a large number of non-availability intervals (see right-upper corner of Table 3.20). Thus, the sufficient conditions derived in this section are disappointing for the problems with $p_{i2} = 2p_{i1}$ and $w = w_1 > n$. Such a class of problems needs to be studied in more detail.

Note that the easiest problems for a stability analysis based on the above computational scheme were obtained for the union of equality $w = w_2$ and equality $p_{i2} = 2p_{i1}$: All the 10,000,000 instances of such problems were optimally solved in our experiments.

Problems of Large Size and Resume

Due to the small running times, we were able to investigate much larger instances than those considered in [195]. We tested problems $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$) with

- $1000, 2000, \ldots, 10000$ jobs with integer processing times uniformly distributed in the range $[1, 1000]$ and with
- $10, 100, 500, 1000$ non-availability intervals with integer lengths uniformly distributed in the range $[1, 1000]$ on both machines.

Table 3.20: *Percentage of solved instances with* $p_{i2} = 2p_{i1}$ *and* $w = w_1$

Number of jobs	Number of non-availability intervals									
	1	2	3	4	5	6	7	8	9	10
5	27.4%	16.2%	9.5%	5.0%	3.0%	1.4%	0.7%	0.5%	0.2%	0.1%
10	33.0%	26.1%	20.8%	15.3%	11.4%	7.5%	5.3%	3.0%	1.9%	0.8%
15	35.0%	31.6%	27.8%	23.3%	19.6%	16.1%	13.8%	10.7%	8.2%	6.1%
20	36.2%	34.0%	29.9%	27.1%	26.6%	23.0%	20.3%	17.6%	15.2%	13.3%
25	36.9%	35.4%	32.5%	30.3%	28.2%	25.9%	24.2%	21.9%	19.6%	17.9%
30	38.4%	35.8%	33.7%	32.9%	30.2%	29.0%	26.6%	25.6%	23.8%	22.6%
35	38.4%	35.9%	35.6%	32.9%	31.9%	30.7%	30.0%	27.1%	26.2%	24.8%
40	38.9%	37.5%	35.6%	33.9%	33.0%	31.8%	30.7%	29.7%	28.7%	26.8%
45	38.9%	38.0%	36.1%	36.1%	34.0%	32.9%	32.9%	30.5%	29.9%	27.5%
50	39.5%	38.4%	36.5%	36.3%	35.1%	34.4%	33.1%	31.8%	30.7%	30.1%
55	39.5%	38.7%	37.5%	36.8%	36.0%	34.7%	33.8%	33.2%	33.1%	30.7%
60	40.7%	38.6%	37.9%	37.3%	36.8%	34.8%	33.8%	34.1%	32.8%	31.7%
65	40.7%	39.6%	38.8%	37.4%	36.9%	36.6%	35.7%	34.2%	33.3%	32.6%
70	39.8%	40.6%	39.2%	38.0%	37.7%	36.5%	36.3%	34.4%	34.6%	33.7%
75	41.2%	39.7%	39.3%	38.0%	38.4%	36.7%	36.6%	35.6%	35.0%	34.2%
80	40.6%	39.9%	40.4%	39.2%	38.7%	37.5%	36.6%	36.1%	36.1%	35.9%
85	41.5%	40.1%	40.2%	39.4%	38.0%	38.7%	37.4%	36.6%	35.5%	34.6%
90	41.2%	39.6%	40.4%	39.5%	39.4%	38.4%	37.2%	37.4%	37.1%	37.0%
95	41.6%	40.6%	40.2%	40.2%	38.9%	38.8%	37.9%	38.3%	35.8%	36.6%
100	41.3%	40.6%	40.0%	39.3%	39.3%	39.3%	38.2%	38.5%	37.8%	36.5%

Table 3.21: *Average running time and percentage of solved instances with* $w_1 > 0$ *and* $w_2 > 0$

Number of jobs	Number of non-availability intervals			
	10	100	500	1000
1000	99.9% (0.008 s)	100% (0.008 s)	99.9% (0.010 s)	99.8% (0.027 s)
2000	100% (0.029 s)	100% (0.029 s)	100% (0.031 s)	100% (0.037 s)
3000	100% (0.065 s)	100% (0.065 s)	99.8% (0.101 s)	100% (0.073 s)
4000	100% (0.116 s)	100% (0.116 s)	100% (0.118 s)	100% (0.123 s)
5000	100% (0.180 s)	100% (0.180 s)	100% (0.182 s)	100% (0.188 s)
6000	100% (0.259 s)	100% (0.259 s)	100% (0.261 s)	100% (0.266 s)
7000	100% (0.352 s)	100% (0.352 s)	100% (0.354 s)	100% (0.359 s)
8000	100% (0.459 s)	100% (0.459 s)	100% (0.461 s)	100% (0.466 s)
9000	100% (0.580 s)	100% (0.580 s)	100% (0.582 s)	100% (0.588 s)
10000	100% (0.715 s)	100% (0.716 s)	100% (0.717 s)	100% (0.723 s)

The computational results for the latter problems are given in Table 3.21 which is analogue to Table 3.16.

Tables 3.21 – 3.25 give the percentage of (optimally) solved instances due to a stability analysis based on the above computational scheme and the

Table 3.22: *Average running time and percentage of solved instances with* $w = w_2$

Number of jobs	Number of non-availability intervals			
	10	100	500	1000
1000	100% (0.008 s)	99.9% (0.008 s)	99.3% (0.020 s)	99.8% (0.073 s)
2000	100% (0.029 s)	100% (0.029 s)	100% (0.032 s)	100% (0.041 s)
3000	99.9% (0.067 s)	100% (0.066 s)	100% (0.068 s)	100% (0.077 s)
4000	100% (0.116 s)	100% (0.116 s)	100% (0.119 s)	100% (0.127 s)
5000	100% (0.180 s)	99.9% (0.187 s)	100% (0.183 s)	100% (0.191 s)
6000	100% (0.259 s)	100% (0.259 s)	100% (0.262 s)	100% (0.270 s)
7000	100% (0.352 s)	100% (0.352 s)	100% (0.354 s)	100% (0.363 s)
8000	100% (0.459 s)	100% (0.459 s)	100% (0.461 s)	100% (0.470 s)
9000	100% (0.580 s)	100% (0.580 s)	100% (0.583 s)	100% (0.591 s)
10000	100% (0.716 s)	100% (0.716 s)	100% (0.719 s)	100% (0.727 s)

Table 3.23: *Average running time and percentage of solved instances with* $w = w_1$

Number of jobs	Number of non-availability intervals			
	10	100	500	1000
1000	99.8% (0.008 s)	99.9% (0.008 s)	99.7% (0.017 s)	100% (0.019 s)
2000	99.9% (0.030 s)	99.8% (0.031 s)	99.8% (0.041 s)	100% (0.041 s)
3000	99.9% (0.070 s)	100% (0.065 s)	100% (0.068 s)	100% (0.077 s)
4000	100% (0.116 s)	100% (0.116 s)	100% (0.118 s)	100% (0.127 s)
5000	100% (0.180 s)	100% (0.180 s)	100% (0.183 s)	100% (0.191 s)
6000	100% (0.259 s)	100% (0.259 s)	100% (0.262 s)	100% (0.270 s)
7000	100% (0.351 s)	100% (0.352 s)	100% (0.354 s)	100% (0.363 s)
8000	100% (0.459 s)	100% (0.459 s)	100% (0.461 s)	100% (0.470 s)
9000	100% (0.580 s)	100% (0.580 s)	100% (0.583 s)	100% (0.591 s)
10000	100% (0.715 s)	100% (0.716 s)	100% (0.719 s)	100% (0.727 s)

average running time in seconds for each series of instances (in parentheses). Comparing Table 3.16 with Table 3.21 shows that increasing simultaneously both numbers n and w increases the number of solved instances. The same issue follows from comparing Table 3.17 with Table 3.4, Table 3.18 with Table 3.4, Table 3.19 with Table 3.24, and Table 3.20 with Table 3.25.

Of course, the running time increases with increasing the product nw. Fortunately, the running time for the stability analysis of the algorithm given on page 248 remains rather small even for large problem sizes. Moreover, we can conclude that the order of the considered cases (classes) of problems with respect to an increase of their complexity in our experiments was as follows:

$$[(p_{i2} = 2p_{i1})\&(w = w_2)] \rightarrow [(w_1 > 0)\&(w_2 > 0)] \rightarrow [(w = w_2)] \rightarrow [(w = w_1)]$$

$$\rightarrow [(p_{i2} = 2p_{i1})\&(w_1 > 0)\&(w_2 > 0)] \rightarrow [(p_{i2} = 2p_{i1})\&(w = w_1)].$$

Table 3.24: *Average running time and percentage of solved instances with* $p_{i2} = 2p_{i1}$, $w_1 > 0$ *and* $w_2 > 0$

Number of jobs	Number of non-availability intervals			
	10	100	500	1000
1000	80.5% (0.032 s)	75.9% (0.036 s)	60.3% (0.175 s)	48.9% (1.329 s)
2000	95.4% (0.074 s)	92.3% (0.084 s)	85.1% (0.155 s)	73.5% (0.802 s)
3000	99.2% (0.137 s)	97.6% (0.147 s)	90.4% (0.233 s)	85.4% (0.602 s)
4000	99.3% (0.238 s)	98.9% (0.245 s)	93.4% (0.338 s)	88.0% (0.696 s)
5000	99.9% (0.359 s)	99.5% (0.368 s)	96.2% (0.454 s)	91.7% (0.752 s)
6000	100% (0.514 s)	99.6% (0.526 s)	95.6% (0.664 s)	91.3% (0.984 s)
7000	99.9% (0.703 s)	99.3% (0.728 s)	96.6% (0.853 s)	92.1% (1.221 s)
8000	100% (0.912 s)	99.6% (0.934 s)	97.1% (1.081 s)	93.4% (1.444 s)
9000	99.9% (1.168 s)	99.3% (1.210 s)	98.3% (1.286 s)	93.9% (1.740 s)
10000	100% (1.447 s)	99.7% (1.472 s)	97.4% (1.678 s)	93.0% (2.209 s)

Table 3.25: *Average running time and percentage of solved instances with* $p_{i2} = 2p_{i1}$ *and* $w = w_1$

Number of jobs	Number of non-availability intervals			
	10	100	500	1000
1000	79.4% (0.032 s)	69.6% (0.042 s)	37.4% (0.457 s)	0.3% (4.968 s)
2000	95.0% (0.075 s)	89.7% (0.093 s)	69.5% (0.352 s)	46.6% (2.856 s)
3000	99.1% (0.136 s)	96.2% (0.158 s)	83.5% (0.360 s)	66.1% (2.054 s)
4000	99.4% (0.237 s)	98.6% (0.248 s)	86.5% (0.498 s)	72.8% (1.937 s)
5000	99.7% (0.363 s)	98.3% (0.393 s)	89.0% (0.661 s)	79.6% (1.797 s)
6000	99.8% (0.520 s)	97.8% (0.581 s)	90.8% (0.854 s)	83.2% (1.858 s)
7000	99.9% (0.703 s)	98.1% (0.778 s)	92.7% (1.049 s)	83.0% (2.246 s)
8000	99.8% (0.923 s)	99.3% (0.950 s)	92.8% (1.350 s)	87.8% (2.181 s)
9000	100% (1.161 s)	99.1% (1.224 s)	93.5% (1.653 s)	89.0% (2.469 s)
10000	99.9% (1.455 s)	99.1% (1.524 s)	94.5% (1.956 s)	89.3% (2.899 s)

The hardest class of large problems was that with $p_{i2} = 2p_{i1}$ and $w = w_1$, and the easiest class of large problems was that with $p_{i2} = 2p_{i1}$ and $w = w_2$. All problems from the latter class were optimally solved (the table for this class is omitted). Note that the above order of problem classes had exceptions in our experiments, and it was more expressive for small problems. Within a problem class, the complexity of the instances usually increased with decreasing the difference $n - w$.

In this section, sufficient conditions were proven for a Johnson permutation to be optimal in the case of given non-availability intervals on machines M_1 and M_2 in the two-machine flow shop problem. Due to Theorem 3.25 and Lemmas 3.6 and 3.7, these conditions may be tested in polynomial time in the number n of jobs and the number w of non-availability intervals. In Chapter 2, the notion of the *relative stability radius* was used for a job shop problem. Such a relative stability radius may be used instead of the stability polytope defined in this section. It should be noted that the stability analysis may be used also for other scheduling problems with limited machine availability if an optimal schedule for the corresponding *pure setting* of the problem (i.e., when all machines are continuously available during the whole planning horizon) may be constructed by applying a priority rule to the jobs such as SPT, LPT (longest processing time) and so on. The above computational results show that our stability analysis is efficient for small problems ($n \leq 100$ and $w \leq 10$) and especially for large problems ($1000 \leq n \leq 10000$ and $10 \leq w \leq 1000$). For most classes of the randomly generated problems, only a few instances were not optimally solved within a few seconds of running time. The only exception are problems in which the processing times on machine M_1 are a double of the processing times on machine M_2, the number of jobs is less than the number of non-availability intervals, and machine M_2 is continuously available during the planning horizon. For such a type of problems $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$, other sufficient conditions have to be derived. One can use the above results for some type of on-line scheduling problems when there is no prior information about the exact location of the non-availability intervals on the time axis but the values d_{ij} or the values $\delta_j,\ J_i \in J,\ j = \{1, 2\}$, are known before scheduling. In the next section, we show how to extend the results of this section to a two-machine job shop problem.

3.5. Job Shop with Limited Machine Availability

This section deals with the problem of minimizing the length of a schedule (makespan) for processing n jobs on two machines with w given non-availability intervals. The processing of each job has no more than two stages (i.e., there are at most two operations per job), and the routes through the machines may differ from job to job (it is a job shop). If there is no non-availability interval ($w = 0$), this problem may be polynomially solved using a Jackson pair of job permutations (see Section 3.3), otherwise it is binary NP-hard even if there is only one non-availability interval ($w = 1$), and all

jobs have the same machine route, i.e., for a flow shop which is a special case of a job shop (see Section 3.4). The latter problem becomes unary NP-hard if the number of non-availability intervals may be arbitrarily large. In practice, the limited machine availability may be caused, e.g., by unfinished jobs from the previous schedule, machine breakdowns and machine maintenance, appearance of an unexpected job with high priority or with a close due date.

Next, we find some sufficient conditions when a Jackson pair of permutations remains optimal for the two-machine job shop problem with $w > 1$ given non-availability intervals. Extensive computational studies show the effectiveness (in the number of problems solved) and the efficiency (in computational time) of these sufficient conditions for randomly generated instances with $n \leq 10000$ jobs and $w \leq 1000$ non-availability intervals.

Definitions and Notations

Let a set of jobs $J = \{J_1, J_2, \ldots, J_n\}$ have to be processed in a job shop with the machine set $M = \{M_1, M_2\}$. Each machine $M_j \in M$ can process any job $J_i \in J$ no more than once. Let $J(12) \subseteq J$ be the set of jobs with the machine route (M_1, M_2) (job $J_i \in J(12)$ has to be processed first on machine $M_1 \in M$ and then on machine $M_2 \in M$). Let $J(21) \subseteq J$ be the set of jobs with the opposite machine route (M_2, M_1), and $J(k) \subseteq J$ be the set of jobs which have to be processed on only one machine $M_k \in M$. Thus, we have $J = J(1) \cup J(2) \cup J(12) \cup J(21)$, and we set $n_l = |J_l|$, where $l \in \{1, 2, 12, 21\}$.

Let Q_{ir} denote the operation of job $J_i \in J$ on machine $M_{i_r} \in M$ at stage $r \in \{1, 2\}$ of the machine route (M_{i_1}, M_{i_2}), if $J_i \in J(12) \cup J(21)$, or of the machine route (M_{i_1}), if $J_i \in J(1) \cup J(2)$ and $r = 1$. The processing time p_{ii_r} of operation Q_{ir} of job $J_i \in J$ on machine $M_{i_r} \in M$ is known before scheduling. All the n jobs are available at time $t = 0$. The criterion $\mathcal{C}_{max}$ under consideration is the minimization of the schedule length (makespan): $\mathcal{C}_{max} = \max_s C_{max}(s) = \max_s\{\max\{C_i(s) | J_i \in J\}\}$, where $C_i(s)$ denotes the completion time of job $J_i \in J$ in schedule s.

It is assumed that all w non-availability intervals are known offline (before scheduling) and the processing of a job is resumable. In other words, if the processing of operation Q_{ir} is preempted at the beginning of the unavailable interval of machine $M_{i_r} \in M$, then it can be resumed at the end of this interval without increasing its processing time p_{ii_r} given before scheduling. (Of course, the total processing interval used for operation Q_{ir} will be increased due to the unavailable interval of machine M_{i_r}.) To indicate this, we use the notation *pmtn* since the latter assumption is close to the allowance

of operation preemption. Using the three-field notation, the problem under consideration is denoted as $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$.

Let (π', π'') be a Jackson pair of permutations (see Section 3.3) of the jobs of set J (here π' is a permutation of the jobs $J(12) \cup J(21) \cup J(1)$ on machine M_1, and π'' is a permutation of the jobs $J(12) \cup J(21) \cup J(2)$ on machine M_2) which is optimal when all machines are continuously available ($w = 0$) during the planning horizon (see Theorem 3.21 on page 228). The main aim of this section is to study the question whether this pair of permutations remains optimal for the case of $w \geq 1$ non-availability intervals known offline. To this end, we use an approach similar to that proposed in Section 3.4 for the case of a flow shop problem with limited machine availability.

First, we present some properties of an optimal schedule and remind some useful results for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ proven in Section 3.4. Then we prove sufficient conditions for the optimality of a Jackson pair of permutations in the case of $w > 1$ given intervals of machine non-availability. The end of this section contains some computational results for a stability analysis of a Jackson pair of permutations.

We use the notations similar to those used in Section 3.4. Machine $M_j \in M$ is not available for processing any job from set J from the starting point $s(N_{kj})$ until the endpoint $f(N_{kj}) = s(N_{kj}) + h(N_{kj}),\ k \in \{1, 2, \ldots, w_j\}$. The operation Q_{ii_r} started before the time point $s(N_{kj})$ with $j = i_r$ but not finished until the time point $s(N_{kj})$ is suspended during a time period of length $h(N_{kj})$ starting from time point $s(N_{kj})$. Then the processing of operation Q_{ii_r} is resumed from time point $f(N_{kj})$. Since the minimization of the completion time is a regular criterion, we can consider only semiactive schedules (see Definition 1.1 on page 22). Each semiactive schedule is uniquely defined by a permutation of the jobs on machine M_1 and one of the jobs on machine M_2. For a semiactive schedule s, let $c_1(s)$ and $c_2(s)$ denote the completion time of all jobs on machine M_1 and on machine M_2, respectively. The completion time of schedule s (and so the length or the makespan of schedule s) may be defined as follows: $C_{max}(s) = \max\{c_1(s), c_2(s)\}$. We use the following definition of the main machine in schedule s.

Definition 3.6 *Machine $M_j \in M$ is called the main machine in schedule s if under this schedule, the following equality holds: $C_{max}(s) = c_j(s)$.*

If equality $w = 0$ holds, then problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ turns into a classical job shop problem $\mathcal{J}2/pmtn/\mathcal{C}_{max}$ with two machines which is polynomially solvable. Note that the use of operation preemptions cannot decrease the length of an optimal schedule. Thus, if $w = 0$, it is sufficient

to construct an optimal schedule for problem $\mathcal{J}2//\mathcal{C}_{max}$ which is defined by the two permutations (sequences) (π', π''), where π' is the sequence of the jobs $J(1) \cup J(12) \cup J(21)$ on machine M_1, and π'' is the sequence of the jobs $J(2) \cup J(12) \cup J(21)$ on machine M_2 (see Theorem 3.21 on page 228).

We use the notation $s(\pi', \pi'')$ for the semiactive schedule (uniquely) defined by a permutation π' of the jobs on machine M_1 and by a permutation π'' of the jobs on machine M_2. If schedule $s(\pi', \pi'')$ is optimal for problem $\mathcal{J}2//\mathcal{C}_{max}$ (for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$), then the pair of permutations (π', π'') is called optimal for problem $\mathcal{J}2//\mathcal{C}_{max}$ (for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$, respectively). Note that here the pair of permutations (π', π'') is the same for both problems $\mathcal{J}2//\mathcal{C}_{max}$ and $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$, while the schedules $s(\pi', \pi'')$ usually are different (in spite of the same notation $s(\pi', \pi'')$ used).

In schedule $s(\pi', \pi'')$ which is optimal for problem $\mathcal{J}2//\mathcal{C}_{max}$, the jobs of set $J(12) \subset J$ (of set $J(21) \subset J$) are processed with respect to a Johnson permutation (see condition (3.2) on page 181 in Section 3.1). Therefore, it is sufficient to look for a solution to problem $\mathcal{J}2//\mathcal{C}_{max}$ using a set of pairs of permutations: $(\pi' = (\pi_{12}, \pi_1, \pi_{21}), \pi'' = (\pi_{21}, \pi_2, \pi_{12}))$, where the permutations π_{12} and π_{21} are defined by Johnson's algorithm given on page 181. Hereafter, job J_i belongs to permutation π_l if and only if $J_i \in J(l)$, $l \in \{1, 2, 12, 21\}$. Moreover, the processing order of the jobs of set $J(1)$ (set $J(2)$) may be arbitrary in such an optimal schedule, therefore, we fix both sequences π_1 and π_2 (e.g., in lexicographical order of the jobs numbers). If in sequences π_{12} and π_{21} the jobs are ordered due to Johnson's algorithm, then we say that this is a Jackson pair (π', π'') of permutations of $(n_{12} + n_{21} + n_1)$ jobs and of $(n_{12} + n_{21} + n_2)$ jobs, respectively (see Theorem 3.21 on page 228 in Section 3.3).

Since preemption of an operation in an optimal semiactive schedule s cannot decrease the value $C_{max}(s)$ of the objective function, an optimal schedule defined by a Jackson pair of permutations for an instance of problem $\mathcal{J}2//\mathcal{C}_{max}$ remains an optimal schedule for the instance of problem $\mathcal{J}2/pmtn/\mathcal{C}_{max}$ with the same input data.

Properties of an Optimal Schedule

Let (π', π'') be a Jackson pair of permutations (which is optimal when both machines are continuously available during the planning horizon, i.e., for problem $\mathcal{J}2//\mathcal{C}_{max}$). Our aim is to answer the question when this pair of permutations remains optimal for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ with

$w \geq 1$ given non-availability intervals and with operation processing times being the same as in problem $\mathcal{J}2//\mathcal{C}_{max}$. In the proofs of the sufficient conditions, we shall use the following obvious claim.

Lemma 3.9 *If the main machine M_j in schedule $s(\pi', \pi'')$ is completely filled in the closed interval $[0, c_j(s)]$ by operations on machine M_j and by non-availability intervals of this machine (in this case, we say that machine M_j works without idles), then the pair of permutations (π', π'') is optimal for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$.*

It should be noted that in the above schedule $s(\pi', \pi'')$, the pair of permutations (π', π'') may be not only a Jackson pair of permutations for problem $\mathcal{J}2//\mathcal{C}_{max}$. Nevertheless, if the conditions of Lemma 3.9 hold for some pair of permutations (π', π''), then schedule $s(\pi', \pi'')$ would be optimal for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ without fail. So, the conditions of Lemma 3.9 are sufficient for the optimality of schedule $s(\pi', \pi'')$ for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ regardless whether the permutation pair (π', π'') is a Jackson pair of permutations or not.

Let us assume that for a Jackson pair of permutations (π', π''), the main machine for schedule $s(\pi', \pi'')$ works with idle. W.l.o.g. we can assume that it is machine M_2 since in a two-machine job shop problem, there is a machine symmetry. The main machine M_2 processes the jobs of set $J(2) \cup J(12) \cup J(21)$ with respect to sequence $\pi'' = (\pi_{21}, \pi_2, \pi_{12})$. It is easy to see that all jobs from the sequences π_{21} and π_2 have to be processed on machine M_2 without idles in any semiactive schedule. Therefore, an idle time on machine M_2 may arise only due to the operations of the jobs from set $J(12)$. Hence, the jobs of the set $J(12)$ define the makespan for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$.

Therefore, in what follows, we can mainly concentrate on the processing of only this set $J(12)$ of jobs. Note that the set $J(12)$ of jobs forms a special case of a flow shop problem generated by the original job shop problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ under consideration. Namely, we shall consider the following problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ (we call it problem A) of processing n_{12} jobs of set $J(12)$ with the same machine route (M_1, M_2). Problem A has the same non-availability intervals of machine M_1 and includes all the non-availability intervals of machine M_2 as the original job shop problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ has. Furthermore, problem A has one additional non-availability interval on machine M_2 as follows: $[0, c_2(\pi_{21}, \pi_2))$.

Thus, the job shop problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ under consideration may be reduced to a flow shop problem A in the sense as problem $\mathcal{J}2//\mathcal{C}_{max}$

was reduced to problem $\mathcal{F}2//\mathcal{C}_{max}$ in [168]. Therefore, a lot of results obtained for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ so far may be transformed to those for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$. Next, we demonstrate how to use the results for the flow shop problem presented in Section 3.4 for solving problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$. If the jobs of set $J(12)$ are processed on the machines M_1 and M_2 with respect to the same sequence π_{12} which is a Johnson permutation of these jobs, then the schedule defined by permutation π_{12} is optimal for problem $\mathcal{F}2//\mathcal{C}_{max}$. The next question is whether permutation π_{12} is optimal for the new problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ or not.

Section 3.4 contains sufficient conditions for a Johnson permutation π_{12} for problem $\mathcal{F}2//\mathcal{C}_{max}$ to be an optimal permutation for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ as well. More precisely, to answer the above question, we can test sufficient conditions for the stability of an optimal permutation $\sigma = \pi_v$ given in Theorem 3.26, Corollary 3.2 and Corollary 3.3 (see Section 3.4, pages 245, 246 and 246, respectively). Let at least one of these sufficient conditions hold for permutation $\sigma = \pi_{12}$ of the jobs from set $J(12)$ for the flow shop problem A. Then this set $J(12)$ of jobs defines the minimum makespan value for the original job shop problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$. Hence, due to these conditions, the pair of permutations $s(\pi', \pi'')$ remains optimal for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$. Next, we prove that in the semiactive schedule $s(\pi', \pi'')$, at least one machine has to work without idles, i.e., idle times (not caused by non-availability intervals) are only possible on one of the two machines of set M in the semiactive schedule $s(\pi', \pi'')$.

W.l.o.g., we can assume that machine M_2 works with idles in the schedule $s(\pi', \pi'')$, and let $t = s(N_{1,2})$ be the starting point of the first such idle time of machine M_2, i.e., the first idle time (not caused by non-availability intervals) in the order from left to right at the numerical axis. Obviously, such an idle time of machine M_2 is only possible if some job J_i from set $J(12)$ is still in process on machine M_1, while this job J_i could be processed on machine M_2 at the same time (i.e., machine M_2 has already been completed all jobs of sets $J(21)$ and $J(2)$, and machine M_2 is available at this time). Since schedule $s(\pi', \pi'')$ is defined by the pair of permutations (π', π'') where $\pi' = (\pi_{12}, \pi_1, \pi_{21})$, the following inequality must hold:

$$c_2(\pi_{21}, \pi_2) < c_1(\pi_{12}). \tag{3.44}$$

Analogously, such an idle time of machine M_1 may arise only due to some jobs from set $J(21)$ being still in process by machine M_2, while they could be processed by machine M_1 (i.e., machine M_1 has already completed all jobs of the sets $J(12)$ and $J(1)$, and machine M_1 is available).

Since inequality (3.44) holds, machine M_1 can freely process jobs of set $J(21)$, and therefore, machine M_1 has to work without idles. So, the following claim has been proved.

Lemma 3.10 *Under schedule $s(\pi', \pi'')$, at least one of the two machines in set M works without idles.*

Lemma 3.10 implies the following claim.

Corollary 3.4 *If under schedule $s(\pi', \pi'')$, equality $c_1(s) = c_2(s)$ holds, then the pair of permutations (π', π'') is optimal for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$.*

Indeed, since at least one machine works without idles (due to Lemma 3.10), then due to equality $c_1(s) = c_2(s)$, it is impossible to decrease the makespan $C_{max}(s(\pi', \pi''))$.

Sufficient Conditions for the Optimality of a Permutation Pair

We say that problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ *corresponds* to problem $\mathcal{J}2//\mathcal{C}_{max}$ (and vice versa), if equality $w = 0$ holds for the latter problem, inequality $w > 0$ holds for the former problem, and all the other parameters and conditions are the same for both problems. We want to answer the following question. When would the pair of Jackson permutations (π', π'') constructed for problem $\mathcal{J}2//\mathcal{C}_{max}$ be optimal for the corresponding problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$?

Let (π', π'') be a Jackson pair of permutations constructed for problem $\mathcal{J}2//\mathcal{C}_{max}$. W.l.o.g., we assume that machine M_2 is the main machine in schedule $s(\pi', \pi'')$ for problem $\mathcal{J}2//\mathcal{C}_{max}$. In the following claim, the same notation $s(\pi', \pi'')$ is used twice but in a different sense: It is a schedule for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$, and it is a schedule for problem $\mathcal{J}2//\mathcal{C}_{max}$.

Theorem 3.27 *Let (π', π'') be a Jackson pair of permutations constructed for problem $\mathcal{J}2//\mathcal{C}_{max}$. The schedule $s(\pi', \pi'')$ is optimal for the corresponding problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ if for the corresponding problem $\mathcal{J}2//\mathcal{C}_{max}$, the main machine M_2 works without idles in schedule $s(\pi', \pi'')$, and the following two conditions hold:*

$$\sum_{k=1}^{w_1} h(N_{k1}) \leq c_2(\pi'') - c_1(\pi'), \tag{3.45}$$

$$s(N_{1,1}) \geq \sum_{J_i \in J(12)} p_{i1}. \tag{3.46}$$

PROOF. We shall show that, due to the above conditions, machine M_2 remains the main machine for the corresponding problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ and works without idles. As a result, the conditions of Lemma 3.9 will hold, and so schedule $s(\pi',\pi'')$ will be optimal for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ as well.

Indeed, from inequality (3.46), we obtain that the non-availability intervals of machine M_1 cannot increase the idle time of machine M_2. Therefore, machine M_2 is completely filled in the closed interval $[0, c'_2(\pi'')]$ by processing times of jobs from permutation π'' and by non-availability intervals of machine M_2. Hence, machine M_2 works without idles, and the following equality holds:

$$c'_2(\pi'') = c_2(\pi'') + \sum_{k=1}^{w_2} h(N_{k2}).$$

Next, we prove that machine M_2 remains the main machine for the corresponding problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$. Note, that the non-availability intervals of machine M_2 can increase the idle time of machine M_1. This additional idle time (IT) of machine M_1 (if it exists) is not greater than the sum of the lengths of the non-availability intervals of machine M_2:

$$IT \leq \sum_{i=1}^{w_2} h(N_{i2}).$$

Thus, the completion times of the jobs in permutation π' on machine M_1 cannot be increased by more than the sum of the lengths of the non-availability intervals:

$$c'_1(\pi') \leq c_1(\pi') + IT + \sum_{l=1}^{w_1} h(N_{l1}) \leq c_1(\pi') + \sum_{l=1}^{w_1} h(N_{l1}) + \sum_{i=1}^{w_2} h(N_{i2}).$$

We obtain

$$c'_2(\pi'') = c_2(\pi'') + \sum_{k=1}^{w_2} h(N_{k2}) = c_2(\pi'') - c_1(\pi') + c_1(\pi') + \sum_{k=1}^{w_2} h(N_{k2})$$

$$\geq \sum_{l=1}^{w_1} h(N_{l1}) + c_1(\pi') + \sum_{k=1}^{w_2} h(N_{k2}) \geq c'_1(\pi').$$

Therefore, machine M_2 remains the main machine for the corresponding problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$.

Thus, the conditions of Lemma 3.9 hold, and therefore, schedule $s(\pi',\pi'')$ is optimal for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$. This completes the proof.

$\diamond$

Next, we show that all the above conditions of Theorem 3.27 are essential. In other words, if at least one condition of Theorem 3.27 does not hold, then the Jackson pair of permutations (π', π'') mentioned in Theorem 3.27 may be not optimal for the corresponding problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$. We consider the following three examples (one example for each condition of Theorem 3.27 which is violated).

Example 3.8 *We consider problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ with the processing times of the four jobs given in Table 3.26, where the subset of jobs $J(kl)$ is indicated in the first column such that $J_i \in J(kl)$ and $\{k,l\} = \{1,2\}$. One non-availability interval $(5,7)$ is given on machine M_2.*

Table 3.26: *Processing times for Example 3.8*

Job J_i	i	p_{i1}	p_{i2}
$J(12)$	1	5	4
	2	4	3
$J(21)$	3	1	2
	4	1	1

The following condition of Theorem 3.27 does not hold: The main machine M_2 works with idle for the corresponding problem $\mathcal{J}2//\mathcal{C}_{max}$. A Jackson pair of permutations for the corresponding problem $\mathcal{J}2//\mathcal{C}_{max}$ is $((J_1,J_2,J_4,J_3),\ (J_4,J_3,J_1,J_2))$, and machine M_2 works with idle from time point 3 to time point 5. It is easy to see that for the original problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$, the pair of permutations $((J_1,J_2,J_4,J_3),\ (J_4,J_3,J_1,J_2))$ is not optimal. Namely, for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$, an optimal pair of permutations is $((J_2,J_1,J_4,J_3),\ (J_4,J_3,J_2,J_1))$ (by the way, it is not a Jackson pair of permutations for the corresponding problem $\mathcal{J}2//\mathcal{C}_{max}$).

Let now inequality (3.45) of Theorem 3.27 be violated, i.e., let the sum of the non-availability intervals on machine M_2 be larger than the difference $c_2(\pi'') - c_1(\pi')$. For such a case, an example exists (see Example 3.9) for which there is no optimal schedule defined by a Jackson pair of permutations.

Example 3.9 *Let the job processing times for this example be given in Table 3.27, and let one non-availability interval (5, 7) be given on machine M_1. A Jackson pair of permutations for the corresponding problem $\mathcal{J}2//\mathcal{C}_{max}$ is $((J_1,J_2,J_3,J_4),\ (J_3,J_4,J_1,J_2))$. Here, we have $c_1(\pi') = 12$, $c_2(\pi'') = 13$ and*

$$c_2(\pi'') - c_1(\pi') = 1 < \sum_{i=1}^{w_1=1} h(N_{i1}) = h(N_{1,1}) = 2.$$

Table 3.27: *Processing times for Example 3.9*

Job J_i	i	p_{i1}	p_{i2}
$J(12)$	1	2	2
	2	1	2
$J(21)$	3	4	5
	4	3	4

It is easy to see that for the original problem $\mathcal{J}2$,$NC^{off}/pmtn/\mathcal{C}_{max}$, the pair of permutations $((J_1, J_2, J_3, J_4), (J_3, J_4, J_1, J_2))$ is not optimal. (For problem $\mathcal{J}2$,$NC^{off}/pmtn/\mathcal{C}_{max}$, an optimal pair of permutations is $((J_1, J_2, J_4, J_3), (J_4, J_3, J_1, J_2))$ which is certainly not a Jackson pair of permutations.)

Example 3.10 *Let now inequality (3.46) of Theorem 3.27 be violated, i.e., the starting point of the first non-availability interval of machine M_1 is less than the endpoint of the processing of all jobs of set $J(12)$ on machine M_1.*

Let the processing times of the jobs be given in Table 3.28. We consider two non-availability intervals: The first non-availability interval (2, 3) on machine M_1, and the second non-availability interval (4, 7) on machine M_2.

Table 3.28: *Processing times for Example 3.10*

Job J_i	i	p_{i1}	p_{i2}
J_{12}	1	3	2
	2	2	1
J_{21}	3	3	1
	4	2	1

An optimal Jackson pair of permutations for the corresponding problem $\mathcal{J}2//\mathcal{C}_{max}$ is $((J_1, J_2, J_4, J_3), (J_4, J_3, J_1, J_2))$. Here, we have $c_1(\pi_{12}) = 5$ while $s(N_{1,1}) = 2$, and so the beginning of the first non-availability interval of machine M_1 is less than the endpoint of the processing of all jobs of set $J(12)$ on machine M_1. For the original problem $\mathcal{J}2$,$NC^{off}/pmtn/\mathcal{C}_{max}$, the pair of permutations $((J_1, J_2, J_4, J_3), (J_4, J_3, J_1, J_2))$ is not optimal.

For problem $\mathcal{J}2$,$NC^{off}/pmtn/\mathcal{C}_{max}$, an optimal pair of permutations is $((J_2, J_1, J_4, J_3), (J_4, J_3, J_2, J_1))$ which is not a Jackson pair of permutations.

In the following claim, the same notation $s(\pi', \pi'')$ is used twice but in

a different sense: First, it is a schedule for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$, and then it is a schedule for problem $\mathcal{J}2//\mathcal{C}_{max}$.

Theorem 3.28 *Let (π', π'') be a Jackson pair of permutations for problem $\mathcal{J}2//\mathcal{C}_{max}$. Then schedule $s(\pi', \pi'')$ is optimal for the corresponding problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ if for the corresponding problem $\mathcal{J}2//\mathcal{C}_{max}$, the main machine M_2 works without idles in schedule $s(\pi', \pi'')$, and the following three conditions hold:*

$$\sum_{k=1}^{w_1} h(N_{k1}) \geq \sum_{k=1}^{w_2} h(N_{k2}), \tag{3.47}$$

$$s(N_{1,2}) \geq \sum_{J_i \in J(12)} p_{i1}, \tag{3.48}$$

$$s(N_{1,1}) \geq \sum_{J_i \in J(21)} p_{i2}. \tag{3.49}$$

PROOF. Next, we show that, if all the above conditions hold in common, then machine M_2 remains the main machine for the corresponding problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ and works without idles. Therefore, the conditions of Lemma 3.9 hold and therefore, schedule $s(\pi', \pi'')$ has to be optimal for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$.

Indeed, inequality (3.48) implies that the non-availability intervals of machine M_1 cannot increase the idle time on machine M_2. Hence, machine M_2 is completely filled in the closed interval $[0, c'_2(\pi'')]$ with processing times of all jobs from permutation π'' and non-availability intervals of this machine. So, machine M_2 works without idles and we obtain equality

$$c'_2(\pi'') = c_2(\pi'') + \sum_{k=1}^{w_2} h(N_{k2}).$$

Next, we show that machine M_2 remains the main machine for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$. Inequality (3.49) implies that the non-availability intervals of machine M_2 cannot increase the idle time on machine M_1. Therefore, the endpoint on machine M_1 cannot be enlarged more than by the sum of the lengths of all non-availability intervals on this machine, i.e., we have

$$c'_1(\pi') \leq c_1(\pi') + \sum_{k=1}^{w_1} h(N_{k1}).$$

We obtain

$$c'_1(\pi') \leq c_1(\pi') + \sum_{k=1}^{w_1} h(N_{k1}) \leq c_2(\pi'') + \sum_{k=1}^{w_2} h(N_{k2}) = c'_2(\pi'').$$

Therefore, machine M_2 is the main machine for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$. Thus, the conditions of Lemma 3.9 hold and hence, schedule $s(\pi', \pi'')$ is optimal for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$. This completes the proof.

$\diamond$

Computational Results

Our computational study of the above sufficient conditions was performed on a huge number of randomly generated problems $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$. For each randomly generated instance $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$, we constructed a Jackson pair of permutations for the corresponding job shop problem $\mathcal{J}2//\mathcal{C}_{max}$ (i.e., that with the same job processing times) and answered the question: Is this pair of permutations optimal for the original problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$? To this end, we tested the sufficient conditions proven in Section 3.4 for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ and those proven in this section for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$. To minimize the running time of the software, these sufficient conditions were tested in an increasing order of their complexity up to the first positive answer (if any) to the above question. More formally, the following algorithm was realized.

Algorithm for $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$

Input: Processing times of the jobs J
and non-availability intervals of the machines M.

Output: An optimal schedule for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$
or a feasible schedule if the algorithm terminates in *Step 6*.

Step 1: Construct a Jackson pair of permutations
(π', π'') for problem $\mathcal{J}2//\mathcal{C}_{max}$
and a Johnson permutation σ for the flow shop problem A.

Step 2: **IF** schedule $s(\sigma) = s(\pi_{12})$ for problem A satisfies
the condition of Corollary 3.2 **THEN GOTO** *Step 7*.

Step 3: **IF** schedule $s(\sigma) = s(\pi_{12})$ for problem A satisfies
the condition of Corollary 3.3 **THEN GOTO** *Step 7*.

Step 4: Construct $\lambda = \min\{\lambda^*, 2^k\}$ Johnson permutations $\sigma_1, \sigma_2, \ldots,$
σ_λ for problem A, and λ Jackson pairs of permutations
$(\pi'_1, \pi''_1), (\pi'_2, \pi''_2), \ldots, (\pi'_\lambda, \pi''_\lambda)$ for problem $\mathcal{J}2//\mathcal{C}_{max}$.

Step 5: **FOR** $i = 1, 2, \ldots, \lambda$ **DO**
BEGIN

Set $\sigma := \sigma_i$.
IF schedule $s(\sigma)$ for problem A satisfies the condition of Corollary 3.2 **THEN GOTO** *Step 7*.
IF schedule $s(\sigma)$ for problem A satisfies the condition of Corollary 3.3 **THEN GOTO** *Step 7*.
IF schedule $s(\pi', \pi'')$ satisfies the condition of Theorem 3.27 **THEN GOTO** *Step 7*.
IF schedule $s(\pi', \pi'')$ satisfies the condition of Theorem 3.28 **THEN GOTO** *Step 7*.
END

Step 6: Optimality of the pairs of permutations (π'_i, π''_i), $i = 1, \ldots, \lambda$, for problem $\mathcal{J}2{,}NC^{off}/pmtn/\mathcal{C}_{max}$ is not proven **STOP**.

Step 7: The pair of permutations (π', π'') is optimal for problem $\mathcal{J}2{,}NC^{off}/pmtn/\mathcal{C}_{max}$ **STOP**.

Steps 4 and 5 of the above algorithm are realized since there may be more than one Jackson pair of permutations (if equalities $p_{ij} = p_{kj}$ hold for different jobs $J_i \in J$ and $J_k \in J$). We tested up to $\min\{\lambda^*, 2^k\}$ pairs of Jackson permutations for each instance. In the experiments, we set $\lambda^* = 1024$. For each combination of n and w, Tables 3.29 – 3.40 present the percentage of instances in the series for which the calculation based on the above algorithm found a Jackson pair of permutations which remains optimal in spite of the given non-availability intervals.

Small and Moderate Problems

First, we tested problems similar to those considered in [195, 45], i.e., with

- $5, 10, 15, \ldots, 100$ jobs having integer processing times uniformly distributed in the range $[1, 1000]$ and with
- $1, 2, 3, \ldots, 10$ non-availability intervals (of both machines) with integer lengths uniformly distributed in the range $[1, 1000]$.

The above algorithm was coded in C++. For the computational experiments, we used an AMD 1200 MHz processor with 1024 MB main memory. For the *small* and *moderate* instances similar to those considered in [195] and in [45], we made 10000 tests in each series (i.e., for each combination of n and

Table 3.29: *Percentage of solved moderate (easy) instances with $w_1 > 0$ and $w_2 > 0$*

$\|J\|$	Number of non-availability intervals									
	1	2	3	4	5	6	7	8	9	10
5	86.2%	85.5%	87.8%	85.9%	85.7%	85.2%	85.0%	83.2%	83.4%	85.8%
10	92.1%	93.5%	93.3%	92.4%	92.8%	92.1%	92.6%	93.2%	93.0%	94.3%
15	95.0%	95.1%	95.7%	95.8%	96.2%	96.5%	95.7%	96.2%	95.2%	95.4%
20	97.3%	97.4%	97.2%	97.9%	97.6%	97.5%	96.7%	97.5%	96.4%	97.4%
25	97.7%	98.2%	98.5%	97.8%	99.1%	97.5%	98.5%	98.7%	98.6%	97.6%
30	99.1%	99.2%	98.9%	99.3%	99.0%	98.9%	99.0%	98.7%	98.5%	99.4%
35	99.3%	99.0%	99.4%	99.3%	98.6%	98.7%	99.6%	99.7%	99.7%	99.5%
40	99.6%	99.1%	99.2%	99.4%	99.0%	99.8%	99.2%	99.5%	99.4%	99.5%
45	99.1%	99.2%	99.3%	99.5%	99.8%	98.8%	99.6%	99.4%	98.7%	99.7%
50	99.3%	99.1%	99.7%	99.2%	99.6%	99.8%	99.9%	100.0%	99.7%	99.6%
55	99.6%	99.7%	99.5%	99.8%	99.8%	99.6%	99.8%	99.5%	99.4%	99.4%
60	99.7%	99.8%	99.9%	99.8%	99.9%	99.9%	99.7%	100.0%	99.8%	99.7%
65	99.6%	99.6%	99.8%	99.7%	99.9%	99.9%	99.7%	99.9%	99.6%	99.6%
70	99.7%	99.8%	99.7%	99.9%	99.6%	99.7%	99.8%	99.6%	100%	100%
75	100%	99.9%	100%	99.9%	99.8%	99.7%	99.6%	99.9%	100%	99.9%
80	99.8%	100%	99.9%	99.7%	99.9%	100.0%	99.9%	99.8%	99.9%	99.8%
85	99.9%	100%	99.9%	100%	99.4%	99.9%	100%	100%	99.8%	99.9%
90	100%	99.9%	99.8%	99.9%	99.9%	99.9%	99.8%	99.9%	100%	99.5%
95	100%	99.9%	99.9%	100.0%	99.9%	99.7%	100%	99.9%	100%	100%
100	99.7%	99.8%	99.8%	99.9%	99.9%	99.9%	100%	100%	99.9%	99.9%

w). For the *large* instances, we made 1000 tests in each series. We answer the question of how many schedules following the Jackson order (constructed for problem $\mathcal{J}2//\mathcal{C}_{max}$) remain optimal in spite of the non-availability intervals of the machines (i.e., for problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$). We tested only the sufficient conditions and our algorithm does not guarantee to find an optimal schedule for some problems $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ (however, our experiments showed that such instances arise rather seldom for most randomly generated problems). We considered 10000 instances in each series of small problems.

Tables 3.29 – 3.34 present the percentage of problem instances which were (optimally) solved due to a stability analysis by the above algorithm. We do not present the running times for the series of small instances, since the running times were very close for different instances and very small: The *maximum* running time of an instance in the series presented in Tables 3.29 – 3.34 was 0.001 seconds.

Along with the case when the non-availability intervals are on both machines: $w_1 > 0$ and $w_2 > 0$, we tested the cases (classes) of problems when either $w_1 = 0$ or $w_2 = 0$.

Table 3.30: *Percentage of solved moderate (hard) instances with $w_1 > 0$ and $w_2 > 0$*

$\|J\|$	Number of non-availability intervals									
	1	2	3	4	5	6	7	8	9	10
5	64.9%	68.2%	69.0%	69.1%	70.0%	70.2%	69.6%	65.7%	67.7%	66.3%
10	81.3%	80.4%	80.8%	81.0%	81.8%	79.0%	80.4%	81.7%	81.5%	82.4%
15	87.9%	88.8%	87.8%	89.0%	88.2%	88.6%	89.1%	89.8%	88.6%	88.8%
20	91.1%	92.1%	92.3%	93.0%	91.0%	91.9%	92.2%	92.1%	90.3%	91.2%
25	95.3%	94.7%	93.9%	94.9%	92.5%	94.1%	95.3%	92.9%	94.0%	93.3%
30	94.8%	94.9%	95.7%	96.3%	96.1%	96.2%	95.2%	95.7%	96.0%	94.9%
35	96.4%	97.0%	97.4%	95.9%	95.9%	96.6%	96.9%	95.3%	95.4%	96.6%
40	96.6%	97.5%	96.6%	96.9%	97.4%	96.1%	95.0%	95.9%	96.5%	97.1%
45	97.8%	97.4%	98.1%	97.5%	96.3%	97.4%	97.4%	97.9%	96.9%	97.2%
50	98.4%	98.7%	97.7%	97.0%	97.7%	97.9%	97.2%	97.9%	97.5%	97.7%
55	97.8%	98.0%	97.6%	97.0%	98.0%	97.8%	97.9%	98.4%	97.6%	97.7%
60	98.1%	97.1%	97.9%	98.2%	98.7%	98.2%	97.7%	98.1%	98.5%	98.3%
65	98.4%	98.3%	98.2%	98.5%	97.7%	99.1%	97.8%	98.7%	98.3%	98.4%
70	98.1%	98.3%	98.9%	98.4%	98.8%	98.7%	97.6%	98.3%	98.0%	98.2%
75	97.5%	98.4%	98.3%	98.5%	98.9%	98.4%	98.7%	98.3%	98.4%	98.6%
80	99.1%	98.3%	98.2%	98.4%	98.3%	98.2%	98.7%	98.6%	99.3%	98.6%
85	98.5%	98.7%	98.7%	98.4%	98.7%	99.0%	98.8%	98.7%	98.5%	98.3%
90	99.0%	98.6%	98.8%	98.9%	98.6%	98.6%	99.0%	98.8%	98.7%	97.9%
95	98.5%	98.8%	99.0%	98.9%	98.8%	99.2%	98.9%	99.2%	98.9%	98.4%
100	98.4%	98.7%	98.5%	98.6%	99.4%	99.1%	98.7%	98.5%	98.4%	98.7%

Large Problems

Due to the small running time, we were able to investigate very large instances (as in Section 3.4). We tested instances of problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ with

- $1000, 2000, \ldots, 10000$ jobs having integer processing times uniformly distributed in the range $[1, 1000]$ and with
- $10, 100, 500, 1000$ non-availability intervals with integer lengths uniformly distributed in the range $[1, 1000]$ on both machines.

The computational results for large instances of problem $\mathcal{J}2,NC^{off}/pmtn/\mathcal{C}_{max}$ are given in Tables 3.36 – 3.40 which are analogous to Tables 3.29 – 3.34.

Tables 3.36 – 3.40 give the percentage of (optimally) solved instances due to a stability analysis based on the above algorithm and the average running time in seconds for each series of instances (in parentheses). The maximum running time for one instance was 0.01000 seconds.

Table 3.31: *Percentage of solved moderate (easy) instances with* $w = w_1$

$\|J\|$	Number of non-availability intervals									
	1	2	3	4	5	6	7	8	9	10
5	89.3%	89.7%	88.9%	90.6%	89.6%	89.5%	89.9%	89.9%	88.9%	90.0%
10	96.5%	96.0%	95.0%	96.8%	96.6%	95.8%	95.8%	96.1%	96.6%	95.3%
15	97.5%	98.7%	97.6%	97.3%	97.5%	98.5%	98.4%	98.8%	98.2%	97.4%
20	98.4%	98.6%	99.0%	98.6%	98.7%	99.1%	98.3%	98.6%	98.2%	98.4%
25	98.8%	99.4%	99.5%	99.5%	98.7%	98.7%	98.7%	99.3%	99.2%	99.3%
30	99.3%	99.6%	99.9%	99.1%	99.1%	99.1%	99.6%	99.3%	99.3%	99.3%
35	99.5%	99.4%	99.2%	99.6%	99.7%	99.8%	99.7%	99.6%	99.5%	99.5%
40	99.3%	99.6%	99.6%	99.8%	99.8%	99.4%	99.6%	99.6%	100%	98.9%
45	99.7%	99.7%	99.7%	99.7%	99.1%	100%	99.7%	99.7%	99.8%	99.8%
50	99.6%	99.9%	99.8%	99.8%	100%	99.8%	99.9%	99.7%	100%	99.8%
55	99.7%	99.7%	99.8%	99.7%	99.9%	99.8%	99.9%	99.7%	99.6%	99.6%
60	99.7%	99.7%	99.9%	99.8%	99.8%	99.8%	99.9%	99.6%	99.7%	99.8%
65	99.8%	99.9%	99.9%	99.9%	99.8%	99.8%	100%	99.9%	100%	99.9%
70	99.7%	99.5%	99.8%	99.7%	100.0%	99.9%	99.9%	99.8%	99.9%	99.9%
75	100%	99.4%	100%	99.9%	99.7%	99.7%	100%	99.8%	99.8%	100%
80	99.9%	99.9%	100%	100%	100%	99.9%	99.8%	99.9%	100%	100%
85	100%	99.9%	99.9%	99.8%	100%	99.8%	100%	99.8%	99.5%	99.9%
90	99.9%	99.7%	99.8%	99.8%	99.8%	99.7%	99.9%	100%	99.9%	100%
95	99.8%	100%	99.7%	99.8%	99.8%	99.9%	100%	100%	99.9%	100%
100	99.9%	100%	99.8%	99.8%	99.8%	99.8%	99.9%	100%	99.8%	100%

In this section, sufficient conditions have been proven for a Jackson pair of permutations to be optimal in the case of w given non-availability intervals of machines M_1 and M_2 in the two-machine job shop. Due to Theorems 3.27 and 3.28, Corollaries 3.2 and 3.3 (see Section 3.4), these conditions may be tested in polynomial time in the number n of jobs and the number w of non-availability intervals. However, there are instances of the problems which cannot be solved exactly using these sufficient conditions. This is not surprising since the problem is NP-hard even for $w = 1$ and a single machine route (see Theorem 3.23 and Theorem 3.24 in Section 3.4 given on pages 236 and 238). However, it is worth noticing that the number of unsolved instances decreases with the increase of the number n of jobs. In addition, within the huge computational study, we did not find a type of combination of numbers n and w and a type of the relation between the lengths of non-availability intervals of machine M_1 and machine M_2 for which our sufficient conditions in general give bad results. Note that such types of bad instances exist for flow shop problems considered in Section 3.4.

The above computational results show that our sufficient conditions are very efficient in computational time and effective (in the number of problems

Table 3.32: *Percentage of solved moderate (hard) instances with* $w = w_1$

$\|J\|$	Number of non-availability intervals									
	1	2	3	4	5	6	7	8	9	10
5	69.3%	65.7%	68.3%	65.6%	69.7%	69.8%	69.0%	70.8%	68.9%	70.5%
10	79.4%	79.2%	79.2%	79.4%	79.6%	79.6%	79.3%	77.9%	79.5%	79.1%
15	84.8%	86.3%	84.8%	85.3%	85.0%	83.6%	86.2%	83.4%	84.5%	85.4%
20	90.6%	88.9%	90.3%	89.4%	91.7%	89.5%	90.6%	88.9%	90.3%	90.3%
25	91.9%	95.1%	93.5%	92.0%	93.2%	93.2%	93.0%	93.4%	92.1%	92.2%
30	95.1%	96.2%	95.4%	95.0%	94.8%	95.8%	94.9%	93.8%	95.5%	95.1%
35	96.2%	96.6%	96.4%	96.3%	95.4%	95.7%	96.2%	96.6%	95.9%	95.6%
40	97.1%	95.9%	96.1%	96.8%	95.7%	97.2%	96.4%	96.2%	96.5%	97.6%
45	96.0%	97.1%	97.4%	97.9%	97.3%	97.6%	97.1%	98.1%	97.3%	96.5%
50	97.0%	97.2%	96.9%	97.4%	98.5%	98.2%	98.6%	97.5%	97.0%	97.0%
55	98.0%	98.1%	97.6%	97.8%	97.9%	98.2%	97.8%	97.6%	97.9%	96.8%
60	97.9%	97.9%	98.1%	97.8%	98.0%	97.7%	97.4%	98.6%	97.7%	97.8%
65	98.7%	98.8%	97.8%	98.4%	98.0%	98.1%	98.1%	98.9%	98.8%	98.5%
70	98.5%	98.2%	97.5%	98.3%	98.8%	98.7%	99.0%	98.7%	98.8%	98.5%
75	98.9%	99.1%	98.2%	98.7%	98.8%	98.6%	98.6%	98.1%	98.8%	98.7%
80	98.9%	98.9%	99.0%	99.3%	98.7%	98.9%	98.4%	98.6%	98.3%	98.3%
85	98.8%	98.9%	98.8%	98.2%	98.2%	98.7%	98.6%	98.6%	98.7%	98.6%
90	99.2%	98.8%	99.2%	99.1%	98.8%	99.1%	98.5%	98.5%	98.9%	98.9%
95	98.9%	99.1%	98.3%	99.2%	98.8%	98.6%	98.6%	99.1%	98.8%	99.0%
100	98.7%	99.2%	99.2%	99.0%	99.0%	98.9%	98.7%	98.9%	98.7%	99.1%

solved) for small problems ($n \leq 100$ and $w \leq 10$) and especially for large problems ($1000 \leq n \leq 10000$ and $10 \leq w \leq 1000$). For most classes of randomly generated problems, only a few instances were not optimally solved within a few seconds of running time.

In Section 3.4, analogous sufficient conditions have been proven and used for the case of a flow shop problem. Such sufficient conditions may also be used for other scheduling problems with limited machine availability if an optimal schedule for the corresponding pure setting of the problem (i.e., when all machines are continuously available during the whole planning horizon) can be constructed by applying a priority rule to the jobs such as SPT, LPT and so on. Moreover, one can use some of the above results for some type of online settings of scheduling problems when there is no prior information about the exact location of the non-availability intervals on the time axis. Another topic for future research may be connected with the use of the above sufficient conditions in the framework of an exact solution algorithm like a branch-and-bound one.

Table 3.33: *Percentage of solved moderate (easy) instances with* $w = w_2$

$\|J\|$	Number of non-availability intervals									
	1	2	3	4	5	6	7	8	9	10
5	83.7%	86.3%	86.7%	87.4%	86.6%	83.9%	86.7%	83.9%	85.0%	86.1%
10	93.2%	93.9%	93.0%	92.8%	91.5%	93.6%	93.5%	91.4%	93.6%	92.8%
15	96.2%	95.3%	96.1%	96.6%	96.7%	95.3%	94.4%	95.3%	95.9%	96.1%
20	97.5%	98.0%	96.6%	97.3%	97.3%	97.9%	98.1%	96.8%	97.3%	97.1%
25	98.8%	98.4%	98.2%	98.9%	98.7%	98.3%	97.6%	98.5%	98.3%	98.4%
30	98.6%	99.2%	99.5%	98.4%	99.2%	98.5%	99.0%	98.6%	99.0%	99.1%
35	99.3%	99.4%	99.2%	99.2%	99.2%	99.4%	99.7%	99.4%	99.5%	99.1%
40	99.5%	99.4%	99.1%	99.5%	99.4%	99.1%	98.9%	99.4%	99.3%	99.5%
45	99.6%	99.6%	99.7%	99.6%	99.5%	99.0%	99.9%	99.7%	99.5%	99.2%
50	99.8%	99.1%	99.6%	99.2%	99.6%	99.7%	99.4%	99.8%	99.3%	99.6%
55	99.8%	99.5%	99.8%	99.9%	99.8%	99.6%	99.7%	99.8%	99.7%	99.5%
60	99.4%	99.6%	100.0%	99.8%	99.7%	99.9%	99.6%	99.6%	99.9%	99.5%
65	99.6%	99.6%	99.8%	99.6%	99.5%	99.8%	99.8%	99.5%	99.7%	99.8%
70	99.6%	99.8%	99.4%	99.9%	99.7%	99.7%	99.9%	99.9%	99.8%	99.9%
75	99.9%	99.9%	99.5%	99.7%	99.7%	99.9%	99.9%	99.9%	99.8%	99.7%
80	99.9%	99.8%	99.7%	99.9%	99.8%	100%	99.6%	99.9%	99.6%	99.9%
85	99.9%	99.9%	99.9%	100.0%	99.7%	99.7%	100%	99.8%	99.8%	99.9%
90	99.9%	99.8%	100%	99.9%	99.5%	99.9%	99.9%	99.8%	99.9%	99.8%
95	99.9%	99.8%	99.8%	99.8%	99.8%	99.8%	99.8%	99.9%	99.8%	100%
100	100%	100%	100%	99.9%	99.9%	99.9%	100%	99.7%	99.9%	99.9%

3.6. Comments and References

Most real-life sequencing and scheduling problems involve some forms of uncertainty. Several approaches complementing one another are available for dealing with sequencing and scheduling under uncertainty. In a stochastic approach, uncertain scheduling parameters (processing times and others) are assumed to be random variables with specific probability distributions (see the second part of monograph [269]). There are two types of stochastic shop scheduling problems traditionally addressed in the OR literature, where one is concerned with a stochastic job and the other is concerned with a stochastic machine. In a stochastic job problem, the job processing time is assumed to be a random variable following a certain (and given) probability distribution. For the objective of stochastically minimizing the makespan (i.e., minimizing the expected schedule length), the flow shop problem was considered in the articles [108, 178, 194], among others. In a stochastic machine problem, the job processing time is fixed, while the job completion time is a random variable due to machine breakdowns or other reasons of machine non-availability. In the articles [274, 275] (in [10, 11, 14], respec-

Table 3.34: *Percentage of solved moderate (hard) instances with* $w = w_2$

$\|J\|$	Number of non-availability intervals									
	1	2	3	4	5	6	7	8	9	10
5	68.0%	68.8%	69.3%	70.8%	70.8%	69.2%	69.1%	71.0%	70.2%	70.5%
10	78.5%	76.9%	80.2%	78.2%	79.1%	78.4%	78.5%	76.6%	77.6%	78.9%
15	84.2%	85.4%	85.8%	85.0%	86.2%	83.0%	86.6%	85.1%	83.7%	84.3%
20	90.5%	90.9%	91.5%	90.3%	89.1%	88.3%	91.2%	90.6%	90.9%	90.4%
25	93.0%	94.0%	93.0%	92.9%	93.1%	93.7%	93.5%	89.7%	93.2%	93.8%
30	94.2%	93.7%	94.7%	94.3%	95.3%	95.4%	94.5%	95.2%	94.9%	94.6%
35	96.6%	96.2%	96.4%	95.7%	96.8%	97.1%	96.0%	96.4%	96.5%	96.6%
40	96.6%	96.7%	96.4%	96.5%	97.6%	95.7%	96.2%	96.4%	97.7%	96.3%
45	96.4%	96.7%	98.3%	96.9%	97.3%	97.0%	96.7%	96.4%	96.4%	97.3%
50	97.1%	96.8%	98.0%	97.8%	97.2%	98.0%	97.7%	97.3%	97.5%	98.2%
55	98.4%	96.5%	98.0%	97.9%	98.1%	97.8%	98.2%	98.4%	97.8%	98.0%
60	97.3%	97.8%	97.1%	98.6%	97.7%	98.6%	97.8%	97.9%	97.8%	98.6%
65	98.1%	98.1%	97.6%	97.9%	98.7%	98.1%	98.3%	98.1%	98.4%	97.1%
70	98.4%	98.1%	98.7%	99.2%	97.9%	99.0%	98.5%	98.2%	98.2%	98.8%
75	97.9%	98.1%	99.2%	98.5%	98.7%	98.5%	99.1%	98.1%	98.5%	98.9%
80	98.8%	98.6%	98.5%	98.9%	98.5%	98.3%	98.3%	98.2%	98.3%	99.2%
85	97.9%	98.7%	99.0%	99.1%	98.7%	98.7%	99.0%	98.0%	98.8%	99.0%
90	98.4%	98.4%	98.1%	98.4%	98.8%	98.5%	99.2%	98.7%	98.7%	98.0%
95	98.8%	99.0%	99.0%	98.4%	99.4%	98.5%	98.7%	99.1%	98.5%	99.5%
100	99.1%	99.1%	98.6%	99.4%	99.0%	98.2%	98.6%	98.9%	98.5%	99.3%

tively), a flow shop problem to stochastically minimize the makespan (the total completion time) was considered. A couple of researchers [10, 14] considered flow shops with machine breakdowns. Article [4] was addressed to the so-called group shop (which is a generalization of the flow shop, the job shop and the open shop) with random processing and release times having known probability distributions. The objective is to find a schedule which stochastically minimizes the total weighted completion time.

Unfortunately, in many real-life situations, one may have no sufficient information to characterize the probability distribution of each random parameter. In such situations, other approaches are needed [24, 283]. For the uncertain problem $\alpha/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\gamma$, there usually does not exist a single schedule (a job permutation for each machine) that remains optimal for all scenarios of set T, where a scenario denotes a possible realization of all the uncertain processing times. So, an additional criterion is often introduced for dealing with the uncertain problem $\alpha/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\gamma$.

In particular, in the approach of seeking a *robust schedule* [19, 88, 187, 189, 205, 293, 371], the decision-maker prefers a schedule that hedges against

Table 3.35: *Average running time and percentage of solved instances with* $w_1 > 0$ *and* $w_2 > 0$

Number of jobs	Number of non-availability intervals			
	10	100	500	1000
1000	100% (0.004 s)	100% (0.004 s)	100% (0.004 s)	100% (0.004 s)
2000	100% (0.016 s)	100% (0.016 s)	100% (0.016 s)	100% (0.016 s)
3000	100% (0.036 s)	100% (0.034 s)	100% (0.036 s)	100% (0.036 s)
4000	100% (0.064 s)	100% (0.062 s)	100% (0.059 s)	100% (0.062 s)
5000	100% (0.101 s)	100% (0.102 s)	100% (0.097 s)	100% (0.100 s)
6000	100% (0.144 s)	100% (0.144 s)	100% (0.145 s)	100% (0.155 s)
7000	100% (0.196 s)	100% (0.201 s)	100% (0.203 s)	100% (0.212 s)
8000	100% (0.263 s)	100% (0.282 s)	100% (0.272 s)	100% (0.281 s)
9000	100% (0.348 s)	100% (0.359 s)	100% (0.334 s)	100% (0.347 s)
10000	100% (0.444 s)	100% (0.416 s)	100% (0.434 s)	100% (0.425 s)

Table 3.36: *Average running time and percentage of solved instances with* $p_{i,2} = 2p_{i,1}, J_i \in J(12)$, $p_{k,1} = 2p_{k,2}, J_k \in J(21))$, $w_1 > 0$ *and* $w_2 > 0$

Number of jobs	Number of non-availability intervals			
	10	100	500	1000
1000	99.9% (0.008 s)	99.9% (0.008 s)	99.8% (0.008 s)	100% (0.008 s)
2000	99.9% (0.030 s)	100% (0.030 s)	100% (0.031 s)	99.8% (0.029 s)
3000	100% (0.069 s)	99.9% (0.070 s)	100% (0.068 s)	100% (0.069 s)
4000	100% (0.125 s)	100% (0.118 s)	99.8% (0.118 s)	100% (0.123 s)
5000	100% (0.206 s)	100% (0.191 s)	99.9% (0.191 s)	99.8% (0.191 s)
6000	100% (0.269 s)	100% (0.273 s)	100% (0.284 s)	99.9% (0.294 s)
7000	100% (0.387 s)	100% (0.377 s)	99.9% (0.398 s)	99.9% (0.382 s)
8000	100% (0.486 s)	100% (0.549 s)	100% (0.520 s)	100% (0.515 s)
9000	100% (0.700 s)	100% (0.644 s)	100% (0.654 s)	100% (0.677 s)
10000	100% (0.812 s)	100% (0.866 s)	100% (0.853 s)	100% (0.823 s)

the worst scenario among all scenarios T. Kouvelis et al. [187] focused on a scheduling problem, where the job processing times are uncertain. In their problem setting, the scheduling decision-maker is exposed to the risk that an optimal schedule with respect to a deterministic or stochastic model will perform poorly when evaluated relative to the *actual* processing times. A robust schedule was relatively insensitive to the potential realizations of the job processing times. The paper [187] focused on a two-machine flow shop problem with the makespan criterion. A measure of schedule robustness that explicitly considers the risk of poor system performance over all potential realizations of the job processing times was presented.

A similar robust schedule was constructed for a single machine problem by Daniels and Kouvelis [88]. In the robust scheduling approach the scenario

Table 3.37: *Average running time and percentage of solved instances with* $w = w_1$

Number of jobs	Number of non-availability intervals			
	10	100	500	1000
1000	100% (0.004 s)	100% (0.005 s)	100% (0.004 s)	100% (0.004 s)
2000	100% (0.016 s)	100% (0.016 s)	100% (0.016 s)	100% (0.016 s)
3000	100% (0.036 s)	100% (0.035 s)	100% (0.035 s)	100% (0.036 s)
4000	100% (0.063 s)	100% (0.061 s)	100% (0.060 s)	100% (0.061 s)
5000	100% (0.098 s)	100% (0.099 s)	100% (0.097 s)	100% (0.101 s)
6000	100% (0.152 s)	100% (0.142 s)	100% (0.151 s)	100% (0.141 s)
7000	100% (0.195 s)	100% (0.205 s)	100% (0.196 s)	100% (0.201 s)
8000	100% (0.257 s)	100% (0.273 s)	100% (0.285 s)	100% (0.267 s)
9000	100% (0.345 s)	100% (0.331 s)	100% (0.344 s)	100% (0.347 s)
10000	100% (0.430 s)	100% (0.427 s)	100% (0.432 s)	100% (0.424 s)

Table 3.38: *Average running time and percentage of solved instances with* $p_{i,2} = 2p_{i,1}, J_i \in J(12)$, $p_{k,1} = 2p_{k,2}, J_k \in J(21)$, $w = w_1$

Number of jobs	Number of non-availability intervals			
	10	100	500	1000
1000	100% (0.008 s)	99.8% (0.008 s)	99.9% (0.008 s)	99.9% (0.008 s)
2000	99.9% (0.031 s)	100% (0.030 s)	100% (0.029 s)	100% (0.028 s)
3000	100% (0.067 s)	99.9% (0.067 s)	100% (0.069 s)	100% (0.068 s)
4000	100% (0.123 s)	99.9% (0.118 s)	100% (0.119 s)	100% (0.118 s)
5000	100% (0.205 s)	100% (0.191 s)	100% (0.205 s)	100% (0.196 s)
6000	100% (0.280 s)	100% (0.269 s)	100% (0.293 s)	100% (0.286 s)
7000	100% (0.383 s)	100% (0.410 s)	100% (0.393 s)	100% (0.410 s)
8000	100% (0.519 s)	100% (0.544 s)	100% (0.533 s)	100% (0.526 s)
9000	100% (0.689 s)	100% (0.693 s)	100% (0.659 s)	100% (0.666 s)
10000	100% (0.847 s)	100% (0.848 s)	100% (0.861 s)	100% (0.830 s)

set T could contain a continuum of scenarios (i.e., T is the Cartesian product of the n closed intervals as defined in (4) on page 15) or just contains a finite number of m discrete scenarios (i.e., $T = \{p^j = (p_1^j, p_2^j, \ldots, p_n^j) : p^j \in R_+^n, j \in \{1, 2, \ldots, m\}\}$). A robust schedule minimizing the worst-case absolute or relative deviation from optimality has been used in [88, 189] to hedge against data uncertainty. For a scenario $p \in T$, let γ_p^t denote the optimal value of the objective function $\gamma = f(C_1, C_2, \ldots, C_n)$ for the deterministic single machine problem $1//\gamma$ with the fixed scenario p. Permutation $\pi_t \in S$ is optimal, if $f(C_1(\pi_t, p), C_2(\pi_t, p), \ldots, C_n(\pi_t, p)) = \gamma_p^t = \min_{\pi_k \in S} \gamma_p^k = \min_{\pi_k \in S} f(C_1(\pi_k, p), C_2(\pi_k, p), \ldots, C_n(\pi_k, p))$. For any permutation $\pi_k \in S$ and any scenario $p \in T$, the difference $\gamma_p^k - \gamma_p^t = r(\pi_k, p)$ is called the *regret* for permutation π_k with the objective function equal to γ_p^k under scenario p.

Table 3.39: *Average running time and percentage of solved instances with* $w = w_2$

Number of jobs	Number of non-availability intervals			
	10	100	500	1000
1000	100% (0.004 s)	100% (0.004 s)	100% (0.005 s)	100% (0.004 s)
2000	100% (0.016 s)	100% (0.016 s)	100% (0.016 s)	100% (0.016 s)
3000	100% (0.036 s)	100% (0.035 s)	100% (0.035 s)	100% (0.036 s)
4000	100% (0.062 s)	100% (0.063 s)	100% (0.059 s)	100% (0.063 s)
5000	100% (0.098 s)	100% (0.100 s)	100% (0.101 s)	100% (0.100 s)
6000	100% (0.145 s)	100% (0.142 s)	100% (0.141 s)	100% (0.142 s)
7000	100% (0.200 s)	100% (0.199 s)	100% (0.204 s)	100% (0.205 s)
8000	100% (0.267 s)	100% (0.285 s)	100% (0.265 s)	100% (0.268 s)
9000	100% (0.331 s)	100% (0.357 s)	100% (0.346 s)	100% (0.356 s)
10000	100% (0.422 s)	100% (0.430 s)	100% (0.439 s)	100% (0.428 s)

Table 3.40: *Average running time and percentage of solved instances with* $p_{i,2} = 2p_{i,1}, J_i \in J(12)$, $p_{k,1} = 2p_{k,2}, J_k \in J(21))$, $w = w_2$

Number of jobs	Number of non-availability intervals			
	10	100	500	1000
1000	99.9% (0.008 s)	100% (0.008 s)	100% (0.008 s)	99.8% (0.008 s)
2000	100% (0.029 s)	100% (0.030 s)	100% (0.029 s)	100% (0.030 s)
3000	100% (0.070 s)	99.9% (0.064 s)	100% (0.062 s)	100% (0.068 s)
4000	100% (0.124 s)	99.9% (0.116 s)	99.9% (0.118 s)	100% (0.123 s)
5000	100% (0.193 s)	100% (0.189 s)	100% (0.197 s)	100% (0.196 s)
6000	100% (0.265 s)	100% (0.296 s)	100% (0.261 s)	100% (0.291 s)
7000	100% (0.393 s)	100% (0.399 s)	100% (0.391 s)	100% (0.410 s)
8000	100% (0.495 s)	100% (0.543 s)	100% (0.550 s)	100% (0.505 s)
9000	100% (0.678 s)	100% (0.675 s)	100% (0.666 s)	100% (0.676 s)
10000	100% (0.836 s)	100% (0.864 s)	100% (0.855 s)	100% (0.846 s)

For permutation $\pi_k \in S$, the value $Z(\pi_k) = \max\{r(\pi_k, p) \ : \ p \in T\}$ is called the worst-case absolute regret while the worst-case relative regret is defined as

$$Z'(\pi_k) = \max\left\{\frac{r(\pi_k, p)}{\gamma_p^t} \ : \ p \in T\right\}$$

provided that $\gamma_p^t \neq 0$.

In [19, 20, 88, 205, 221, 222, 223, 371], a finite number of discrete scenarios have been considered. In [88, 205, 371], problem $1/p_i^L \leq p_i \leq p_i^U / \sum \mathcal{C}_i$ of minimizing total flow time was considered. While the deterministic problem $1//\sum \mathcal{C}_i$ is polynomially solvable in $O(n \log_2 n)$ time due to Smith [301], finding a permutation $\pi_t \in S$ of minimizing the worst-case absolute regret $Z(\pi_t)$ (see [88]) and the relative regret $Z'(\pi_t)$ (see [371]) are both binary

NP-hard even for two possible scenarios. The latter problem becomes unary NP-hard in the case of an unbounded number of discrete scenarios [371].

In [20], it was demonstrated by an example that there is no direct relationship between a given finite set of discrete scenarios and a set with a continuum of scenarios regarding the complexity of the uncertain problem. In [205], it was proven that minimizing the worst-case absolute regret $Z(\pi_k)$ for problem $1/p_i^L \le p_i \le p_i^U / \sum \mathcal{C}_i$ is binary NP-hard even if the closed intervals $[p_i^L, p_i^U]$ for all jobs $J_i \in J$ have the same center in the real axis. In [187], binary NP-hardness was proven for finding a permutation $\pi_t \in S$ that minimizes the worst-case absolute regret $Z(\pi_t)$ for the uncertain two-machine flow-shop problem with the $\mathcal{C}_{max}$ criterion (i.e., with minimizing the makespan $\max\{C_{max}(\pi_t, p) : J_i \in J\}$) even for two possible scenarios.

Only a few special cases are known to be polynomially solvable for minimizing the worst-case regret for the uncertain scheduling problems. Namely, an $O(n^4)$ algorithm was developed [180] for minimizing the worst-case regret for problem $1/p_i^L \le p_i \le p_i^U, d_i^L \le d_i \le d_i^U / \mathcal{L}_{max}$ with the criterion L_{max} of minimizing the maximum lateness: $\max\{C_i(\pi_t, p) - d_i : J_i \in J\} = \min_{\pi_k \in S}\{\max\{C_i(\pi_k, p) - d_i : J_i \in J\}\}$ when both the intervals of the processing times and the intervals of the due dates d_i are given as input data. In [205], it was proven that minimizing $Z(\pi_k)$ for problem $1/p_i^L \le p_i \le p_i^U / \sum \mathcal{C}_i$ can be realized in $O(n \log_2 n)$ time, if all intervals $[p_i^L, p_i^U]$, $J_i \in J$, of the scenarios have the same center provided that the number n of jobs is even. In [21], an $O(m)$ algorithm was proposed for minimizing the worst-case regret for the m-machine two-job flow shop problem $Fm/p_i^L \le p_i \le p_i^U, n = 2/\mathcal{C}_{max}$ provided that each of the m machines processes the jobs $J = \{J_1, J_2\}$ in the same order, i.e., only two semiactive schedules exist.

Bertsimas and Sim [31, 32] limited the conservatism of a robust solution by arguing that it is unlikely that all input data assume their worst possible values simultaneously whereas both absolute robustness and relative robustness seek solutions for such an assumption. In most practical situations, it is unlikely that all numerical parameters simultaneously take their worst values from the set T of possible scenarios. In [31, 32, 369], a restricted version of absolute robustness was introduced in order to make the robustness measure more useful for practice. In [369], it was proven that the r-restricted robust deviation version of the maximization problem on a uniform matroid is polynomially solvable. Computational experiments reported in [369] demonstrated that an r-restricted robust measure did not lead to a decrease in robustness and they have shown a superior behavior to the previously used robust measure.

Other robust decision-making formulations were presented by Artigues et al. [18], Briand et al. [49], Chen et al. [69], Daniels and Carrillo [86], Goren and Subuncuoglu [142], James and Buchanan [170], Jensen [175], Hall and Potts [156], Rosenblatt and Lee [281], Kouvelis et al. [188], Kutanoglu and Wu [197], Liu et al. [234], Mulvey et al. [252], Penz et al [265] and Wu et al. [367]. In particular, Daniels and Carrillo [86] considered β-robust scheduling as finding the job sequence which maximizes the probability of the flow time being less than the given limit. It is assumed that the probability distribution of the job processing times is known in advance. The same robustness measure was used in [367] for a single-machine scheduling problem, where the random processing times are normally distributed. Leon et al. [214] considered robustness measures and robust scheduling methods that generate job shop schedules that maintain a high performance over a range of system disturbances. James and Buchanan [170] studied the minimization of the weighted sum of early and tardy penalties for each job, each penalty being proportional to the amount of time the job is early or tardy. The authors examine the sensitivity of a schedule to errors in estimating the penalties. In [175], two robustness measures for a job shop were introduced. The first one was based on the idea of minimizing the makespan of a set of schedules instead of that of a single schedule. The second robustness measure was defined as an estimate of the first one in order to reduce the computational time. By computational experiments, it was demonstrated that, using a genetic algorithm, it is possible to find a rather robust schedule. Such a schedule performs better in rescheduling after a machine breakdown. In the experiments, it was assumed that the breakdown duration was known when the breakdown started. Hall and Potts [156] considered different scheduling problems, where a subset of the given jobs is already scheduled to minimize the objective function when a new set of jobs arrives and so creates a disruption. The scheduler has to insert the new jobs into the existing schedule without excessively disrupting it. Two questions were considered by the authors. How to minimize the objective function, subject to a limit on the disruption caused to the original schedule? How to minimize both the objective function and the cost of disruption. For all scheduling problems considered in [156], Hall and Potts provided either a polynomial algorithm or a proof that such an algorithm does not exist unless $\mathcal{P} = \mathcal{NP}$. Liu et al. [234] considered both robustness and stability when generating a predictive schedule for a single machine scheduling problem with minimizing the total weighted tardiness of the jobs. When generating the predictive schedule, all data are known exactly, except the machine breakdown occurring time

and the duration. The probability distribution and the mean duration of a breakdown are also available for the scheduler. The stability of the predictive schedule is measured by the sum of the absolute deviations of the job completion times between the realized schedule and the predictive one. In [213], a game-like approach was proposed for rescheduling in a job shop after a machine breakdown. A game tree was constructed to evaluate the consequences of the decisions on the future performance. The rescheduling decisions in the game tree were evaluated in order to find the decisions leading to the lowest expected makespan (provided that the tree holds a random sample of the possible scenarios).

Wu et al. [368] studied the weighted tardiness job shop problem. A basic thesis of the latter paper is that "global scheduling performance is determined primarily by a subset of the scheduling decisions to be made". Wu et al. [368] proposed a solution approach to the problem as follows. First, to identify a critical subset of the scheduling decisions at the beginning of the planning horizon and to relegate the rest of the scheduling decisions to future points in time. The above research presents a philosophy of great practical importance: Local scheduling should be allowed to be sufficiently flexible without losing a global view of the system. Note that our approach to scheduling problems with uncertainty considered in Chapters 2 and 3 uses a similar philosophy but in a formal way.

Article [7] (and many other articles published in the proceedings of international conferences on parallel and distributed computer systems) addressed different measures of robustness of a resource allocation with respect to desired computer system performance features against multiple perturbations in a multiple system and the environment condition. The robustness measure introduced and studied in [7] is close to the philosophy of the stability radius introduced in Chapter 1 and 2 of this book but it is more general since it is applied to a more general processing system.

The approach under consideration in Chapters 2 and 3 was originally proposed in [200, 201] for the makespan criterion and was developed in [203] for the total completion time criterion $\sum \mathcal{C}_i$. In particular, the formula for calculating the *stability radius* of an optimal schedule (i.e., the largest value of simultaneous independent variations of the job processing times such that this schedule remains optimal) has been provided in [201]. In paper [203], a stability analysis of a schedule minimizing total completion time was involved in a branch and bound method for solving the job shop problem $\mathcal{J}m/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\sum \mathcal{C}_i$ with m machines and different technological routes of n jobs. In [16], for the two-machine flow shop problem

$\mathcal{F}2/p_{ij}^L \le p_{ij} \le p_{ij}^U/\sum \mathcal{C}_i$, sufficient conditions have been found when a transposition of two jobs minimizes the makespan. The total completion time in a flow shop with uncertain processing times was studied in [318, 330]. In particular, a geometrical algorithm has been developed for solving the flow shop problem $\mathcal{F}m/p_{ij}^L \le p_{ij} \le p_{ij}^U, n = 2/\sum \mathcal{C}_i$ with m machines and two jobs [330]. For a flow shop problem with two machines and one with three machines, sufficient conditions have been found when a transposition of two jobs minimizes total completion time. Paper [13] was devoted to the case of separate setup times to minimize the makespan $\mathcal{C}_{max}$ or the total completion time $\sum \mathcal{C}_i$. Namely, the processing times are fixed while the setup times are relaxed to be distribution-free random variables with only the lower and upper bounds for the setup times being given before scheduling. In [16], local and global dominance relations were found for an uncertain flow shop problem with two machines. The two-machine job shop scheduling problem with bounded processing times was studied in [328].

Minimizing total weighted flow time of n jobs with interval processing times on a single machine was investigated in [323], where a minimal dominant set $S(T)$ of permutations of the n jobs was characterized. Necessary and sufficient conditions have been proven for the case $|S(T)| = 1$ and for the case $|S(T)| = n!$. Computational experiments have shown that the established precedence-dominance relations are useful in reducing the total weighted flow time.

The results of Section 3.1 were originally published in [219, 220, 237, 238, 258], the results of Section 3.2 in [106, 217, 220, 218, 216, 237], the results of Section 3.3 in [219, 331]. It should be noted that the uncertainties of the processing times in the problems $\alpha/p_{ij}^L \le p_{ij} \le p_{ij}^U/\gamma$ considered in Chapters 2 and 3 are due to external forces while in a scheduling problem with *controllable processing times* of jobs in a shop or in a project, the objective is both to set the processing times and find an optimal schedule (see articles [70, 71, 72, 87, 128, 150, 165, 166, 172, 173, 185, 294, 296, 297, 345, 359, 360]). Most known results obtained for scheduling problems with controllable processing times have been summarized in the monograph [129].

The stability analysis presented in Chapter 2 and used in Chapter 3 is a generalization of the irreducibility analysis developed by Bräsel et al. [39, 40, 41, 42, 43] for a possible variation of a job processing time in the range $[0, +\infty)$ (see [93] for a survey). Instead of the mixed (disjunctive) graph model described in Section 1.1, Bräsel et al. used the block-matrices model which allowed them to present the structural data of a shop scheduling problem by means of a special latin rectangle, also called a sequence. A

set of sequences is called a potentially optimal solution (irreducible set of sequences) of a shop scheduling problem if it contains an optimal sequence for each scenario (it is a minimal potentially optimal solution with respect to inclusion). In [40, 41, 42, 43] (in [39, 41], respectively), an irreducible set of sequences was investigated for the open (classical job) shop scheduling problem with the makespan criterion.

Personal financial planning is the preparation of target-oriented decisions concerning personal asserts, incomes and expenses. A part of comprehensive personal financial planning consists of scheduling future incomes and expenses during the planner's lifetime. In [47], the authors adopted the view on the realization of time points of extraordinary incomes and expenses. They developed a model of personal financial scheduling and derived a corresponding mathematical program. Solving the integer programming problem gives an optimal financial schedule concerning personal goals and preferences. Since some input data for personal financial scheduling are random in nature, it was assumed that only lower and upper bounds for the real value of the uncertain parameter are known before personal financial scheduling. In particular, instead of using exact values for credit and interest rates, a worst-case analysis for personal financial scheduling based on lower and upper bounds of credit and interest rates was provided. Such an analysis is useful to valuate a credibility of a personal financial schedule.

Several *on-line* models and algorithms have been proposed (see e.g. [22, 23, 66, 80, 85, 103, 105, 182, 283, 291]), and the main difference between these models are the assumptions on the information that becomes available to the scheduler. For a description of these on-line models, we refer to the survey by Sgall [283]. According to [66], *on-line* means that jobs arrive over time, and all job characteristics become known at their arrival time [66]. Jobs do not have to be scheduled immediately upon arrival. At each time a machine is idle and a job is available, the algorithm decides which one of the available jobs is scheduled, if any. An on-line algorithm for the problem of scheduling jobs on identical parallel machines with the objective of minimizing the makespan was proposed and analyzed by Chen and Vestjens [66]. This problem is NP-hard when the *off-line* version is considered, although it can be solved in polynomial time by an on-line algorithm if preemption is allowed [66]. Seiden [291] studied on-line scheduling of jobs with fixed starting and completion times. Jobs must be scheduled on a single machine which runs at most one job at a given time. The problem is on-line since jobs are unknown until their starting times. Each job must be started or rejected immediately when it becomes known. The goal is to maximize the sum of the value of

the payoff (the sum of the values of those jobs which run to completion).

Edmonds [103] studied the on-line problem when the scheduler has no information about the jobs (which have to be processed) except for knowing when a job arrives and when a job completes. A job can arrive at an arbitrary time and may have an arbitrary number of phases. Moreover, the execution characteristics of each job phase can be anywhere being fully parallelizable or being completely sequential. The non-clairvoyant scheduler makes scheduling decisions (continuously allocates and reallocates machines to jobs as they arrive or complete) having no knowledge about the jobs. The objective is to minimize the average response time, i.e., the time during which a job has arrived but has not completed. The non-clairvoyant scheduling is a model in a time-sharing operating system, where the scheduler must provide a fast switch between machines without any knowledge of the future behavior of jobs and machines. Edmonds called such an uncertain situation as "scheduling in the dark". He proved that the scheduler Equi-partition performs within a constant factor as well as the optimal scheduler as long as there are given at least twice as many machines. The extra machines are enough to compensate for some machine being wasted on sequential jobs. Equi-partition scheduling partitions the machines evenly among all the jobs that are alive. The articles [103, 104, 105, 182] continued similar research for some modification of "scheduling in the dark". In [104], it was proven that Equi-partition efficiently schedules batch jobs with different execution characteristics when all the jobs arrive at the same time. In [182], it was assumed that the weight of a job becomes known when the job arrives, while its processing time remains unknown until it will be completed. The objective is to minimize the total weighted flow time.

The articles [97, 98] were devoted to supply planning in an MRP environment. The main problem considered in these papers was to find the planned lead time, which minimize expected backlogging and holding cost. Along with demand uncertainty, lead time uncertainty was included into consideration. In [97], supply planning was analyzed on the basis of a Markov model. Yang [372] proposed to explore a modified single machine problem to maximize the job revenue. He noticed that products like chips and computers are characterized by short life cycles and rapidly declining sales prices. This implies that the total amount of revenue generated as a result of completing a job (product) is decreasing as its completion time is delayed. So, the performance measure was to maximize the product revenues. Based on the assumption that a decreasing rate of revenue is dependent on the jobs, Yang [372] developed a branch-and-bound algorithm (and heuristics) to locate op-

timal (near-optimal) job sequences and thereby to maximize the total job revenue.

In most of the shop scheduling models, it was assumed that an individual processing time incorporates all other time parameters (lags) attached to a job or to an operation. In practice, however, such parameters often have to be viewed separately from the actual processing times. For example, if for an operation some pre-processing and/or post-processing is required, then we obtain a scheduling model with *set-up* and/or removal times separated. Strusevich [346] considered a two-machine open shop problem with involved interstage transportation times. He assumed that there is a known *time lag* (transportation time) between the completion of an operation and the beginning of the next operation of the same job. Sequence-dependent setup times in a two-machine job shop problem with minimizing the makespan were investigated in [324]. In [10], Allahverdi considered a two-machine flow shop problem with the objective to minimize the expected makespan, where machines suffer breakdowns and the job set-up and removal times were separated from the processing times. The same author [9] proposed a dominance relation, where no assumption about the breakdown processes was made. In some special cases (if certain assumptions about the distributions of the breakdowns and counting processes hold), it is possible to obtain an optimal schedule.

The approach presented in Chapter 3 of this book was implemented by Allahverdi [12] for the two-machine flow shop problem with the total completion time objective, interval processing and setup times. In [99], a general model motivated by a concrete industrial application has been developed. The authors proposed a two-phase method based on solving a mixed-integer programming problem and improving the initial schedule by a tabu search heuristic. Such an approach may be used to handle various scheduling problems when there are sequence-dependent set-up times, the jobs have to be processed in batches and the machines have non-availability intervals.

A number of papers have been devoted to makespan minimization in a flow shop scheduling problem with two machines which have $w \geq 1$ non-availability intervals, see the survey [288]. The scheduling problem with an *availability constraint* is very important, as it happens often in industry [99]. For example, a machine may not be available during the scheduling horizon due to a breakdown (stochastic) or preventive maintenance (deterministic). In an *on-line* setting, machine availabilities are not known in advance. Unexpected machine breakdowns are a typical example of events that arise on-line. Sometimes schedulers have a partial knowledge of the availabilities,

i.e., they have some 'look-ahead' information. They might know the next time interval when a machine requires a maintenance or they might know when a broken machine will be available again [285]. In an *off-line* setting, one assumes complete information, i.e., all machine availabilities are known prior to the schedule generation [288]. In [195], it was proven that problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ is unary NP-hard if an arbitrary number w of non-availability intervals occurs on one of the two machines. In [206], it has been shown that this problem is binary NP-hard even if there is a single non-availability interval ($w = 1$) either of machine M_1 or of machine M_2 (these results have been presented in Theorems 3.23 and 3.24 on pages 236 and 238, respectively).

In [374], NP-hardness of the following problem was proven. A set of jobs has to be processed using a single resource, where the availability of the resource varies over time. For each job J_i, r_i denotes the resource requirement. Once r_i units of the resource are applied to job J_i, this job is completed. The objective is to minimize the total weighted completion time.

In [206], a heuristic was provided with a makespan which is at most (3/2) times larger than the minimal makespan if this non-availability interval is on the first machine, and (4/3) times larger than the minimal makespan if the non-availability interval is on the second machine. In [78], it was shown that the error bound of (3/2) for the situation with a non-availability interval of the first machine is tight. For the problem with availability constraints imposed on each machine when the non-availability interval of one machine is followed immediately by the non-availability interval of the other machine, paper [77] provided a heuristic with a worst-case error bound of (5/3).

In [195], a branch-and-bound algorithm was developed which outperformed the dynamic programming algorithm proposed in [206]) for the case of one non-availability interval. In [257], for problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$ with one non-availability interval, *a fully polynomial approximation scheme* has been developed with complexity $O(n^5/\epsilon^4)$, where ϵ is an upper bound for the amount of exceeding the minimal makespan in the approximate schedule that may be obtained. Problem $\mathcal{F}m,NC^{off}/pmtn/\mathcal{C}_{max}$ was considered in [3], where a heuristic approach was developed on the basis of the geometrical algorithm from [305, 308] for the two-job shop scheduling problem with any regular criterion.

In [45], the stability of a Johnson schedule constructed for problem $\mathcal{F}2//\mathcal{C}_{max}$ has been tested for the case of problem $\mathcal{F}2,NC^{off}/pmtn/\mathcal{C}_{max}$. Section 3.4 (except Theorems 3.23 and 3.24) was written on the basis of article [45]. Section 3.5 was written on the basis of article [46]. Reviews in

this area were given in [195, 207, 286, 288].

It is clear that a mass uncertain problem $\alpha/p_{ij}^L \leq p_{ij} \leq p_{ij}^U/\gamma$ cannot be easier (in an asymptotical sense) than its deterministic counterpart $\alpha//\gamma$ since the latter is a special case of the former (if $p_{ij}^L = p_{ij}^U$ for each $J_i \in J$ and $M_j \in M$). So, the statement by Lenstra [210] about "the mystical power of twoness" that a shop scheduling problem may be solved in polynomial time only if at least one number of machines or jobs is restricted by two (and $\mathcal{P} \neq \mathcal{NP}$) is applicable to the uncertain problems $\alpha|p_{ij}^L \leq p_{ij} \leq p_{ij}^U|\gamma$, too. As it was proven in [308, 313], the deterministic problems $\mathcal{F}/n = 3/\mathcal{C}_{max}$ and $\mathcal{F}/n = 3/\sum \mathcal{C}_i$ and all other deterministic flow shop scheduling problems $\mathcal{F}/n = 3/\Phi$ with a non-trivial regular criterion Φ are binary NP-hard. Even the deterministic problems $\mathcal{J}3/n = 3/\mathcal{C}_{max}$ and $\mathcal{J}3/n = 3/\sum \mathcal{C}_i$ and so other job shop scheduling problems $\mathcal{F}/n = 3/\Phi$ are binary NP-hard [326]. The complexity of deterministic shop scheduling problems with a fixed number of jobs or (and) machines was surveyed in [54, 210, 295].

Table 3.41: *Notations for the two-machine flow shop and job shop*

Symbols	Description
π_k	Permutation of the set of jobs $J = \{1, 2, \ldots, n\}$: $\pi_k = \{J_{k_1}, J_{k_2}, \ldots, J_{k_n}\}$,
S^π	Set of all permutations $S^\pi = \{\pi_1, \pi_2, \ldots, \pi_{n!}\}$ of n jobs J defining the set of all *permutation schedules*
$S^\pi(T)$	Set of permutations $S^\pi(T) \subseteq S^\pi$ which is a solution to the *uncertain* flow shop problem
N_1	Set of jobs with $p_{i1} \leq p_{i2}$
N_2	Set of jobs with $p_{i1} \geq p_{i2}$
N_{M_1}	Set of jobs with left intervals of job processing times on machine M_1
N_{M_2}	Set of jobs with left intervals of job processing times on machine M_2
$J(ij)$	Subset of jobs of set J with machine route (M_i, M_j), where $\{i, j\} = \{1, 2\}$
$J(i)$	Subset of jobs of set J with one-machine route (M_i), where $i \in \{1, 2\}$
$C_i(\pi_k, p)$	Completion time of job $J_i \in J$ in schedule $\pi_k \in S^\pi$ provided that the processing times are given by vector $p \in T$
$C_{max}(\pi_k, p)$	Maximal completion time of schedule $\pi_k \in S^\pi$ provided that the processing times are given by vector $p \in T$: $C_{max}(\pi_k, p) = \max\{C_i(\pi_k, p)$
T^J	Set of feasible vectors of the job processing times in an uncertain job shop problem: $T^J = T(1) \cup T(2) \cup T(12) \cup T(21)$, where $T(w)$ defines the intervals of the feasible processing times of jobs $J(k)$, $k \in \{1, 2, 12, 21\}$
w_j	Number of non-availability intervals of machine M_j
w	Total number of non-availability intervals: $w = w_1 + w_2$
N_{kj}	k^{th} non-availability interval of machine M_j
$s(N_{kj})$	Starting point of the k^{th} non-availability interval of machine M_j
$f(N_{kj})$	Endpoint of the k^{th} non-availability interval of machine M_j
$h(N_{kj})$	Length of the k^{th} non-availability interval of machine M_j: $h(N_{kj}) = f(N_{kj}) - s(N_{kj})$
r_{ij}	Maximal possible enlargement of the processing time of job J_i on machine M_j such that a Johnson permutation is not changed
ρ_j	Stability radius of a Johnson permutation on machine M_j
A_{kj}	k^{th} availability interval of machine M_j
s_j	Earliest possible starting time of a job on machine M_j
c_j	Latest possible completion time of a job on machine M_j
d_{ij}	Maximal possible enlargement of the scheduling time of operation O_{ij} forced by the non-availability intervals of machine M_j
δ_j	Enlargement radius of the scheduling times of the operations on machine M_j
$s_{ij}(\pi_k)$	Starting time of operation O_{ij} in the schedule defined by permutation π_k
$c_{ij}(\pi_k)$	Completion time of operation O_{ij} in the schedule defined by permutation π_k

Conclusion

It is difficult for us to write a conclusion since the work we reported in this book is definitely not finished yet. Writing this book, we are forced to recognize that we are only at the beginning of the road to develop mathematical tools for dealing with scheduling problems with uncertain parameters. Next, we summarize what we have learned from studying the scheduling paradigm when the job processing times are uncertain before scheduling. We outline also some topics for future research which follow from the above results.

In spite of a large number of papers and books published about scheduling, the utilization of numerous results of scheduling theory in most production environments is far from the desired volume. One of the reasons for such a gap between scheduling theory and practice is connected with the usual assumption that the processing times of the jobs are known exactly before scheduling or that they are random values with a priori known probability distributions. In this book, a model of one of the more realistic scheduling scenarios was considered. It is assumed that in the realization of a schedule, the job processing time may take any real value between given lower and upper bounds (within the given polytope T), and there is no prior information about the probability distributions of the random processing times. For such an uncertain scheduling problem, there does usually not exist a unique schedule that remains optimal for all possible realizations of the processing times and a set of schedules has to be considered which dominates all other schedules for the given criterion. To find such a set of schedules, our idea was to use a stability analysis of an optimal schedule with respect to the perturbations of the processing times (Chapter 1).

In Chapter 2, we introduced the notion of the relative stability radius of an optimal schedule s as the maximal value of the radius of a stability ball within which schedule s remains the best among the given set B of schedules (Definition 2.2 on page 95 for the makespan criterion $\mathcal{C}_{max}$ and Definition 2.6 on page 126 for the mean flow time criterion $\sum \mathcal{C}_i$). The relativity was considered with respect to the polytope T of feasible vectors of the processing times and with respect to the set B of semiactive schedules

for which the superiority of a schedule s at hand has to be guaranteed.

In Chapters 1 and 2, we used the mixed (disjunctive) graph model which is suitable for the whole scheduling process from the initial mixed graph G representing the input data until a final digraph G_s representing a semiactive schedule s. The mixed graph model may be used for different requirements on the numerical input data (Table 2.1 on page 86). Most results of Chapters 1 and 2 are formulated in terms of paths in the digraphs G_s.

In Chapter 2, we focused on dominance relations between feasible schedules taking into account the given polytope T (Section 2.2). We established necessary and sufficient conditions for the case of an infinitely large relative stability radius of an optimal schedule s for the maximum flow time criterion (Theorem 2.2 on page 101). Under such conditions, schedule s remains optimal for any feasible perturbations of the processing times. We established also necessary and sufficient conditions for the case of a zero relative stability radius of an optimal schedule s (Theorem 2.1 on page 98). Under such conditions, the optimality of schedule s is unstable: There are some small changes of the given processing times which imply that another schedule from set B will be better (will have a smaller length) than schedule s.

Formulas for calculating the exact value of the relative stability radius were based on a comparison of an optimal schedule s with other schedules from set B (Theorem 2.3 on page 103), and we showed how it is possible to restrict the number of schedules from set B examined for such a calculation of the relative stability radius (Lemma 2.4 on page 114). To this end, we considered the schedules from set B in non-decreasing order of the values of the objective function until some inequalities hold (Corollary 2.4 on page 116).

In Chapter 2, analogous results were obtained for the mean flow time criterion, and the focus was on dominance relations between feasible schedules taking into account the given criterion (Definition 2.5 on page 118). Formulas for calculating the exact value of the relative stability radius were given in Theorem 2.6 on page 131. We established necessary and sufficient conditions for an infinitely large relative stability radius of an optimal schedule for the mean flow time criterion (Theorem 2.7 on page 131) and necessary and sufficient conditions for a zero relative stability radius of an optimal schedule (Theorem 2.8 on page 133). Using these results, we developed several exact and heuristic algorithms for constructing a solution and a minimal solution (Definition 2.1 on page 86) of a scheduling problem with uncertain processing times. The developed software was tested on randomly generated job shop problems. For the maximum and mean flow time criteria, we calculated

the stability radii of the optimal schedules for more than 10,000 randomly generated instances. For randomly generated uncertain scheduling problems with the same criteria, we constructed G-solutions and minimal G-solutions. The most critical parameter for the running time of the programs was the number of edges in the mixed graph.

In conclusion, we present some topics for future research. The most part of this book was devoted to the scheduling problem with uncertain data for the criteria $\mathcal{C}_{max}$ and $\sum \mathcal{C}_i$. Other criteria may be a subject of further research. In Section 3.2, we introduced the *on-line* problem (when an uncertain processing time becomes known after the realization of some jobs), which follows after the *scheduling problem*. For the *on-line* problem, only very preliminary results are known and it may be a subject for future research.

The next important direction for further research is to construct more efficient algorithms for the uncertain scheduling problem, in particular, to use this approach for uncertain scheduling problems whose deterministic counterpart have polynomial algorithms for constructing optimal schedules.

After carrying out computational experiments for the calculation of the stability radii of an optimal schedule (see Chapter 1), we can select the following topics for future research. It is useful to develop further a branch-and-bound algorithm for constructing the k best schedules (instead of one, which is usually constructed) and to combine such a calculation with a stability analysis. Another possible topic is to improve the bounds (2.34) on page 114 and (2.63) on page 148 in order to restrict the number of digraphs G_s, with which an optimal digraph has to be compared, while calculating its stability radius. A more complex question is to find simpler (practical) formulas for calculating the stability radius or at least lower and/or upper bounds for it (without considering the paths of digraph G_s). The algorithms used in Chapters 1 and 2 were based on an direct enumeration scheme for calculating the stability radii, and an implicit enumeration scheme was used *before* performing the stability analysis (i.e., for calculating optimal and near optimal schedules). The application of the stability analysis *within* an implicit enumeration framework should have a practical utility and it may be a topic for future research. Finishing this book about scheduling under uncertainty, we can certainly claim that uncertain scheduling problems remain an interesting and challenging subject for future studies which may combine different theoretical results with practical problems.

Index

Actual processing time, 274
Anti-reflexivity, 28
Arc
  disjunctive, 19
  non-transitive, 18
Arc tolerance, 174
Availability interval, 241

Backtracking, 118
Branch-and-bound, 152

Capacity constraints, 18
Chain, 18
Chebyshev metric, 26
Completion time, 18
  earliest, 21, 81
Conflict edge, 153
Conflict measure, 154
Controllable duration, 5
Criterion
  makespan, 30
  maximum flow time, 10, 30
  mean flow time, 10
  regular, 10, 17, 22
Critical
  path, 30, 81
    method, 118
  set of paths, 44, 118
  sum of weights, 44, 124
  weight, 31, 81, 93

Density function, 82
Digraph, 20
  acyclic, 21
  competitive, 38, 89, 102, 107
  feasible, 20
  optimal, 21, 81
Dominance relation, 27
  for digraphs, 116
  on set of paths, 93, 94
Due date, 24, 25

Earliest due date, 75
Expected value, 83

Flow shop, 5, 9, 178
Fuzzy scheduling, 75

G-solution, 84, 116, 122, 161
  minimal, 84, 116
  of problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$, 83
  single-element, 123
Gantt chart, 22
General shop, 5, 11, 17
Graph
  directed, *see* Digraph
  disjunctive, 18, 19
  mixed, 18

Hoist scheduling, 5

Individual problem, 84
Infinite relative stability radius, 99
Input data
  numerical, 13, 19
  structural, 13, 18, 19

Job, 3, 7, 9
  completion time, 10

duration, 7
shop, 9, 18, 23
weight, 25
Job shop, 5, 255
classical, 162

List scheduling problem, 172
Longest processing time, 227
Lower bound, 154
for the processing time, 8, 13

Machine, 3, 7, 9
Machine route, 9
Makespan, 10
Mass general shop problem, 84
Maximum lateness, 72
Maximum metric, 26
Minimal G-solution
for problem $\mathcal{G}/a_i \leq p_i \leq b_i/\Phi$, 84
Mixed graph
weighted, 19, 81
Model
deterministic, 7, 10
stochastic, 4, 8
Multi-stage system, 9, 72
Multiset, 42

Non-availability interval, 232, 233
Non-linear programming, 28
NP-hard problem, 72, 174

Objective function, 10, 17
Off-line phase, 8
On-line phase, 5, 8
Open shop, 47
Operation, 3, 9
completion time, 10, 17
dummy, 86
processing time, 17
starting time, 10
Operation preemption, 10, 233
Optimal digraph
stable, 29, 99
unstable, 29, 99

Path
dominant, 28
Personal financial
planning, 281
scheduling, 281
Planning horizon, 14
Polynomially solvable problem, 72, 174
Polytope, 27
feasible, 83
Potentially optimal schedules, 8
Precedence constraints, 17, 18
Preschedule, 75
Probability distribution
cumulative, 82
exponential, 82
Problem
NP-hard, 16
partition, 234
unary NP-hard, 22
Processing time, 7
controllable, 74, 280
random, 8
uncertain, 13, 82

Recirculation, 48
Relation
strict order, 28
Relative stability radius, 93, 124
infinite, 129
Release date, 24, 25
Removal time, 24
Rescheduling problem, 74
Resource constrained project scheduling, 12

Resource constraints, 18
Route, 9, 18

Schedule, 3, 10, 18
  combinatorial structure, 172
  feasible, 17, 20
  optimal, 21, 30
    potentially, 115
  semiactive, 20
  signature, 22
  temporal structure, 172
Scheduling, 3
  off-line, 4
  on-line, 4
  policy, 83
    dynamic, 85
    static, 85
  proactive, 4
  problem
    deterministic, 7
    stochastic, 8, 82
    with uncertain data, 13
  problem in real-time, 74
  reactive, 4, 75
  single machine, 5
  time, 240
Sequencing, 3
Set of representatives, 42, 117
  critical, 118
Setup time, 24
  sequence-dependent, 25
  sequence-independent, 25
Shop, 9
Shortest processing time, 75, 227
Single-stage system, 72
Solution
  tree, 16
Stability, 3
  analysis, 8, 13–15
    a posteriori, 48
  ball, 26
  radius, 13, 15, 27
    infinitely large, 29
    upper bound, 66
    zero, 29
  region, 54, 91, 123
Starting time, 18
  earliest, 118
  latest, 154
Strong dominance relation
  for digraphs, 116
Subgraph
  minimal, 118
Symmetric difference of sets, 101

Technological route, 9, 18
  stage, 9
Temporal constraints, 17
Three-field notation $\alpha/\beta/\gamma$, 11
Transitivity, 28
Transportation time, 25
Traveling salesman problem, 173

Uncertainty, 3
Upper bound
  for $\overline{\varrho}_s(p)$, 47
  for $\widehat{\varrho}_s(p)$, 34
  for the processing time, 8, 13

Vector space, 26
Vertex
  weight, 19

Weighted digraph
  optimal, 21

Bibliography

[1] Abdelmaguid T.F. Permutation-induced acyclic networks for the job shop scheduling problem // Applied Mathematics Modelling. – 2009. – V. 33. – P. 1560–1572.

[2] Adams J., Balas E., Zawack D. The shifting bottleneck procedure for job shop scheduling // Management Science. – 1988. – V. 34. – P. 391–401.

[3] Aggoune R., Portmann M.-C. Flow shop scheduling problem with limited machine availability: A heuristic approach // International Journal of Production Economics. – 2006. – V. 99. – P. 4–15.

[4] Ahmadizar F., Ghazanfari M., Ghomi E.M.T.F. Application of chance-constrained programming for stochastic group shop scheduling problem // International Journal of Advanced Manufacturing Technology. – 2009. – V. 42. – P. 321–334.

[5] Akers S.B. A graphical approach to production scheduling problems // Operations Research. – 1956. – V. 4. – P. 244–245.

[6] Akturk M.S., Gorgulu E. Match-up scheduling under a machine breakdown // Journal of the Operations Research Society. – 1999. – V. 112. – P. 81–97.

[7] Ali S., Maciejewski A.A., Siegel H.J., J.-K. Kim Measuring the robustness of a resource allocation // IEEE Transactions on Parallel and Distributed Systems. – 2004. – V. 15. – Nº 7. – P. 630–641.

[8] Alidaee B., Womer N.K. Scheduling with time dependent processing times: Review and extensions // Journal of the Operations Research Society. – 1999. – V. 50. – P. 711–720.

[9] Allahverdi A. Two-stage production scheduling with separated setup times and stochastic breakdowns // Journal of the Operations Research Society. – 1995. – V. 46. – P. 896–904.

[10] Allahverdi A. Scheduling in stochastic flowshop with independent setup, processing and removal times // Computers & Operations Research. – 1997. – V. 24. – P. 955–960.

[11] Allahverdi A. Stochastically minimizing total flowtime in flowshops with no waiting space // European Journal of Operational Research. – 1999. – V. 113. – P. 101–112.

[12] Allahverdi A. Two-machine flowshop scheduling problem to minimize total completion time with bounded setup and processing times // International Journal of Production Economics. – 2006. – V. 103. – P. 386–400.

[13] Allahverdi A., Aldowaisan T., Sotskov Yu.N. Two-machine flowshop scheduling problem to minimize makespan or total completion time with random and bounded setup times // International Journal of Mathematics and Mathematical Sciences. – 2003. – V. 39. – № 11. – P. 2475–2486.

[14] Allahverdi A., Mittenthal J. Two-machine ordered flowshop scheduling under random breakdowns // Mathematical and Computer Modelling. – 1994. – V. 20. – P. 9–17.

[15] Allahverdi A., Mittenthal J. Dual criteria scheduling on a two machine flow shop subject to random breakdowns // International Transactions in Operations Research. – 1998. – V. 5. – P. 317–324.

[16] Allahverdi A., Sotskov Yu.N. Two-machine flowshop minimum-length scheduling problem with random and bounded processing times // International Transactions in Operations Research. – 2003. – V. 10. – № 1. – P. 65–76.

[17] Alyushkevich V.B., Sotskov Yu.N. Stability in production scheduling problem // Vesti Akad. Navuk BSSR, Ser. Fiz.-Techn. Navuk. – 1989. – № 3. – P. 102–107 (In Russian).

[18] Artigues C., Billaut J.-C., Esswein C. Maximization of solution flexibility for robust shop scheduling // European Journal of Operational Research. – 2005. – V. 165. – P. 314–328.

[19] Averbakh I. Minmax regret solutions for minmax optimization problems with uncertainty // Operations Research Letters. – 2000. – V. 27. – P. 57–65.

[20] Averbakh I. On the complexity of a class of combinatorial optimization problems with uncertainty // Mathematical Programming, Series A. – 2001. – V. 90. – P. 263–272.

[21] Averbakh I. The minmax regret permutation flow-shop problem with two jobs // European Journal of Operational Research. – 2006. – V. 169. – P. 761–766.

[22] Averbakh I. On-line integrated production-distribution scheduling problems with capacitated deliveries // European Journal of Operational Research. – 2009 (In press).

[23] Averbakh I., Xue Z. On-line supply chain scheduling problems with preemption // European Journal of Operational Research. – 2007. – V. 181. – P. 500–504.

[24] Aytug H., Lawley M.A., McKay K., Mohan S., Uzsoy R. Executing production schedules in the face of uncertainties: A review and some future directions // European Journal of Operational Research. – 2005. – V. 161. – P. 86–110.

[25] Baker K.R. Introduction to Sequencing and Scheduling // John Wiley, New York, USA. – 1974.

[26] Balakrishnan J., Cheng C.H. Multi-period planning and uncertainty issues in cellular manufacturing: A review and future directions // European Journal of Operational Research. – 2007. – V. 177. – P. 281–309.

[27] Balas E., Lenstra J.K., Vazacopoulos A. The shifting bottleneck procedure for job shop scheduling // Management Science. – 1992. – V. 34. – P. 391–401.

[28] Bandelloni M., Tucci M., Rinaldi R. Optimal resource leveling using non-serial dynamic programming // European Journal of Operational Research. – 1994. – V. 78. – P. 162–177.

[29] Bean J.C., Birge J.R., Mittenehal J., Noon C.E. Match-up scheduling with multiple resources, release dates and disruption // Operations Research. – 1991. – V. 39. – P. 470–483.

[30] Bellman R. Mathematical aspects of scheduling theory // Journal of the Society of Industrial and Applied Mathathematics. – 1956. – V. 4. – P. 168–205.

[31] Bertsimas D., Sim M. Robust discrete optimization and network flows // Mathematical Programming. Series B. – 2003. – V. 98. – P. 49–71.

[32] Bertsimas D., Sim M. The price of robustness // Operations Research. – 2004. – V. 52. – P. 35–53.

[33] Birge J.R., Glazebrook K.D. Bounds on optimal values in stochastic scheduling // Operations Research Letters. – 1997. – V. 21. – P. 107–114.

[34] Biskup D. Single-machine scheduling with learning consideration // European Journal of Operational Research. – 1999. – V. 115. – P. 173–178.

[35] Blazewicz J., Domschke W., Pesch E. The job shop scheduling problem: Conventional and new solution techniques // European Journal of Operational Research. – 1996. – V. 93. – P. 1–33.

[36] Blazewicz J., Ecker K., Schmidt G., Weglarz J. Scheduling in Computer and Manufacturing Systems // Springer-Verlag, Germany. – Berlin. – 1993.

[37] Blazewicz J., Ecker K.H., Pesch E., Schmidt G., Weglarz J. Scheduling in Computer and Manufacturing Processes // Springer-Verlag, Germany. – Berlin. – 1996.

[38] Bockenhauer H.-J., Forlizzi L., Hromkovie J., Kneis J., Kupke J., G. Proietti, Widmayer P. Reusing Optimal TSP Solutions for Locally Modified Input Instances // Navarro G., Bertossi L., Kohayakwa Y. International Federation for Information Processing. Fourth IFIP International Conference on Theoretical Computer Science – TCS 2006. – Springer. – Boston. – 2006. – P. 251–270.

[39] Bräsel H., Harborth M., Tautenhahn T., Willenius P. On the hardness of the classical job shop problem // Annals of Operations Research. – 1999. – V. 92. – P. 265–279.

[40] Bräsel H., Harborth M., Tautenhahn T., Willenius P. On the set of solutions of the open shop problem // Annals of Operations Research. – 1999. – V. 92. – P. 241–263.

[41] Bräsel H., Harborth M., Willenius P. Isomorphism for digraphs and sequences of shop scheduling problems // Journal of Combinatorial Mathematics and Combinatorial Computing. – 2001. – V. 37. – P. 115–128.

[42] Bräsel H., Kleinau M. On the number of feasible schedules of the open-shop problem – an application of special latin rectangles // Optimization. – 1992. – V. 23. – P. 251–260.

[43] Bräsel H., Kleinau M. New steps in the amazing world of sequences and schedules // Mathematical Methods of Operations Research. – 1996. – V. 43. – P. 195–214.

[44] Bräsel H., Sotskov Yu.N., Werner F. Stability of a schedule minimizing mean flow time // Mathematical and Computer Modelling. – 1996. – V. 24. – № 10. – P. 39–53.

[45] Braun O., Lai T.-C., Schmidt G., Sotskov Yu.N. Stability of Johnson's schedule with respect to limited machine availability // International Journal of Production Research. – 2002. – V. 40. – № 17. – P. 4381–4400.

[46] Braun O., Leshchenko N.M., Sotskov Yu.N. Optimality of Jackson's permutation with respect to limited machine availability // International Transactions in Operations Research. – 2006. – V. 13. – P. 59–74.

[47] Braun O., Sotskov Yu.N. Financial scheduling: feasibility and optimality // Preceedings of the POMS 20th Annual Conference. – 1-35. Orlando, Florida, USA. – May 1 - 4, 2009.

[48] Briand C., La H.T., Erschler J. A new sufficient condition of optimality for the two-machine flowshop problem // European Journal of Operational Research. – 2006. – V. 169. – P. 712–722.

[49] Briand C., La H.T., Erschler J. A robust approach for the single machine scheduling problem // Journal of Scheduling. – 2007. – V. 10. – P. 209–221.

[50] Brucker P. An efficient algorithm for the job-shop problem with two jobs // Computing. – 1988. – V. 40. – P. 353–359.

[51] Brucker P. Scheduling Algorithms // Springer, Berlin. – 1995.

[52] Brucker P., Drexl A., Möhring R., Neumann K., E. Pesch Resource-constrained project scheduling: Notation, classification, models and methods // European Journal of Operational Research. – 1999. – V. 112. – P. 3–41.

[53] Brucker P., Kravchenko S.A., Sotskov Yu.N. Preemptive job-shop scheduling problems with fixed number of jobs // Mathematical Methods of Operations Research. – 1999. – V. 49. – P. 41–76.

[54] Brucker P., Sotskov Yu.N., Werner F. Complexity of shop-scheduling problems with fixed number of jobs: a survey // Mathematical Methods of Operations Research. – 2007. – V. 65. – P. 461–481.

[55] Bukchin J., Tzur M. Design of flexible assembly line to minimize equipment cost // IIE Transactions. – 2000. – V. 32. – P. 585–598.

[56] Bukhtoyarov S.E., Emelichev V.A. On stability of an optimal situation in a finite cooperative game with a parametric concept of equilibrium (from lexicographic optimality to Nash equilibrium) // Computer Science Journal of Moldova. – 2004. – V. 12. – № 3. – P. 371–380.

[57] Bukhtoyarov S.E., Emelichev V.A. Measure of stability for a finite cooperative game with a parametric optimality principle (from Pareto to Nash) // Computational Mathematics and Mathematical Physics. – 2006. – V. 46. – № 7. – P. 1193–1199.

[58] Bukhtoyarov S.E., Emelichev V.A., Stepanishina Yu.V. Stability of discrete vector problems with the parametric principle of optimality // Cybernetics and Systems Analysis. – 2003. – V. 39. – № 4. – P. 604–614.

[59] Buxey G. Production scheduling: Theory and practice // European Journal of Operational Research. – 1989. – V. 39. – № 1. – P. 17–31.

[60] Cai X., Tu F.S. Scheduling jobs with random processing times on a single machine subject to stochastic breakdowns to minimize early-tardy penalties // Naval Research Logistics. – 1996. – V. 43. – P. 1127–1146.

[61] Calhoun K.M., Deckro R.F., Moore J.T., Chrissis J.W., Van Hove J.C. Planning and re-planning in project and production scheduling // OMEGA – International Journal of Management Science. – 2002. – V. 30. – P. 155–170.

[62] Carlier J., Pinson E. An algorithm for solving the job shop problem // Management Science. – 1989. – V. 35. – P. 164–176.

[63] Chakravarti N., Wagelmans A.P.M. Calculation of stability radii for combinatorial optimization problems // Operations Research Letters. – 1998. – V. 23. – № 1. – P. 1–7.

[64] Chand S., Traub R., Uzsoy R. Single-machine scheduling with dynamic arrivals: Decomposition results and an improved algorithm // Naval Research Logistics. – 1996. – V. 43. – P. 709–716.

[65] Che A., Chu C. Cyclic hoist scheduling in large real-life electroplating lines // OR Spectrum. – 2007. – V. 29. – P. 445–470.

[66] Chen B., Vestjens A.P.A. Scheduling on identical machines: How good is LPT in an on-line setting? // Operations Research Letters. – 1997. – V. 21. – P. 165–169.

[67] Chen H., Chu C., Proth J.M. Cyclic scheduling of a hoist with time window constraints // IEEE Transactions on Robotics and Automation. – 1998. – V. 14. – P. 144–152.

[68] Chen K.J., Ji P. A genetic algorithm for dynamic advanced planning and scheduling (DAPS) with a frozen interval // Expert Systems with Applications. – 2007. – V. 33. – № 4. – P. 1004–1010.

[69] Chen X., Hu J., Hu X. A new model for path planning with interval data // Computers & Operations Research. – 2009. – V. 36. – P. 1893–1899.

[70] Cheng T.C.E., Janiak A., Kovalyov M.Y. Bicriterion single machine scheduling with resource dependent processing time // SIAM Journal on Computing. – 1998. – V. 8. – № 2. – P. 617–630.

[71] Cheng T.C.E., Kovalyov M.Y., Shakhlevich N.V. Scheduling with controllable release dates and processing times: Makespan minimization // European Journal of Operational Research. – 2006. – V. 175. – P. 751–768.

[72] Cheng T.C.E., Kovalyov M.Y., Shakhlevich N.V. Scheduling with controllable release dates and processing times: Total completion time minimization // European Journal of Operational Research. – 2006. – V. 175. – P. 769–781.

[73] Cheng T.C.E., Kovalyov M.Y., Tuzikov A.V. Single machine group scheduling with two ordered criteria // Journal of the Operations Research Society. – 1996. – V. 47. – P. 315–320.

[74] Cheng T.C.E., Shakhlevich N. Single Machine Scheduling of Unit-Time Jobs with Controllable Release Dates // Working Paper No. 01/98-9, The Hong Kong Polytechnic University. – 1998.

[75] Cheng T.C.E., Shakhlevich N.V. Proportionate flow shop with controllable processing times // Journal of Scheduling. – 1999. – V. 27. – P. 253–265.

[76] Cheng T.C.E., Shakhlevich N.V. Single machine scheduling of unit-time jobs with controllable release dates // Journal of Global Optimization. – 2003. – V. 27. – P. 293–311.

[77] Cheng T.C.E., Wang G. Two-machine flowshop scheduling with consecutive availability constraints // Information Processing Letters. – 1999. – V. 71. – P. 49–54.

[78] Cheng T.C.E., Wang G. An improved heuristic for two-machine flowshop scheduling with an availability constraint // Operations Research Letters. – 2000. – V. 26. – P. 223–229.

[79] Cheng T.C.E., Wang G. Single-machine scheduling with learning effect consideration // Annals of Operations Research. – 2000. – V. 98. – P. 273–290.

[80] Chou M.C., Queyranne M., Simchi-Levi D. The asymptotic performance ratio of an on-line algorithm for uniform parallel machine scheduling with release dates // Mathematical Programming. Series A. – 2006. – V. 106. – P. 137–157.

[81] Chretienne P., Coffman E.G., Lenstra J.K., Liu Z. (editors) Scheduling Theory and its Applications // John Wiley & Sons. – 1995.

[82] Chrysslouris G., Dicke K., Lee M. An approach to real-time flexible scheduling // International Journal of Flexible Manufacturing Systems. – 1994. – V. 6. – P. 235–253.

[83] Chu C., Gordon V. TWK due date determination and scheduling model: NP-hardness and polynomially solvable case // Proth J.-M., Tanaev V. Proceedings of the International Workshop on Discrete Optimization Methods an Scheduling and Computer-aided Design. – P. 99–101. – Minsk, Belarus. – September 5–6, 2000.

[84] Conway R.W., Maxwell W.L., Miller L.W. Theory of Scheduling // Addison-Wesley, Reading, MA, USA. – 1967.

[85] Correa J.R., Wagner M.R. LP-based online scheduling: from single to parallel machines // Mathematical Programming. Series A. – 2009. – V. 119. – P. 109–136.

[86] Daniels R.L., Carrillo J.E. Beta-robust scheduling for single-machine systems with uncertain processing times // IIE Transactions. – 1997. – V. 29. – P. 977–985.

[87] Daniels R.L., Hoopes B.J., Mazzola J.B. Scheduling parallel manufacturing cells with resource flexibility // Management Science. – 1996. – V. 42. – Nº 9. – P. 1260–1276.

[88] Daniels R.L., Kouvelis P. Robust scheduling to hedge against processing time uncertainty in single stage production // Management Science. – 1995. – V. 41. – Nº 2. – P. 363–376.

[89] Davenport A.J., Beck J.C. A survey for techniques for scheduling with uncertainty // Unpublished. – 2000 Note: available at http://www.eil.toronto.ca/profiles/chris/chris.papers.html.

[90] Davis E.W. Project scheduling under resource constraints - historical review and categorization of procedures // AIIE Transactions. – 1973. – V. 5. – Nº 3. – P. 297–313.

[91] Davis W.J., Jones A.T. A real-time production scheduler for a stochastic manufacturing environment // International Journal of Production Research. – 1988. – V. 1. – Nº 2. – P. 101–112.

[92] Dempe S., Müller E. Stability analysis for a special interval cutting stock problem // European Journal of Operational Research. – 1995. – V. 87. – P. 188–199.

[93] Dhamala T.N. On the potentially optimal solutions of classical shop scheduling problems // International Journal of Operations Research. – 2007. – V. 4. – Nº 2. – P. 80–89.

[94] Dijkstra E.W. A note on two problems in connection with graphs // Numerische Mathematik. – 1959. – V. 1. – P. 269–271.

[95] Dobin B. Determining the optimal sequences and the distributional properties of their completion times in stochastic flow shops // Computers & Operations Research. – 1996. – V. 23. – Nº 9. – P. 829–843.

[96] Dolgui A., Guschinski N., Harrath Y., Levin G. Une approche de programmation linaire pour la conception des lignes de transfert // European Journal of Automated Systems (APII-JESA). – 2002. – V. 36. – Nº 1. – P. 11–33.

[97] Dolgui A., Ould-Louly M.-A. A model for supply planning under lead time uncertainty // International Journal of Production Economics. – 2002. – V. 78. – P. 145–152.

[98] Dolgui A., Prodhon C. Supply planning under uncertainties in MRP environment: A state of the art // Annual Reviews in Control. – 2007. – V. 31. – P. 269–279.

[99] Drotos M., Erdos G., Kis T. Computing lower and upper bounds for a large-scale industrial job shop scheduling problem // European Journal of Operational Research. – 2009. – V. 197. – P. 296–306.

[100] Dudek R.A., Panwalker S.S., M.L. Smith The lessons of flow shop scheduling research // Operations Research. – 1992. – V. 40. – № 1. – P. 7–13.

[101] Dumitru V., Luban F. Membership functions, some mathematical programming models and production scheduling // Fuzzy Sets and Systems. – 1982. – V. 8. – P. 19–33.

[102] Easa S.M. Resource leveling in construction by optimization // Journal of Construction Engineering and Management. – 1989. – V. 115. – P. 302–316.

[103] Edmonds J. Scheduling in the dark // Theoretical Computer Science. – 2000. – V. 235. – P. 109–141.

[104] Edmonds J., Chinn D.D., Brecht T., Deng X. Non-clairvoyant multiprocessor scheduling of jobs with changing execution characteristics // Journal of Scheduling. – 2003. – V. 6. – P. 231–250.

[105] Edmonds J., Pruhs K. Multicast pull scheduling: When fairness is fine // Algorithmica. – 2003. – V. 36. – P. 315–330.

[106] Egorova N.G., Matsveichuk N.M., Sotskov Yu. N. Selection of an optimal order for processing jobs on two machines during scheduling realization // Vesti Akad. Navuk BSSR, Ser. Fiz.-Mat. Navuk. – 2006. – V. 5. – P. 20–24 (In Russian).

[107] Elmaghraby S.E. On criticality and sensitivity in activity networks // European Journal of Operational Research. – 2000. – V. 127. – P. 220–238.

[108] Elmaghraby S.E., Thoney K.A. Two-machine flowshop problem with arbitrary processing time distributions // IIE Transactions. – 1999. – V. 31. – № 5. – P. 467–477.

[109] Emelichev V.A., Girlih E., Nikulin Y.V., Podkopaev D.P. Stability and regularization of vector problem of integer linear programming // Optimization. – 2002. – V. 51. – № 4. – P. 645–676.

[110] Emelichev V.A., Kravtsov M.K. About the stability in trajectory problems of vectorial optimization // Cybernetics and System Analysis. – 1995. – V. 4. – P. 137–143 (In Russian).

[111] Emelichev V.A., Kravtsov M.K., Yanyshkevich O.A. Conditions for Pareto optimality for one vector discrete problem on a system of subsets // U.S.S.R. Computational Mathematics and Mathematical Physics. – 1995. – V. 35. – № 11. – P. 1641–1652.

[112] Emelichev V.A., Kuz'min K., Nikulin Y.V. Stability analysis of the Pareto optimal solutions for some vector Boolean optimaization problem // Optimization. – 2005. – V. 54. – № 6. – P. 545–561.

[113] Emelichev V.A., Kuz'min K.G. Stability radius for a strictly efficient solution to a vector minimization problem for threshold functions in l_1 metric // Cybernetics and Systems Analysis. – 2004. – V. 40. – № 3. – P. 62–67.

[114] Emelichev V.A., Kuz'min K.G. Stability radius of a lexicographic optimum of a vector problem of Boolean programming // Cybernetics and Systems Analysis. – 2004. – V. 41. – № 2. – P. 71–81.

[115] Emelichev V.A., Kuz'min K.G. Stability radius of an efficient solution of a vector problem of integer linear programming in the Golder metric // Cybernetics and Systems Analysis. – 2006. – V. 42. – № 4. – P. 175–181.

[116] Emelichev V.A., Nikulin Yu.V. Numerical measure of strong stability and strong quasistability in the vector problem of integer linear programming // Computer Science Journal of Moldova. – 1999. – V. 7. – № 1. – P. 105–117.

[117] Emelichev V.A., Podkopaev D.P. On a quantitative measure of the stability of a vector problem of integer programming // U.S.S.R. Computational Mathematics and Mathamatical Physics. – 1998. – V. 38. – № 11. – P. 1727–1731 (Translated from Russian).

[118] Emelichev V.A., Yanyshkevich O.A. Regularization of a lexicographic vector problem of integer programming // Cybernetics and Systems Analysis. – 1999. – V. 35. – № 6. – P. 951–955.

[119] Fisher H., Thompson M.L. Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules // Muth J.F., Thompson G.L. Industrial Scheduling. – Prentice-Hall. – Englewood Cliffs. – 1963. – P. 225–251.

[120] Foley R.D., Suresh S. Stochastically minimizing the makespan in flow shops // Naval Research Logistics Quarterly. – 1984. – V. 31. – P. 551–557.

[121] Forst F.G. Minimizing total expected costs in the two machine stochastic flow shop // Operations Research Letters. – 1983. – V. 2. – P. 58–61.

[122] French S. Sequencing and Scheduling. An Introduction to the Mathematics of the Job-Shop // Ellis Horwood John, Chichester, UK. – 1987.

[123] Gan H.-S., Wirth A. Heuristic stability: A permutation disarray measure // Computers & Operations Research. – 2007. – V. 34. – P. 3187–3208.

[124] Gantt H.L. Efficiency and democracy // Transactions of American Society of Mechanical Engineering. – 1919. – V. 40. – P. 799–808.

[125] Garey M.R., Johnson D.S. Computers and Intractability - A Guide to the Theory of NP-Completeness // W.H. Freeman and Company, San Francisco, USA. – 1979.

[126] Garey M.R., Johnson D.S., Sethi R. The complexity of flowshop and jobshop scheduling // Mathematics of Operations Research. – 1976. – V. 1. – P. 117–129.

[127] Gargeya V.B., Deane R.H. Scheduling research in multiple resource constrained job shops: A review and critique // International Journal of Production Research. – 1996. – V. 34. – № 8. – P. 2077–2097.

[128] Gawiejnomicz S. A note on scheduling on a single processor with speed dependent on a number of executed jobs // Information Processing Letters. – 1996. – V. 57. – P. 297–300.

[129] Gawiejnomicz S. Time-Dependent Scheduling // Springer, Berlin, New-York. – Monographs in theoretical computer science. – 2008.

[130] Gen M., Tsujimura Y., Li Y. Fuzzy assembly line balancing using genetic algorithms // Computers & Industrial Engineering. – 1996. – V. 31. – №º 3/4. – P. 631–634.

[131] Geoffrion A.M., Nauss R. Parametric and postoptimality analysis in integer linear programming // Management Science. – 1977. – V. 23. – P. 453–466.

[132] Giffler B., Thompson G.L. Algorithms for solving production-scheduling problems // Operations Research. – 1960. – V. 8. – №º 4. – P. 487–503.

[133] Gill A. Determining loading dock requirements in production-distribution facilities // Computers & Industrial Engineering. – 2009. – V. 57. – P. 161–168.

[134] Gordeev E.N. Algorithms of polynomial complexity for computing the stability radius in two classes of trajectory problems // U.S.S.R. Computational Mathematics and Mathematical Physics. – 1987. – V. 27. – №º 4. – P. 14–20.

[135] Gordeev E.N. Solution stability of the shortest path problem // Discrete Mathematics. – 1989. – V. 1. – №º 3. – P. 45–56 (In Russian).

[136] Gordeev E.N., Leontev V.K. Stability in bottleneck problems // U.S.S.R. Computational Mathematics and Mathematical Physics. – 1980. – V. 20. – №º 4. – P. 275–280.

[137] Gordeev E.N., Leontev V.K. The complexity of the tabulation of trajectory problems // U.S.S.R. Computational Mathematics and Mathematical Physics. – 1985. – V. 25. – №º 4. – P. 199–201.

[138] Gordeev E.N., Leontev V.K. A general approach to investigating the stability of solutions of problems of discrete optimization // U.S.S.R. Computational Mathematics and Mathematical Physics. – 1996. – V. 36. – №º 1. – P. 53–58.

[139] Gordeev E.N., Leontev V.K., Sigal I.Ch. Computational algorithms for finding stability radius in choice problems // U.S.S.R. Computational Mathematics and Mathematical Physics. – 1983. – V. 23. – №º 4. – P. 128–132.

[140] Gordon V.S., Potts C.N., Strusevich V.A., Whitehead J.D. Single machine scheduling models with deterioration and learning: handling precedence constraints via priority generation // Journal of Scheduling. – 2008. – V. 11. – P. 357–370.

[141] Gordon V.S., Tarasevich A.A. A note: Common due date assignment for a single machine scheduling with the rate-modifying activity // Computers & Operations Research. – 2009. – V. 36. – P. 325–328.

[142] Goren S., Sabuncuoglu I. Robustness and stability measures for scheduling: single-machine environment // IIE Transactions. – 2008. – V. 40. – P. 66–83.

[143] Grabot B., Geneste L. Dispatching rules in scheduling: A fuzzy approach // International Journal of Production Research. – 1994. – V. 32. – №º 4. – P. 903–915.

[144] Graves S.C. A review of production scheduling // Operations Research. – 1981. – V. 29. – №º 4. – P. 646–675.

[145] Greenberg H.J. A bibliography for the development of an intelligent mathematical programming system // Annals of Operations Research. – 1996. – V. 65. – P. 55–90 Note: also available at http://orcs.bus.okstate.edu/itorms.

[146] Greenberg H.J. An annotated bibliography for post-solution analysis in mixed integer programming and combinatorial optimization // Woodruff D.L. Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search. – Kluwer Academic Publishers, Boston, MA. – 1998. – P. 97–148.

[147] Guo Z.X., Wong W.K., Leung S.Y.S, J.T. Fan Intelligent production control decision support system for flexible assembly lines // Expert Systems with Applications. – 2009. – V. 36. – P. 4268–4277.

[148] Gupta J.N.D. An excursion in scheduling theory: an overview of scheduling research in the twentieth century // Production Planning & Control. – 2002. – V. 13. – №. 2. – P. 105–116.

[149] Gupta J.N.D., Hennig K., Werner F. Local search heuristics for the two-stage flowshop problem // Computers & Operations Research. – 2002. – V. 29. – P. 113–149.

[150] Gupta J.N.D., Krüger K., Lauff V., Werner F., Sotskov Yu. N. Heuristics for hybrid flow shops with controllable processing times and assignable due dates // Computers & Operations Research. – 2002. – V. 29. – P. 1417–1439.

[151] Gupta J.N.D., Nepalli V.R., Werner F. Minimizing total flow time in a two-machine flowshop problem with secondary criterion // International Journal of Production Economics. – 2001. – V. 69. – №. 3. – P. 323–338.

[152] Gupta J.N.D., Stafford Jr E.F. Flowshop scheduling research after five decades // European Journal of Operational Research. – 2006. – V. 169. – P. 699–711.

[153] Gupta J.N.D., Werner F., Wulkenhaar G. Two-machine open shop scheduling with secondary criterion // International Transactions in Operations Research. – 2003. – V. 10. – №. 3. – P. 267–294.

[154] Gusfield D. A note on arc tolerances in sparse shortest-path and network flow problems // Networks. – 1983. – V. 13. – №. 2. – P. 191–196.

[155] Hall N.G., Posner M.E. Sensitivity analysis for scheduling problems // Journal of Scheduling. – 2004. – V. 7. – P. 49–83.

[156] Hall N.G., Potts C.N. Rescheduling for new orders // Operations Research. – 2004. – V. 52. – №. 3. – P. 440–453.

[157] Hardgrave W.W., Nemhauser G. A geometric model and graphical algorithm for a sequencing problem // Operations Research. – 1963. – V. 11. – P. 889–900.

[158] Hartmann S., Kolisch R. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem // European Journal of Operational Research. – 2000. – V. 127. – P. 394–407.

[159] Herroelen W. Project scheduling - Theory and practice // Production and Operations Management. – 2005. – V. 14. – №. 4. – P. 413–432.

[160] Herroelen W., De Reyck B., Demeulemeester E. Resource-constrained project scheduling: A survey of recent developments // Computers & Operations Research. – 1998. – V. 25. – P. 279–302.

[161] Herroelen W., Leus R. Robust and reactive project scheduling: a review and classification procedures // International Journal of Production Research. – 2004. – V. 42. – №8. – P. 1599–1620.

[162] Herroelen W., Leus R. Project scheduling under uncertainty: Survey and research potentials // European Journal of Operational Research. – 2005. – V. 165. – P. 289–306.

[163] Hoesel S., Wagelmans A. Sensitivity analysis of the economic lotsizing problem // Discrete Applied Mathematics. – 1993. – V. 45. – P. 291–312.

[164] Hoesel S., Wagelmans A. On the complexity of postoptimality analysis of 0/1 programs // Discrete Applied Mathematics. – 1999. – V. 91. – P. 251–263.

[165] Ishii H., Masuda T., Nishida T. Two machine mixed shop scheduling problems with controllable machine speeds // Discrete Applied Mathematics. – 1987. – V. 17. – P. 29–38.

[166] Ishii H., Nishida T. Two machine open shop scheduling problem with controllable machine speeds // Journal of the Operations Research Society of Japan. – 1986. – V. 29. – P. 123–131.

[167] Ishii H., Tada M. Single machine scheduling problem with fuzzy precedence relation // European Journal of Operational Research. – 1995. – V. 87. – P. 284–288.

[168] Jackson J.R. An extension of Johnson's results on job lot scheduling // Naval Research Logistics Quarterly. – 1956. – V. 3. – №3. – P. 201–203.

[169] Jain A.S., Meeran S. Deterministic job-shop scheduling: Past, present and future // European Journal of Operational Research. – 1999. – V. 113. – P. 390–434.

[170] James R.J.W., Buchnan J.T. Robustness of single machine scheduling problems to earliness and tardeness penalty errors // Annals of Operations Research. – 1998. – V. 76. – P. 219–232.

[171] Jang W. Dynamic scheduling of stochastic jobs on a single machine // European Journal of Operational Research. – 2002. – V. 138. – P. 518–530.

[172] Janiak A. General flow-shop scheduling with resource constraints // International Journal of Production Research. – 1988. – V. 26. – P. 125–138.

[173] Jansen K., Mastrolilli M., Solis-Oba R. Approximation schemes for job shop scheduling problems with controllable processing times // European Journal of Operational Research. – 2005. – V. 167. – P. 297–319.

[174] Jeng A.A.K., Lin B.M.T. Minimizing the total completion time in single machine scheduling with step-deteriorating job // Computers & Operations Research. – 2005. – V. 32. – P. 521–536.

[175] Jensen M. T. Generating robust and flexible job shop schedules using genetic algorithms // IEEE Transactions on Evolutionary Computation. – 2003. – V. 7. – №3. – P. 275–288.

[176] Jia C. Minimizing variation in stochastic flow shop // Operations Research Letters. – 1998. – V. 23. – P. 109–111.

[177] Johnson S.M. Optimal two and three stage production schedules with set up times included // Naval Research Logistics Quarterly. – 1954. – V. 1. – №. 1. – P. 61–68.

[178] Kamburowski J. Stochastically minimizing the makespan in two-machine flow shops without blocking // European Journal of Operational Research. – 1999. – V. 112. – P. 304–309.

[179] Kamburowski J. On three-machine flow shops with random job processing times // European Journal of Operational Research. – 2000. – V. 125. – P. 440–449.

[180] Kasperski A. Minimizing maximal regret in the single machine sequencing problem with maximum lateness criterion // Operations Research Letters. – 2005. – V. 33. – P. 431–436.

[181] Kayan R.K., Akturk M.S. A new bounding mechanizm for the CNC machine scheduling problems with controllable processing times // European Journal of Operational Research. – 2005. – V. 167. – P. 624–643.

[182] Kim J.-H., Chwa K.-Y. Non-clairvoyant scheduling for weighted flow time // Information Processing Letters. – 2003. – V. 87. – P. 31–37.

[183] Kimms A. Multi-level lot sizing and scheduling // Physica Verlag, Heidelberg, Germany. – 1997.

[184] Kimms A. Stability measures for rolling schedules with applications to capacity expansion planning, master production scheduling, and lot sizing // OMEGA – International Journal of Management Science. – 1998. – V. 26. – №. 3. – P. 355–366.

[185] Kis T. A branch-and-cut algorithm for scheduling of projects with variable-intensity activities // Mathematical Programming. Series A. – 2005. – V. 103. – P. 515–539.

[186] Kolen A.W.H., Rinnooy Kan A.H.G., Hoesel C.P.M., Wagelmans A.P.M. Sensitivity analysis of list scheduling algorithms // Discrete Applied Mathematics. – 1994. – V. 55. – P. 145–162.

[187] Kouvelis P., Daniels R.L., Vairaktarakis G. Robust scheduling of a two-machine flow shop with uncertain processing times // IIE Transactions. – 2000. – V. 32. – P. 421–432.

[188] Kouvelis P., Kurawarwala A.A., Gutierrez G.J. Algorithms for robust single- and multiple-period layout planning for manufacturing systems // European Journal of Operational Research. – 1992. – V. 63. – P. 287–303.

[189] Kouvelis P., Yu G. Robust Discrete Optimization and its Application // Kluwer Academic Publishers, Boston, The USA. – 1997.

[190] Kovalyov M.Y., Sotskov Yu. N. ϵ-approximate solution stability of Boolean linear form minimization // Vesti Akad. Navuk BSSR, Ser. Fiz.-Mat. Navuk. – 1990. – №. 2. – P. 111–116 (In Russian).

[191] Kravchenko S.A., Sotskov Yu.N. Makespan optimal schedule with infinitely large stability radius // Vesti Akad. Navuk BSSR, Ser. Fiz.-Mat. Navuk. – 1993. – № 4. – P. 85–91 (In Russian).

[192] Kravchenko S.A., Sotskov Yu.N., Werner F. Optimal schedules with infinitely large stability radius // Optimization. – 1995. – V. 33. – P. 271–280.

[193] Kreipl S., Pinedo M. Planning and scheduling in supply chains: An overview of issues in practice // Production and Operations Management. – 2004. – V. 13. – № 1. – P. 77–92.

[194] Ku P.S., Niu S.C. On Johnson's two-machine flow-shop with random processing times // Operations Research. – 1986. – V. 34. – P. 130–136.

[195] Kubiak W., Blazewicz J., Formanowicz P., Breit J., Schmidt G. Two-machine flow-shops with limited machine availability // European Journal of Operational Research. – 2002. – V. 136. – P. 528–540.

[196] Kuroda M., Wang Z. Fuzzy job shop scheduling // International Journal of Production Economics. – 1996. – V. 44. – P. 45–51.

[197] Kutanoglu E., Wu D. Improving scheduling robustness via preprocessing and dynamic adaptation // IIE Transactions. – 2004. – V. 36. – P. 1107–1124.

[198] Kyparisis G., Koulamas C. Open shop scheduling with makespan and total completion time criteria // Computers & Operations Research. – 2000. – V. 27. – P. 15–27.

[199] Lahlou C., Dauzere-Peres S. Single-machine scheduling with time window-dependent processing times // Journal of the Operations Research Society. – 2006. – V. 57. – P. 133–139.

[200] Lai T.-C., Sotskov Yu.N. Sequencing with uncertain numerical data for makespan minimization // Journal of the Operations Research Society. – 1999. – V. 50. – P. 230–243.

[201] Lai T.-C., Sotskov Yu.N., Sotskova N.Yu., Werner F. Optimal makespan scheduling with given bounds of processing times // Mathematical and Computer Modelling. – 1997. – V. 26. – № 3. – P. 67–86.

[202] Lai T.-C., Sotskov Yu.N., Sotskova N.Yu., Werner F. Mean flow time minimization with uncertain processing times // Otto-von-Guericke-Universität, FMA, Preprint No. 15/98, Magdeburg, Germany. – 1998.

[203] Lai T.-C., Sotskov Yu.N., Sotskova N.Yu., Werner F. Mean flow time minimization with given bounds of processing times // European Journal of Operational Research. – 2004. – V. 159. – P. 558–573.

[204] Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B. Sequencing and Scheduling: Algorithms and Complexity // Graves S.S., Kan A.H.G. Rinnooy, Zipkin P. Handbooks in Operations Research and Management Science. – Logistics of Production and Inventory. – North-Holland, New York. – 1993. – P. 445–522.

[205] Lebedev V., Averbakh I. Complexity of minimizing the total flow time with interval data and minmax regret criterion // Discrete Applied Mathematics. – 2006. – V. 154. – P. 2167–2177.

[206] Lee C.Y. Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint // Operations Research Letters. – 1997. – V. 20. – P. 129–139.

[207] Lee C.Y., Lei L., Pinedo M. Current trends in deterministic scheduling // Annals of Operations Research. – 1997. – V. 70. – P. 1–41.

[208] Lee H.F., Johnson R.V. A line-balancing strategy for designing flexible assembly systems // International Journal of Flexible Manufacturing Systems. – 1991. – V. 3. – P. 91–120.

[209] Lei L., Wang T.J. The minimum common-cycle algorithm for cycle scheduling of two hoists with time window constraints // Management Science. – 1991. – V. 37. – P. 1629–1639.

[210] Lenstra J.K. The mystical power of twoness: In memoriam Eugene L. Lawler // Journal of Scheduling. – 1998. – V. 1. – P. 3–14.

[211] Lenstra J.K., Rinnooy Kan A.H.G. Computational complexity of discrete optimization problems // Annals of Discrete Mathematics. – 1979. – V. 4. – P. 121–140.

[212] Lenstra J.K., Rinnooy Kan A.H.G., Brucker P. Complexity of machine scheduling problems // Annals of Discrete Mathematics. – 1977. – V. 1. – P. 343–362.

[213] Leon V.J., Wu S.D., Storer R.H. A game-theoretic control approach for job shops in the presence of disruptions // International Journal of Production Research. – 1994. – V. 32. – №. 6. – P. 1451–1476.

[214] Leon V.J., Wu S.D., Storer R.H. Robustness measures and robust scheduling for job shops // IIE Transactions. – 1994. – V. 26. – P. 32–43.

[215] Leontev V.K. The stability of the traveling salesman problem // U.S.S.R. Computational Mathematics and Mathematical Physics. – 1975. – V. 15. – №. 5. – P. 199–213.

[216] Leshchenko N.M., Sotskov Yu. N. Minimizing makespan for processing conflict jobs with non-fixed proceesing times // Vesti Akad. Navuk BSSR, Ser. Fiz.-Mat. Navuk. – 2006. – V. 4. – P. 103–110 (In Russian).

[217] Leshchenko N.M., Sotskov Yu. N. Realization of an optimal schedule for the two-machine flow-shop with interval job processing times // International Journal: Information, Theories & Applications. – 2007. – V. 14. – №. 2. – P. 182–189.

[218] Leshchenko N.M., Sotskov Yu.N. Algorithm for selection of an optimal permutation during schedule realization // Proceedings of the Second Conference "Tanaev's readings". – Minsk, Belarus. – 2005. – P. 74-78 (In Russian).

[219] Leshchenko N.M., Sotskov Yu.N. Two-machine minimum-length shop-scheduling problems with uncertain processing times // Proceedings of XI-th International Conference "Knowledge-Dialogue-Solution". – Varna, Bulgaria. – 2005. – P. 375-381.

[220] Leshchenko N.M., Sotskov Yu.N. A dominant schedule for the uncertain two-machine shop-scheduling problem // Proceedings of XII-th International Conference "Knowledge-Dialogue-Solution". – Varna, Bulgaria. – 2006. – P. 291-297.

[221] Leus R., Herroelen W. Stability and resource allocation in project planning // IIE Transactions. – 2004. – V. 36. – P. 667–682.

[222] Leus R., Herroelen W. The complexity of machine scheduling for stability with a single disrupted job // Operations Research Letters. – 2005. – V. 33. – P. 151–156.

[223] Leus R., Herroelen W. Scheduling for stability in single-machine production systems // Journal of Scheduling. – 2007. – V. 10. – P. 223–235.

[224] Levin A., Woeginger G.J. The constrained minimum weighted sum of job completion times problem // Mathematical Programming. Series A. – 2006. – V. 108. – P. 115–126.

[225] Li H., Li Z., Li L.X., Hu B. A production rescheduling expert simulation system // European Journal of Operational Research. – 2000. – V. 124. – P. 283–293.

[226] Li W., Cao J. Stochastic scheduling on a single machine subject to multiple breakdowns according to different probabilities // Operations Research Letters. – 1995. – V. 18. – P. 81–91.

[227] Libura M. Optimality conditions and sensitivity analysis for combinatorial optimization problems. // Control and Cybernetics. – 1996. – V. 25. – Nº 6. – P. 1165–1180.

[228] Libura M. On accuracy of solutions for discrete optimization problems with perturbed coefficients of the objective function // Annals of Operations Research. – 1999. – V. 86. – P. 53–62.

[229] Libura M., Nikulin Yu. Stability and accuracy functions in multicriteria linear combinatorial optimization problems // Annals of Operations Research. – 2006. – V. 147. – P. 255–267.

[230] Libura M., Poort E.S., Sierksma G., Veen J.A.A. Sensitivity Analysis Based on k-best Solutions of the Traveling Salesman Problems // Technical Report 96A14. – Research Institute Systems, Organizations and Management, University of Groningen. – The Netherlands, Groningen. – 1996.

[231] Libura M., Poort E.S., Sierksma G., Veen J.A.A. Stability aspects of the traveling salesman problem based on k-best solutions // Discrete Applied Mathematics. – 1998. – V. 87. – P. 159–185.

[232] Lin Y., Wang X. Necessary and sufficient conditions of optimality for some classical scheduling problems // European Journal of Operational Research. – 2007. – V. 176. – P. 809–818.

[233] Liu J., Jiang Y., Zhou Z. Cyclic scheduling of a single hoist in extended electroplating lines: a comprehensive integer programming solution // IIE Transactions. – 2002. – V. 34. – P. 905–914.

[234] Liu L., Gu H.-Y., Xi Y.-G. Robust and stable scheduling of a single machine with random machine breakdowns // International Journal of Advanced Manufacturing Technology. – 2007. – V. 124. – P. 283–293.

[235] MacCarthy B.L., Liu J. Addressing the gap in scheduling reserach: A review of optimization and heuristic methods in production research // International Journal of Production Research. – 1993. – V. 31. – P. 59–79.

[236] Manier M.A., Bloch C. A classification for hoist scheduling problems // International Journal of Flexible Manufacturing Systems. – 2003. – V. 15. – № 1. – P. 37–55.

[237] Matsveichuk N.M., Sotskov Yu.N., Egorova N.M., Lai T.-C. Schedule execution for two-machine flow-shop with interval processing times // Mathematical and Computer Modelling. – 2009. – V. 49. – № 5-6. – P. 991–1011.

[238] Matsveichuk N.M., Sotskov Yu.N., Werner F. Partial job order for solving the two-machine flow-shop minimum-length problem with uncertain processing times // Proceedings of the 13-th IFAC Symposium on Information Control Problems in Manufacturing, IFAC'2009. – P. 1500-1505. – Moscow, Russia. – June 3–5 2009.

[239] McCahon C. Using PERT as an approximation of fuzzy project-network analysis // IEEE Transactions on Engineering Management. – 1993. – V. 40. – № 2. – P. 146–153.

[240] McKay K.N. Unifying the theory and practice of production scheduling // Journal of Manufacturing Systems. – 1999. – V. 18. – № 4. – P. 241–255.

[241] McKay K.N., Safayeni F.R., Buzacott J.A. Job-shop scheduling theory: What is relevant? // Interfaces. – 1988. – V. 18. – № 4. – P. 84–90.

[242] McKay K.N., Safayeni F.R., Buzacott J.A. 'Common sense' realities of planning and scheduling in printed circuit board production // International Journal of Production Research. – 1995. – V. 33. – № 6. – P. 1587–1603.

[243] McKay K.N., Safayeni F.R., Buzacott J.A. A review of hierarchical production planning and its applicability for modern manufacturing // Production Planning & Control. – 1995. – V. 6. – № 5. – P. 384–394.

[244] Melnikov O.I. Properties of the optimal schedules for the Bellman-Johnson problem with two machines // Vesti Akad. Navuk BSSR, Ser. Fiz.-Mat. Navuk. – 1976. – № 3. – P. 19–25 (In Russian).

[245] Melnikov O.I. Optimal schedule stability for the Bellman-Johnson problem // Vesti Akad. Navuk BSSR, Ser. Fiz.-Mat. Navuk. – 1978. – № 6. – P. 99–101 (In Russian).

[246] Melnikov O.I. On the Bellman-Johnson problem for which all schedules are optimal // Vesti Akad. Navuk BSSR, Ser. Fiz.-Mat. Navuk. – 1984. – № 2. – P. 31–33 (In Russian).

[247] Melnyk S.A., Vickery S.K., Carter P.L. Scheduling, sequencing, and dispatching: Alternative perspectives // Production and Inventory Management Journal. – 1986. – V. 27. – № 2. – P. 58–67.

[248] Morton T.E., Pentico D.W. Heuristic Scheduling Systems - With Applications to Production Systems and Project Management // John Wiley & Sons, A Wiley-Interscience Publication, New York, USA. – 1996.

[249] Mosheiov G. Parallel machine scheduling with a learning effect // Journal of the Operations Research Society. – 2001. – V. 52. – P. 1165–1169.

[250] Mosheiov G. Scheduling problems with a learning effect // European Journal of Operational Research. – 2001. – V. 132. – P. 687–693.

[251] Mosheiov G., Sidney J.B. Scheduling with general job-dependent learning curves // European Journal of Operational Research. – 2003. – V. 147. – P. 665–670.

[252] Mulvey J.M., Vanderbei R.J., Zenios S.A. Robust optimization of large-scale systems // Operations Research. – 1995. – V. 43. – P. 264–281.

[253] Muth J.F., Thompson G.L. Industrial Scheduling // Prentice Nall, Englewood, N.J., USA. – 1963.

[254] Nasution S.H. Fuzzy critical path method // IEEE Transactions on Systems, Man, and Cybernetics. – 1994. – V. 24. – № 1. – P. 48–57.

[255] Neumann K., Nubel H., Schwindt C. Active and stable project scheduling // Mathematical Methods of Operations Research. – 2000. – V. 52. – P. 441–465.

[256] Neumann K., Zimmermann J. Resource levelling for project with schedule-dependent time windows // European Journal of Operational Research. – 1999. – V. 117. – P. 591–605.

[257] Ng C.T., Kovalyov M.Y. An FPTAS for scheduling a two-machine flowshop with one unavailability interval // Naval Research Logistics. – 2004. – V. 51. – P. 307–315.

[258] Ng C.T., Matsveichuk M.Y., Sotskov Yu.N., Cheng T.C.E. Two-machine flow-shop winimum-length scheduling with interval processing times // Asia-Pacific Journal of Operational Research. – 2009. – V. 26. – № 61. – P. 1–19.

[259] Odijk M.A. Sensitivity analysis of a railway station track layout with respect to a given timetable // European Journal of Operational Research. – 1999. – V. 112. – P. 517–530.

[260] O'Donovan R., Uzsoy R., McKay K.N. Predictable scheduling of a single machine with breakdowns and sensitive jobs // International Journal of Production Research. – 1999. – V. 37. – № 18. – P. 4217–4233.

[261] Oh Y., Son S.H. Scheduling real-time tasks for dependability // Journal of the Operations Research Society. – 1997. – V. 48. – P. 629–639.

[262] Özelkan E.C., Duckstein L. Optimal fuzzy counterparts of scheduling rules // European Journal of Operational Research. – 1999. – V. 113. – P. 593–609.

[263] Parunak H., Dyke Van Characterizing the manufacturing scheduling problem // Journal of Manufacturing Systems. – 1991. – V. 10. – № 3. – P. 241–259.

[264] Paz N.M., Leigh W. Maintance scheduling: Issues, results, and research needs // International Journal of Operations and Production Management. – 1994. – V. 47. – P. 47–69.

[265] Penz B., Rapine C., Trystram D. Sensitivity analysis of scheduling algorithms // European Journal of Operational Research. – 2001. – V. 134. – P. 606–615.

[266] Pezzella F., Merelli E. A tabu search method guided by shifting bottleneck for the job shop scheduling problem // European Journal of Operational Research. – 2000. – V. 120. – P. 297–310.

[267] Picard J., Queyranne M. The time-dependent traveling salesman problem and its application to the tardiness problem in one machine scheduling // Operations Research. – 1978. – V. 26. – P. 86–110.

[268] Pinedo M. Minimizing the expected makespan in stochastic flow shops // Operations Research. – 1982. – V. 30. – Nº 1. – P. 148–162.

[269] Pinedo M. Scheduling: Theory, Algorithms, and Systems // Prentice-Hall, Englewood Cliffs, NJ, USA. – 2002.

[270] Pinedo M., Chao X. Operations Scheduling with Applications in Manufacturing and Servies // Irwin/McGraw-Hill, USA. – Singapore. – 1999.

[271] Pinson E. The job shop scheduling problem: A concise survey and some recent development // Chretienne P., Coffman E.G., Lenstra J.K., Liu Z. Scheduling Theory and Its Applications. – John Wiley & Sons. – 1995. – 277–293.

[272] Pinto P.A., Dannenbring D.G., Khumawala B.M. Assembly line balancing with processing alternatives: An application // Management Science. – 1983. – V. 29. – P. 817–830.

[273] Portougal V., Robb D.J. Production scheduling theory: Just where is it applicable // Interfaces. – 2000. – V. 30. – Nº 6. – P. 64–76.

[274] Portougal V., Trietsch D. Makespan-related criteria for comparing schedules in stochastic environment // Journal of the Operations Research Society. – 1998. – V. 49. – P. 1188–1195.

[275] Portougal V., Trietsch D. Johnson's problem with stochastic processing times and optimal service level // European Journal of Operational Research. – 2006. – V. 169. – P. 751–760.

[276] Ramaswamy R., Chakravarti N. Complexity of determining exact tolerances for min-sum and min-max combinatorial optimization problems // Technical Report WPS-247/95. – Indian Institute of Management. – Calcutta. – 1995.

[277] Ramaswamy R., Orlin J.B., Chakravarti N. Sensitivity analysis for shortest path problems and maximum capacity path problems in undirected graphs // Mathematical Programming. Series A. – 2005. – V. 102. – P. 355–369.

[278] Rinnooy Kan A.H.G. Machine Scheduling Problems - Classification, Complexity and Computations // M. Nijhoff, Hague, The Netherlands. – 1976.

[279] Robb D.J., Rohleder T.R. An evaluation of scheduling heuristics for dynamic single-processor scheduling with early/tardy costs // Naval Research Logistics. – 1996. – V. 43. – P. 349–364.

[280] Rommelfanger H.J. Network analysis and information flow in fuzzy environment // Fuzzy Sets and Systems. – 1994. – V. 67. – P. 119–128.

[281] Rosenblatt J.J., Lee H.L. A robustness approach to facilities design // International Journal of Production Research. – 1987. – V. 25. – P. 479–486.

[282] Roy B., Sussmann B. Les problèmes d'ordonnancement avec contraintes disjonctives // Note DS No. 9 bis, SEMA, Montrouge. – 1964.

[283] Sabuncuoglu I., Goren S. Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research // International Journal of Computer Integrated Manufacturing. – 2009. – V. 22. – Nº 2. – P. 138–157.

[284] Sakawa M., Kubota R. Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms // European Journal of Operational Research. – 2000. – V. 120. – P. 393–407.

[285] Sanlaville E. Nearly on line scheduling of preemptive independent tasks // Discrete Applied Mathematics. – 1995. – V. 57. – P. 229–241.

[286] Sanlaville E., Schmidt G. Machine scheduling with availability constraints // Acta Informatica. – 1998. – V. 35. – P. 795–811.

[287] Sarin S.C., Erel E., Dar-El E.M. A methodology for solving single-model, stochastic assembly line balancing problem // OMEGA – International Journal of Management Science. – 1999. – V. 27. – P. 525–535.

[288] Schmidt G. Scheduling with limited machine availability // European Journal of Operational Research. – 2000. – V. 121. – P. 1–15.

[289] Scholl A. Balancing and Sequencing of Assembly Lines // Physica-Verlag, A Springer-Verlag Company. – Heidelberg. – 1999.

[290] Scholl A., Klein R. Balancing assembly lines effectively: A computational comparison // European Journal of Operational Research. – 1998. – V. 114. – P. 51–60.

[291] Seiden S.S. Randomized online interval scheduling // Operations Research Letters. – 1998. – V. 22. – P. 171–177.

[292] Sen T., Gupta S.K. A state-of-art survey of static scheduling research involving due dates // OMEGA – International Journal of Management Science. – 1984. – V. 12. – Nº 1. – P. 63–76.

[293] Sevaux M., Sorensen K. A genetic algorithm for robust schedules in a one-machine environment with ready times and due dates // 4OR: Quarterly Journal of the Belgian, French and Italian Operations Research Societies. – 2004. – V. 2. – Nº 2. – P. 129–147.

[294] Shabtay D., Kaspi M. Minimizing the total weighted flow time in a single machine with controllable processing times // Computers & Operations Research. – 2004. – V. 31. – P. 2279–2289.

[295] Shakhlevich N.V., Sotskov Yu.N., Werner F. Complexity of mixed shop scheduling problems: A survey // European Journal of Operational Research. – 2000. – V. 120. – P. 343–351.

[296] Shakhlevich N.V., Strusevich V.A. Pre-emptive scheduling problems with controllable processing times // Journal of Scheduling. – 2005. – V. 8. – P. 233–253.

[297] Shakhlevich N.V., Strusevich V.A. Single machine scheduling with controllable release and processing parameters // Discrete Applied Mathematics. – 2006. – V. 154. – P. 2178–2199.

[298] Shier D.R., Witzgall G. Arc tolerances in shortest path and network flow problems // Networks. – 1980. – V. 10. – P. 277–291.

[299] Slowinski R., Hapke M. Scheduling Under Fuzziness // Physica-Verlag, Heidelberg, New York. – Heidelberg, New York. – 1999.

[300] Smith S.F., Ow P.S., Muscettola N., Potvin J., Matthy D.C. An integrated framework for generating and revising factory schedules // Journal of the Operations Research Society. – 1990. – V. 41. – № 6. – P. 539–552.

[301] Smith W.E. Various optimizers for single-stage production // Naval Research Logistics Quarterly. – 1956. – V. 3. – № 1. – P. 59–66.

[302] Soroush H.M. Sequencing and due-date determination in the stochastic single machine problem with earliness and tardness costs // European Journal of Operational Research. – 1999. – V. 113. – P. 450–468.

[303] Soroush H.M., Allahverdi A. Stochastic two-machine flowshop scheduling problem with total completion time criterion // International Journal of Industrial Engineering: Theory Application and Practice. – 2005. – V. 12. – № 2. – P. 159–171.

[304] Sotskov Yu.N. Stability of optimal schedule for a set of jobs // Vesti Akad. Navuk BSSR, Ser. Fiz.-Mat. Navuk. – 1988. – № 6. – P. 99–104 (In Russian).

[305] Sotskov Yu.N. Optimal scheduling two jobs with regular criterion // Design Processes Automating. – Institute of Engineering Cybernetics. – Belarus, Minsk. – 1985. – P. 86–95 (In Russian).

[306] Sotskov Yu.N. The stability of high-speed optimal schedules // U.S.S.R. Computational Mathematics and Mathematical Physics. – 1989. – V. 29. – № 3. – P. 57–63.

[307] Sotskov Yu.N. Usage of stability of optimal schedule for design of informational computer systems // Automation and Remote Control. – 1990. – № 3. – P. 12–19 (Translated from Russian).

[308] Sotskov Yu.N. The complexity of shop-scheduling problems with two or three jobs // European Journal of Operational Research. – 1991. – V. 53. – P. 326–336.

[309] Sotskov Yu.N. ϵ-approximate stability radius of Boolean linear form minimization problem // Institute of Engineering Cybernetics, Preprint No. 21, Minsk, Belarus. – 1991 (In Russian).

[310] Sotskov Yu.N. A review of solution stability analysis in sequencing and scheduling // Institute of Engineering Cybernetics, Preprint No. 22, Minsk, Belarus. – 1991.

[311] Sotskov Yu.N. Stability of an optimal schedule // European Journal of Operational Research. – 1991. – V. 55. – P. 91–102.

[312] Sotskov Yu.N. The stability of the approximate Boolean minimization of a linear form // U.S.S.R. Computational Mathematics and Mathematical Physics. – 1993. – V. 33. – № 5. – P. 699–707.

[313] Sotskov Yu.N. Two, three, many or the complexity of scheduling with fixed number of jobs // Ulrich Derigs Andreas Drexl Operations Research Proceedings. – Springer-Verlag Berlin. – Heidelberg. – 1995. – P. 168–172.

[314] Sotskov Yu.N. Software for production scheduling based on the mixed (multi)graph approach // Computing & Control Engineering Journal. – 1996. – V. 7. – № 5. – P. 240–246.

[315] Sotskov Yu.N. Mixed multigraph approach to scheduling jobs on machines of different types // Optimization. – 1997. – V. 42. – P. 245–280.

[316] Sotskov Yu.N. Investigation of optimal schedule stability // Informatics. – 2004. – № 4. – P. 65–75 (In Russian).

[317] Sotskov Yu.N. Recent results on stability analysis of an optimal assembly line balance // International Journal: Information, Theories & Applications. – 2007. – V. 14. – № 2. – P. 174–181.

[318] Sotskov Yu.N., Allahverdi A., Lai T.-C. Flowshop scheduling problem to minimize total completion time with random and bounded processing times // Journal of the Operations Research Society. – 2004. – V. 55. – P. 277–286.

[319] Sotskov Yu.N., Alyushkevich V.B. Stability of an optimal orientation of mixed graph edges // Doklady Akademii Navuk BSSR. – 1988. – V. 32. – № 4. – P. 108–111 (In Russian).

[320] Sotskov Yu.N., Dolgui A., Portmann M.-C. Stability analysis of optimal balance for assembly line with fixed cycle time // European Journal of Operational Research. – 2006. – V. 168. – № 3. – P. 783–797.

[321] Sotskov Yu.N., Dolgui A., Sotskova N.Yu., Werner F. Stability of the optimal line balance for a fixed number of stations // Dolgui A., Soldek J., O. Zaikin Proceedings of the 9th International Multi-Conference Advanced Computer Systems, Production System Design, Supply Chain Management and Logistics, ACS'02 – SCM. – P. 21–28. – Miedzyzdroje, Poland. – October 23 – 25 2002.

[322] Sotskov Yu.N., Dolgui A., Sotskova N.Yu., Werner F. Stability of optimal line balance with given station set // Fiat A., Woeginger G.J. Supply Chain Optimization. Applied Optimization. – Springer. – USA, New York. – 2005. – 135–149.

[323] Sotskov Yu.N., Egorova N.M., Lai T.-C. Minimizing total weighted flow time of a set of jobs with interval processing times // Mathematical and Computer Modelling. – 2009. – V. 50. – P. 556–573.

[324] Sotskov Yu.N., Egorova N.M., Lai T.-C., Werner F. Sequence-dependent setup times in a two-machine job-shop with minimizing the schedule length // International Journal of Operations Research. – 2008. – V. 5. – Nº 1. – P. 68–77.

[325] Sotskov Yu.N., Leontev V.K., Gordeev E.N. Some concepts of stability analysis in combinatorial optimization // Discrete Applied Mathematics. – 1995. – V. 58. – P. 169–190.

[326] Sotskov Yu.N., Shakhlevich N.V. NP-hardness of shop-scheduling problems with three jobs // Discrete Applied Mathematics. – 1995. – V. 59. – P. 237–266.

[327] Sotskov Yu.N., Shilak A.N. Minimization of project-network with given bounds of activity durations // Fleischmann B., Lasch R., Derigs U., Domschke W., Rieder U. Operations Research Proceedings 2000. – P. 384–387. – Dresden, Germany. – June 2001. Springer-Verlag Heidelberg.

[328] Sotskov Yu.N., Shilak A.N. Two-machine job-shop scheduling problem with bounded processing times // 15-th Workshop on Discrete Optimization. – P. 80–81. – Lutherstadt Wittenberg, Germany. – May 13 – 17 2002. Otto-von-Guericke-Universität Magdeburg, FMA, Preprint 12/02, 2002.

[329] Sotskov Yu.N., Shilak A.N. Minimizing the network with bounds of the operation durations // Vesti Akad. Navuk BSSR, Ser. Fiz.-Mat. Navuk. – 2004. – Nº 2. – P. 107–113 (In Russian).

[330] Sotskov Yu.N., Sotskova N.Yu. Stability of an optimal schedule for a job-shop problem with two jobs // International Journal: Information, Theories & Applications. – 2003. – V. 10. – Nº 3. – P. 321–326.

[331] Sotskov Yu.N., Sotskova N.Yu. Scheduling Theory. Systems with Uncertain Numerical Parameters // National Academy of Sciences of Belarus, United Institute of Informatics Problems. – Minsk, Belarus. – 2004 (In Russian).

[332] Sotskov Yu.N., Sotskova N.Yu. Stability of an optimal schedule for a job-shop problem with two jobs // International Journal: Information Theories & Applications. – 2003. – V. 10. – Nº 3. – P. 321–326.

[333] Sotskov Yu.N., Sotskova N.Yu. Stability of an optimal schedule for a job-shop problem with two jobs // Proceedings of the X-th International Conference "Knowledge – Dialogue – Solution". – P. 502-507. – Varna, Bulgaria. – June 16 – 26 2003. Commerce Bulgaria Publishers.

[334] Sotskov Yu.N., Sotskova N.Yu., Werner F. Stability of an optimal schedule in a job shop // OMEGA – International Journal of Management Science. – 1997. – V. 25. – Nº 4. – P. 397–414.

[335] Sotskov Yu.N., Strusevich V.A., Tanaev V.S. Mathematical Models and Methods for Production Scheduling // Belarusian State University. – Universitetskoe, Minsk, Belarus. – 1993 (In Russian).

[336] Sotskov Yu.N., Tanaev V.S. Scheduling theory and practice: Minsk group results // Intelligent Systems Engineering. – 1994. – V. 1. – P. 1–8.

[337] Sotskov Yu.N., Tanaev V.S., Werner F. Stability radius of an optimal schedule: A survey and recent developments // Yu G. Industrial Applications of Combinatorial Optimization. – Kluwer Academic Publishers. – Boston, MA. – 1998. – P. 72-108.

[338] Sotskov Yu.N., Wagelmans A.P.M., Werner F. On the calculation of the stability radius of an optimal or an approximate schedule // Technical Report 9718/A. – Econometric Institute, Erasmus University Rotterdam. – The Netherlands, Rotterdam. – 1997.

[339] Sotskov Yu.N., Wagelmans A.P.M., Werner F. On the calculation of the stability radius of an optimal or an approximate schedule // Annals of Operations Research. – 1998. – V. 83. – P. 213–252.

[340] Sotskova N.Yu. Optimal makespan schedules for a job-shop with uncertain processing times // Carvalho M.F., Müller F.M. Proceedings of the 15th International Conference on CAD/CAM, Robotics & Factories of the Future CARS & FOF99. – MW4:7 – MW4:12. – Águas de Lindóia - Brazil. – August 18-20, 1999. Technological Center for Informatics Foundation Automation Institute, Campinas-SP-Brazil.

[341] Sotskova N.Yu., Tanaev V.S. About the realization of an optimal schedule with operation processing times under conditions of uncertainty // Doklady Natsional'noi Akademii Nauk Belarusi. – 1998. – V. 42. – № 5. – P. 8–12 (In Russian).

[342] Sriskandarajah C., Hall N.G., Kamoun H. Scheduling large robotic cells without buffers // Annals of Operations Research. – 1998. – V. 76. – P. 287–321.

[343] Stanfield P.M., King R.E., Joines J.A. Scheduling arrivals to a production system in a fuzzy environment // European Journal of Operational Research. – 1996. – V. 93. – P. 75–87.

[344] Stein C., Joel W. On the existence of schedules that are near-optimal for both makespan and total weighted completion time // Operations Research Letters. – 1997. – V. 21. – P. 115–122.

[345] Strusevich V.A. Two machine flow shop scheduling problem with no wait in process: Controllable machine speeds // Discrete Applied Mathematics. – 1995. – V. 59. – P. 75–86.

[346] Strusevich V.A. A heuristic for the two-machine open-shop scheduling problem with transportation times // Discrete Applied Mathematics. – 1999. – V. 93. – P. 287–304.

[347] Sun T., Lai K., Lam K., So K. A study of heuristics for bidirectional multi-hoist production scheduling systems // International Journal of Production Economics. – 1998. – V. 14. – P. 144–152.

[348] Sussmann B. Scheduling problems with interval disjunctions // Mathematical Methods of Operations Research. – 1972. – V. 16. – P. 165–178.

[349] Szelke E., Kerr R.M. Knowledge-based reactive scheduling // Production Planning and Control. – 1994. – V. 5. – №. 2. – P. 124–145.

[350] Szwarc W. Solution of the Akers-Friedman scheduling problem // Operations Research. – 1960. – V. 8. – P. 782–788.

[351] Taillard E. Benchmarks for basic scheduling problems // European Journal of Operational Research. – 1993. – V. 64. – P. 278–285.

[352] Tanaev V.S., Gordon V.S., Shafransky Y.M. Scheduling theory. Single-Stage Systems // Kluwer Academic Publishers, Dordrecht, The Netherlands. – 1994.

[353] Tanaev V.S., Kovalyov M.Y., Shafransky Y.M. Scheduling Theory. Batching Technology // National Academy of Sciences of Belarus, Institute of Engineering Cybernetics. – Minsk, Belarus. – 1998 (In Russian).

[354] Tanaev V.S., Sotskov Yu.N., Strusevich V.A. Scheduling Theory: Multi-Stage Systems // Nauka. – Moskow. – 1989 (In Russian).

[355] Tanaev V.S., Sotskov Yu.N., Strusevich V.A. Scheduling Theory: Multi-Stage Systems // Kluwer Academic Publishers, Dordrecht, The Netherlands. – 1994.

[356] Tarjan R.E. Sensitivity analysis of minimum spanning trees and shortest path trees // Information Processing Letters. – 1982. – V. 14. – P. 30–33.

[357] T'Kindt V., Billaut J.-C. Multicriteria scheduling. Theory, models and algorithms // Springer Verlag, Germany. – 2002.

[358] T'kindt V., Gupta J.N.D., Billaut J.-C. Two-machine flowshop scheduling problem with secondary criterion // Computers & Operations Research. – 2003. – V. 30. – P. 505–526.

[359] Trick M.A. Scheduling multiple variable-speed machines // Operations Research. – 1994. – V. 42. – P. 234–248.

[360] Tseng C.-T., Liao C.-J., Huang K.-L. Minimizing total tardiness on a single machine with controllable processing times // Computers & Operations Research. – 2009. – V. 36. – P. 1852–1858.

[361] Tsujimura Y., Gen M., Kubota E. Solving fuzzy assembly-line balancing problem with genetic algorithms // Computers and Industrial Engineering. – 1995. – V. 29. – №. 1-4. – P. 543–547.

[362] Vieira G.E., Herrmann J.W., Lin E. Predicting the performance of rescheduling strategies for parallel machine systems // Journal of Manufacturing Systems. – 2000. – V. 19. – №. 4. – P. 256–266.

[363] Wagner H.M. Global sensitivity analysis // Operations Research. – 1995. – V. 43. – P. 948–969.

[364] Wee H.M., Wang W.T. A variable production scheduling policy for deteriorating items with time-varying demand // Computers & Operations Research. – 1999. – V. 26. – P. 237–254.

[365] Wilson G.R., Jain H.K. An approach to postoptimality and sensitivity analysis of zero-one goal programs // Naval Research Logistics. – 1988. – V. 35. – P. 73–84.

[366] Wong T.N., Leung C.W., Mak K.L., Fung R.Y.K. An agent-based negotiation approach to integrate process planning and scheduling // International Journal of Production Research. – 2006. – 44. – №º 7. – P. 1331–1351.

[367] Wu C.W., Brown K.-N., Beck J.C. Scheduling with uncertain durations: Modeling β-robust scheduling with constraints // Computers & Operations Research. – 2009. – 36. – P. 2348–2356.

[368] Wu S.D., Byeon E.-S., Storer R.H. A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness // Operations Research. – 1999. – 47. – №º 1. – P. 113–124.

[369] Yaman H., Karasan O.E., Pinar M.C. Restricted robust uniform matroid maximization under interval uncertainty // Mathematical Programming. Series A. – 2007. – V. 110. – P. 431–441.

[370] Yamashita D.S., Armentano V.A., Laguna M. Robust optimization models for project scheduling with resource avalability cost // Journal of Scheduling. – 2007. – V. 10. – P. 67–76.

[371] Yang J., Yu G. On the robust single machine scheduling problem // Journal of Combinatorial Optimization. – 2002. – V. 6. – P. 17–33.

[372] Yang W.-H. Scheduling jobs on a single machine to maximize the total revenue of jobs // Computers & Operations Research. – 2009. – V. 36. – P. 565–583.

[373] Younis M.A., Saad B. Optimal resource leveling of multi-resource projects // Computers and Industrial Engineering. – 1996. – V. 31. – P. 1–4.

[374] Yuan J.J., Cheng T.C.E., Ng C.T. NP-hardness of the single-variable-resource scheduling problem to minimize the total weighted completion time // European Journal of Operational Research. – 2007. – V. 178. – P. 631–633.

List of Tables

1.1 Randomly generated problems . . . 62
1.2 Problem $\mathcal{J}6/n=4/\Phi$, $\Phi \in \{\mathcal{C}_{max}, \sum \mathcal{C}_i\}$, with different ranges of the variations of the job processing times p_{ij} . . . 63
1.3 Problem $\mathcal{J}6/n=4/\Phi$, $\Phi \in \{\mathcal{C}_{max}, \sum \mathcal{C}_i\}$, with variability of p_{ij} from job to job 65
1.4 Test problem $\mathcal{J}6/n=6/\mathcal{C}_{max}$ with variability of p_{ij} . . . 70
1.5 Notations for the general shop scheduling problem . . . 80

2.1 Scheduling with different requirements on the numerical data . . . 86
2.2 Calculation of the stability radius $\widehat{\varrho}_1(p)$ for problem $\mathcal{J}3/n=2/\mathcal{C}_{max}$ . . . 91
2.3 Numerical data for problem $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\mathcal{C}_{max}$ . . . 106
2.4 Calculation of the relative stability radius $\widehat{\varrho}_1^{\Lambda(G)}(p \in T)$ for problem $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\mathcal{C}_{max}$ . . . 108
2.5 Constructing a G-solution of problem $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\mathcal{C}_{max}$ by Algorithm $SOL\_\mathcal{C}_{max}(1)$ . . . 108
2.6 Constructing a G-solution to problem $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\mathcal{C}_{max}$ by Algorithm $SOL\_\mathcal{C}_{max}(2)$ . . . 113
2.7 Numerical data for problem $\mathcal{J}3/n=3, a_i \le p_i \le b_i/\sum \mathcal{C}_i$ . . . 122
2.8 G-solution of problem $\mathcal{J}3/n=3, a_i \le p_i \le b_i/\sum \mathcal{C}_i$ with the initial vector $p^0 \in T$ 138
2.9 Auxiliary information for problem $\mathcal{J}3/n=2/\sum \mathcal{C}_i$ . . . 139
2.10 Calculation of the stability radius $\overline{\varrho}_1(p)$ for problem $\mathcal{J}3/n=2/\sum \mathcal{C}_i$ . . . 140
2.11 Auxiliary information for the construction of the sets $\Omega^*_{1k}, k \in \{2,3,4,5\}$, for problem $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\sum \mathcal{C}_i$ . . . 143
2.12 Numerical data for problem $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\sum \mathcal{C}_i$ . . . 144
2.13 Calculation of the relative stability radius $\overline{\varrho}_1^{\Lambda(G)}(p \in T)$ for problem $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\sum \mathcal{C}_i$ . . . 145
2.14 Optimal digraphs for problem $\mathcal{J}3/n=2, a_i \le p_i \le b_i/\sum \mathcal{C}_i$ with different initial vectors $p \in T$ . . . 147
2.15 G-solution of problem $\mathcal{J}3/n=3, a_i \le p_i \le b_i/\sum \mathcal{C}_i$ for different initial vectors $p \in T$ . . . 159
2.16 Types of problems considered in the experiments . . . 163
2.17 Minimal lower and maximal upper bounds for the processing times . . . 164
2.18 Exact G-solutions of randomly generated problems $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$ . . . 165
2.19 Exact G-solutions of randomly generated problems $\mathcal{J}/a_i \le p_i \le b_i/\mathcal{C}_{max}$ . . . 167
2.20 Heuristic G-solutions of randomly generated problems $\mathcal{J}/a_i \le p_i \le b_i/\sum \mathcal{C}_i$ . . . 169
2.21 Heuristic G-solutions of randomly generated problems $\mathcal{J}/a_i \le p_i \le b_i/\mathcal{C}_{max}$ . . . 170
2.22 Common notations . . . 178

3.1 Lower and upper bounds for the job processing times of Example 3.2 . . . 189

3.2 Percentage of instances with $p_{ij}^U = p_{ij}^L + L$ solved due to Theorem 3.1 199
3.3 Percentage of instances with $p_{ij}^U = p_{ij}^L \cdot (1 + l\%/100\%)$ solved due to Theorem 3.1 200
3.4 Intervals of the job processing times for Example 3.3 203
3.5 Percentage of solved instances with an empty set $\mathcal{J}_0$ 221
3.6 Percentage of solved instances with 10% of jobs from set $\mathcal{J}_0$ 222
3.7 Percentage of solved instances with 30% of jobs from set $\mathcal{J}_0$ 224
3.8 Percentage of solved instances with an empty set $\mathcal{J}_0$ 225
3.9 Percentage of solved instances with 30% of jobs from set $\mathcal{J}_0$ 226
3.10 Lower and upper bounds for the job processing times in Example 3.5 231
3.11 Lower and upper bounds for the job processing times in Example 3.6 232
3.12 Non-availability intervals of the machines in Example 3.7 239
3.13 Job processing times for Example 3.7 239
3.14 Stability radii $\rho_j, M_j \in \{M_1, M_2\}$, for Example 3.7 242
3.15 Enlargement radii $\delta_j, M_j \in \{M_1, M_2\}$, for Example 3.7 244
3.16 Percentage of solved instances with $w_1 > 0$ and $w_2 > 0$ 249
3.17 Percentage of solved instances with $w = w_2$ 251
3.18 Percentage of solved instances with $w = w_1$ 252
3.19 Percentage of solved instances with $p_{i2} = 2p_{i1}$, $w_1 > 0$ and $w_2 > 0$ 253
3.20 Percentage of solved instances with $p_{i2} = 2p_{i1}$ and $w = w_1$ 254
3.21 Average running time and percentage of solved instances with $w_1 > 0$ and $w_2 > 0$ 254
3.22 Average running time and percentage of solved instances with $w = w_2$ 255
3.23 Average running time and percentage of solved instances with $w = w_1$ 255
3.24 Average running time and percentage of solved instances with $p_{i2} = 2p_{i1}$, $w_1 > 0$ and $w_2 > 0$ 256
3.25 Average running time and percentage of solved instances with $p_{i2} = 2p_{i1}$ and $w = w_1$ 256
3.26 Processing times for Example 3.8 265
3.27 Processing times for Example 3.9 266
3.28 Processing times for Example 3.10 266
3.29 Percentage of solved moderate (easy) instances with $w_1 > 0$ and $w_2 > 0$ . . . 270
3.30 Percentage of solved moderate (hard) instances with $w_1 > 0$ and $w_2 > 0$ . . . 271
3.31 Percentage of solved moderate (easy) instances with $w = w_1$ 272
3.32 Percentage of solved moderate (hard) instances with $w = w_1$ 273
3.33 Percentage of solved moderate (easy) instances with $w = w_2$ 274
3.34 Percentage of solved moderate (hard) instances with $w = w_2$ 275
3.35 Average running time and percentage of solved instances with $w_1 > 0$ and $w_2 > 0$ 276
3.36 Average running time and percentage of solved instances with $p_{i,2} = 2p_{i,1}, J_i \in J(12)$, $p_{k,1} = 2p_{k,2}, J_k \in J(21))$, $w_1 > 0$ and $w_2 > 0$ 276
3.37 Average running time and percentage of solved instances with $w = w_1$ 277
3.38 Average running time and percentage of solved instances with $p_{i,2} = 2p_{i,1}, J_i \in J(12)$, $p_{k,1} = 2p_{k,2}, J_k \in J(21)$, $w = w_1$ 277
3.39 Average running time and percentage of solved instances with $w = w_2$ 278
3.40 Average running time and percentage of solved instances with $p_{i,2} = 2p_{i,1}, J_i \in J(12)$, $p_{k,1} = 2p_{k,2}, J_k \in J(21))$, $w = w_2$ 278
3.41 Notations for the two-machine flow shop and job shop 288

List of Figures

1.1 Maximal, average and minimal values of $\widehat{\varrho}_s(p)$ for the problems of type EE . 59
1.2 Maximal, average and minimal values of $\overline{\varrho}_s(p)$ for the problems of type EE . 59
1.3 Maximal, average and minimal values of $\widehat{\varrho}_s(p)$ for the problems of types ER, RE and RR 60
1.4 Maximal, average and minimal values of $\overline{\varrho}_s(p)$ for the problems of types ER, RE and RR 61
1.5 Randomly generated mixed graphs (Q^J, A^J, E^J) for problem $\mathcal{J}6/n=4/\Phi$ . . 63
1.6 Percentage of the number γ (and λ') of competitive (considered) digraphs for the problems of type EE for $\widehat{\varrho}_s(p)$ 66
1.7 Percentage of the number γ (and λ') of competitive (considered) digraphs for the problems of type EE for $\overline{\varrho}_s(p)$ 67

2.1 Weighted mixed graph $G(p) = (Q^J(p), A^J, E^J)$ 89
2.2 Optimal digraph $G_1 = (Q^J, A^J \cup E_1^J, \emptyset)$ with the completion times c_{ij} presented near the vertices $O_{ij} \in Q^J$ 89
2.3 Competitive digraph $G_3 = (Q^J, A^J \cup E_3^J, \emptyset)$ for $G_1 = (Q^J, A^J \cup E_1^J, \emptyset)$ which is optimal for vector $p = (75, 50, 40, 60, 55, 30)$ of the processing times 92
2.4 Projections of the stability balls on the plane for problem $\mathcal{J}3/n=2, a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ constructed by Algorithm $SOL\_\mathcal{C}_{max}(1)$ 109
2.5 Projections of the stability balls on the plane for problem $\mathcal{J}3/n=2, a_i \leq p_i \leq b_i/\mathcal{C}_{max}$ constructed by Algorithm $SOL\_\mathcal{C}_{max}(2)$ 114
2.6 Mixed graph $G = (Q^J, A^J, E^J)$ for problem $\mathcal{J}3/n=3, a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$ . . . 122
2.7 Digraphs G_1^T, G_2^T and G_5^T which define a minimal G-solution $\Lambda^T(G)$ for Example 2.2 123
2.8 Digraphs $\Lambda(G) = \{G_1, G_2, \ldots, G_5\}$ numbered in non-decreasing order of the objective function values $\sum \mathcal{C}_i$ 139
2.9 Projections of the stability balls with the center $p = (75, 50, 40, 60, 55, 30)$ on the plane for problem $\mathcal{J}3/n=2, a_i \leq p_i \leq b_i/\sum \mathcal{C}_i$ 147

3.1 Initial part of an optimal schedule with the processing times of jobs $\{J_1, J_2, \ldots, J_7\}$ given in (3.29) 206
3.2 Initial part of an optimal schedule for the jobs of set $\{J_1, J_2, \ldots, J_{k-1}\}$ 214
3.3 Digraph representing the binary relations $\mathcal{A}_{\preceq}$ for Example 3.6 234
3.4 Optimal schedule for problem $\mathcal{F}2,NC^{off}/pmtn, w = 1/\mathcal{C}_{max}$ 237

Curriculum Vitae

Yuri N. Sotskov was born on January 1, 1948 in Sakhalin Region (Russia). He finished secondary school with a gold medal. In 1966, he started studying mathematics at the Belorussian State University, and he graduated at the Faculty of Applied Mathematics in 1971. In 1980, he defended his Ph.D. thesis 'Some scheduling problems on the mixed (disjunctive) graphs' at the Institute of Mathematics of the National Academy of Sciences of Belarus. In 1991, he defended the DSc thesis 'Network models and methods in scheduling theory' at the Institute of Cybernetics of the National Academy of Sciences of Ukraine (Kiev). In 1994, he obtained the title of a Professor in the Application of Mathematical Models and Methods of Scientific Research. He has more than 230 publications on applied mathematics, operations research and scheduling. He was a supervisor of seven Ph.D. scholars. In 1998, he received the National Price of Belarus in Science and Engineering.

Nadezhda Yu. Sotskova was born on May 21, 1975 in Minsk (Belarus). From 1992 till 1997 she studied mathematics at the Belarusian State University, graduated in 1997 on the subject of economical cybernetics, and got the qualification of a mathematician-economist. In 1998, she started as a Ph.D. student at the Faculty of Mathematics of the Otto-von-Guericke University in Magdeburg (Germany). In 2001, she defended her Ph.D. thesis 'Optimal scheduling with uncertainty in the numerical data on the basis of a stability analysis'. She is author of 30 papers in journals on operations research and scheduling.

Tsung-Chyan Lai was born on May 3, 1960. He received a Bachelor degree (1978 – 1982) and a Master degree (1982 – 1984) in industrial engineering both from National Tsing Hua University in Taiwan. After completing the two-year military service, he went abroad for a Ph.D., where he received a Ph.D. (1986 – 1991) in industrial engineering from Stanford University in California, USA. After receiving a Ph.D. from Stanford University, he has been a faculty member at Department of Business Administration, National Taiwan University, where he was an associate professor from 1991 to 1996

and a professor since 1996. His research interests include scheduling theory, optimization and operations management.

Frank Werner was born on May 29, 1955 in Magdeburg (Germany). He finished secondary school with honors in 1973. In 1975, he started studying mathematics at the Technical University 'Otto von Guericke' in Magdeburg, where he graduated in 1980 with honors. In 1984, he defended his Ph.D. thesis 'On the solution of special sequencing problems' with summa cum laude. In 1989, he defended his habilitation thesis 'On the structure and approximate solution of selected combinatorial optimization problems'. In 1992, he received a fellowship from the Alexander-von-Humboldt Foundation. Currently he works as an extraordinary professor at the Otto-von-Guericke-University in Magdeburg. He has about 120 papers in international journals and book contributions in applied mathematics, operations research and scheduling, and he is an author of one textbook. He was also a guest editor of special issues in the journals Annals of Operations Research as well as Computers & Operations Research and a supervisor of several Ph.D. and post-doc students.