MISTA 2009

Parallel Machine Problems with Equal Processing Times

Svetlana A. Kravchenko · Frank Werner

Abstract The basic scheduling problem we are dealing with is the following. There are n jobs, each requiring an identical execution time. All jobs have to be processed on a set of parallel machines. Preemptions can be either allowed or forbidden. The aim is to construct a feasible schedule such that a given criterion is minimized. For a couple of problems of this type, recently the complexity status has been solved and several interesting results have been presented. In this paper, we survey existing approaches for the problem class considered.

1 Introduction

We consider the following basic scheduling problem. There are n jobs, denoted by $J_1, J_2, \ldots, J_n$, each requiring an identical execution time p. With each job $J_j, j = 1, \ldots, n$, there are associated a release time r_j, a due date d_j, a deadline D_j and a weight w_j. All jobs have to be processed on a set of m identical machines. Each machine can process only one job at a time. During the processing of the jobs, preemptions can be allowed or forbidden. In the first case, the processing of any job once started has to be completed without any interruptions. In the second case, the processing of any operation may be interrupted and resumed later, possibly on another machine. The objective is to construct a feasible schedule so as to minimize a given criterion. In this paper, we discuss several solution approaches for the problem considered and expose problems with an open complexity status. For the interested readers, we can recommend in addition the collections of results from [8] and [14].

United Institute of Informatics Problems
Minsk, 220012 Belarus
E-mail: kravch@newman.bas-net.by

Fakultät für Mathematik, Otto-von-Guericke-Universität
Magdeburg, 39106 Germany
E-mail: frank.werner@mathematik.uni-magdeburg.de

2 Problems without preemptions

Whereas the overwhelming majority of scheduling problems appears to be NP-hard, problems with equal processing time jobs form a remarkable case which is still open for most problems. Intuitively, such problems look polynomially solvable.

Due to the possibility to enumerate the possible places of the jobs in an optimal schedule, the model reminds an assignment problem. Therefore, one can intuitively suppose the existence of a polynomial algorithm for any monotonic criterion. Nevertheless, since prime conflicts are caused by overlapping intervals most of the problems have an open complexity status.

A special case is the model with $p = 1$. For most criteria, this model can be solved by a network flow algorithm. However, this approach cannot be applied to the general model with arbitrary p since, in the case of $p = 1$, the main conflicts among overlapping places for processing jobs disappear. We refer the reader to [3] for a survey.

2.1 Classical criteria

If the processing times can be arbitrary, problem $1|r_j, D_j|-$ is unary NP-hard by a polynomial reduction from the 3-partition problem, see [26]. In [17], an $O(n \log n)$ algorithm has been proposed for problem $1|r_j, p_j = p, D_j, prec| C_{max}$. More precisely, the authors consider a problem with unit processing times and arbitrary rational release dates and deadlines. However, by an appropriate scaling one can show that these two models are equivalent. They show that it is possible to modify the release times and deadlines so as to reflect the partial order (i.e., for the problem considered the constraint $prec$ is irrelevant) by assigning $r_j := \max\{\{r_j\} \cup \{r_i + 1 \mid T_i \prec T_j\}\}$ and $d_j := \min\{\{d_j\} \cup \{d_i - 1 \mid T_j \prec T_i\}\}$. After such an assignment condition, $T_i \prec T_j$ implies $r_i < r_j$ and $d_i < d_j$. By straightforward interchange arguments, one can show that any schedule for problem $1 \mid r_j, p_j = p, D_j \mid C_{max}$ can be transformed into a schedule for problem $1 \mid r_j, p_j = p, D_j, prec \mid C_{max}$ without changing the C_{max}-value. To solve problem $1 \mid r_j, p_j = p, D_j \mid C_{max}$, the concept of forbidden regions (intervals) has been proposed (see [17]), i.e., open intervals where no job can start. The set of all forbidden regions is formed iteratively. The main observation used is the following. Assume that one knows the set of forbidden regions in the interval $[r_i, D_j]$, and let $J_1, \ldots, J_k$ be the set of jobs with release times and deadlines from the interval $[r_i, D_j]$. Now we set $r_1 = \ldots = r_k = r_i$ and $D_1 = \ldots = D_k = D_j$ and find the largest value of e such that all jobs $J_1, \ldots, J_k$ can be scheduled in $[e, D_j]$ under the condition that no job can start in a forbidden interval. Now, if $r_i \leq e < r_i + 1$, then $]e - 1, r_i[$ is declared as a forbidden region, since in any feasible schedule a job starting in $]e - 1, r_i[$ is not from $\{J_1, \ldots, J_k\}$ and hence, the set $\{J_1, \ldots, J_k\}$ is finished after D_j. After forming the set of all forbidden regions, the implementation of the earliest deadline scheduling rule gives an optimal schedule.

In [35], it has been shown that problem $P|p_j = 1, D_j, prec|-$ is unary NP-hard by a polynomial reduction from the 3-satisfiability problem. In [27], the authors have shown that even problem $P \mid p_j = 1, D_j = 3, prec \mid -$ is unary NP-hard. Note that problem $P \mid p_j = 1, D_j = 2, prec \mid -$ can be polynomially solved in $O(n)$ time. The important question about the complexity status of problem $P3 \mid p_j = 1, prec \mid C_{max}$ is still open whereas in [16], problem $P2 \mid p_j = 1, prec \mid C_{max}$ was solved in $O(n^3)$ time by a reduction of the problem to the maximum matching problem. In [7], an $O(n \log n)$

algorithm has been proposed for problem $P \mid r_j, p_j = p, outtree \mid C_{max}$ and it has been shown that problem $P \mid r_j, p_j = p, intree \mid C_{max}$ is unary NP-hard.

In [30], a polynomial algorithm with complexity $O(n^3 \log\log n)$ has been developed for problem $P|r_j, p_j = p, D_j|-$. The algorithm is based on an analysis of the structural properties of an optimal schedule. The main features of the optimal structure used in that paper are the following:

1. Any schedule is completely defined by the set of time slots $(1, t_1)$, $(2, t_2)$, ..., (n, t_n), where $i = 1, \ldots, n$ is the slot number and t_i is the starting time of slot i.
2. If we know the set of all occupied time slots, then an optimal schedule can be constructed by the earliest deadline scheduling procedure.
3. If the earliest deadline scheduling procedure generates a sequence $(1, t_1), \ldots, (k, t_k)$, such that job J_l processed in (k, t_k) is late, then job J_l cannot be scheduled in (k, t_k) and has to be scheduled earlier. To provide this, one of the slots $(1, t_1)$, ..., $(k-1, t_{k-1})$ has to be pulled right. To this end, Simons chooses the closest to (k, t_k) slot which is occupied by a job whose deadline exceeds D_l.

It has been shown that the use of these principles leads to the construction of an optimal schedule for problem $P|r_j, p_j = p, D_j|-$. The same algorithm solves the problems $P|r_j, p_j = p, D_j|C_{max}$ and $P|r_j, p_j = p, D_j|\sum C_j$.

In [32], an $O(mn^2)$ time algorithm was proposed for problems $P|r_j, p_j = p, D_j|-$, $P|r_j, p_j = p, D_j|C_{max}$, and $P|r_j, p_j = p, D_j|\sum C_j$. The algorithm is substantially based on the ideas from [17] and [30]. In [10], the following linear programming formulation was proposed for problem $P|r_j, p_j = p, D_j|-$:

$$\sum_{i=1}^{z} x_{ji} = p, \quad j = 1, \ldots, n \tag{1}$$

$$\sum_{j=1}^{n} x_{j,i+1} + \ldots + \sum_{j=1}^{n} x_{j,i+y} \leq mp, \quad i = 0, \ldots, z - y \tag{2}$$

$$x_{ji} = 0 \text{ if } I_i \not\subseteq [r_j, D_j[, \quad i = 1, \ldots, z, \quad j = 1, \ldots, n \tag{3}$$

$$0 \leq x_{ji} \leq p, \quad i = 1, \ldots, z, \quad j = 1, \ldots, n \tag{4}$$

Here x_{ji} is equal to the amount of job J_j processed in the interval I_i, where $\{I_i \mid i \in \{1, \ldots, z\}\} = \{[r_j + kp, r_j + kp + p \; [\mid k \in \{\ldots, -1, 0, 1, \ldots\}\}$, and $y = \max\{k \mid I_{i+1} \cap \ldots \cap I_{i+k} \neq \emptyset, i \in \{0, \ldots, z-k\}\}$. In the above system, the polyhedron (1) and for each $i = 0, \ldots, z - y$, with $i + ey \leq z$, the polyhedron (2) are integer. Nevertheless, their intersection is not an integer polyhedron and therefore, the obtained solution is not necessarily integer. Using an obtained solution, one can construct an optimal schedule in two equivalent ways, namely:

1. It is possible to find the intervals which are occupied in a feasible schedule. Then the earliest deadline scheduling procedure generates an optimal schedule.
2. It is possible to transform the obtained solution into the form $x^*_{ji} \in \{0, p\}$. The obtained vector gives an optimal solution for problem $P \mid r_j, p_j = p, D_j \mid -$.

In both cases, the following property of an optimal solution holds: If $k = \min\{i \mid x^*_{ji} \neq 0, j = 1, \ldots, n\}$ for the optimal solution, then the time slot I_k is occupied in an optimal schedule.

It has been shown in [10] that, to solve problem $P|r_j, p_j = p, D_j|\sum C_j$, it is sufficient to solve the following linear programming problem:

$$\text{Minimize} \sum_{i=1}^{z}\sum_{j=1}^{n} D(I_i)x_{ji}$$

$$\text{subject to (1), (2), (3), (4).}$$

Here $D(I_i)$ is the right endpoint of the interval I_i.

In [15], the proposed linear programming formulation was transformed by means of the substitution $y_t = \sum_{s=1}^{t}\sum_{j=1}^{n} x_{js}$ into the following form:

$$\text{Minimize} \sum_{i=1}^{z} D(I_i)(y_i - y_{i-1})$$

$$\text{subject to } y_z - y_0 = n$$

$$y_i - y_{i-1} \geq 0, \quad i = 1, \ldots, n$$

$$y_i - y_{i-y+1} \leq m, \quad i = y, \ldots, n$$

$$y_0 = 0.$$

The authors have shown that this model can be solved in $O(n^4)$ time.

To solve problem $P|r_j, p_j = p|\sum w_j C_j$, it is sufficient to solve the following linear program (see [10]):

$$\text{Minimize} \sum_{i=1}^{z}\sum_{j=1}^{n} w_j D(I_i)x_{ji}$$

$$\text{subject to (1), (2), (3), (4).}$$

In [11], it has been shown that, in order to solve problem $P|r_j, p_j = p|\sum T_j$, it is sufficient to

$$\text{Minimize} \sum_{i=1}^{z}\sum_{j=1}^{n} \max\{0, D(I_i) - d_j\}\, x_{ji}$$

$$\text{subject to (1), (2), (3), (4).}$$

In [2], a polynomial time algorithm with the complexity $O(n^{6m+1})$ was proposed for problem $Pm \mid r_j, p_j = p \mid \sum w_j U_j$. The algorithm is based on the following observations and definitions:

1. It is possible to restrict the set of starting times by $\{r_j + zp \mid z \in \{0, \ldots, n\}, j \in \{0, \ldots, n\}\}$.
2. If the set of time slots $(1, t_1), \ldots, (n, t_n)$ is known in advance for an optimal schedule, where $i = 1, \ldots, n$ is a slot number and t_i is the starting time of slot i, then the desired schedule can be constructed by the earliest due date rule.
3. The only situation when job J_j follows J_i with $d_j < d_i$ is the situation when J_i is processed within $]r_j - p, r_j + p[$, i.e., the starting time of J_i is before r_j.
4. A profile is defined as a vector $(a_1, \ldots, a_m)$, where $a_i \in \{r_j + zp \mid z \in \{0, \ldots, n\}, j \in \{0, \ldots, n\}\}$, and $\max\{a_1, \ldots, a_m\} - \min\{a_1, \ldots, a_m\} \leq p$. In an optimal schedule, for any job J_j scheduled at t_j, only a profile a with $t_j \in \{a_1, \ldots, a_m\}$ can be prescribed.

5. $k[a,b]$ is defined as the set of early jobs from $\{J_1,\ldots,J_k\}$ scheduled between the profiles a and b, and $W_k[a,b]$ is the maximal weight for such a set.

Now, dynamic programming can be applied by using the formulas

$$W_k[a,b] = \max\{W'_k[a,b], W_{k-1}[a,b]\} \text{ if } \max\{a_1,\ldots,a_m\} - p \le r_k < \min\{b_1,\ldots,b_m\},$$

$$\text{and } W_k[a,b] = W_{k-1}[a,b] \text{ if } r_k \notin [\max\{a_1,\ldots,a_m\} - p, \min\{b_1,\ldots,b_m\}[,$$

$$\text{where } W'_k[a,b] = \max_{\substack{\min\{x_1,\ldots,x_m\}\in \\ [r_k, d_k - p]}} \{W_{k-1}[a,x] + w_k + W_{k-1}[x',b]\}.$$

In [12], an $O(n^5)$ algorithm was proposed for problem $1 \mid r_j, p_j = p \mid \sum U_j$. This algorithm is analogous to the algorithm from [2], however, the authors do not use $W_k[a,b]$ but $w_k[a,B]$, i.e., for the given w,k and a, the authors minimize B. For the single machine case, B is the length of the considered subschedule. Thus, they define:

1. $k[a,\cdot]$ is the set of jobs from $\{J_1,\ldots,J_k\}$ such that $r_j \ge a$ holds,
2. $w_k[a,B]$ equals the minimal value B such that it is possible to execute w jobs from $k[a,\cdot]$ in the time interval $[a+p,b]$.

In [29], a polynomial time algorithm was proposed for problem $P \mid r_j, p_j = p, D_j \mid L_{max}$. It is based on a binary search technique and on the fact that the number of possible starting points is polynomial in the problem size.

In [13], the case with identical jobs and uniform parallel machines has been considered, i.e., when each machine has some given speed. It has been shown how to solve problems $Q|p_j = p|\sum \varphi_j$ and $Q|p_j = p|\max \varphi_j$ in $O(n^2)$ time, where $\max \varphi_j = \max_{1\le j\le n} \varphi_j(C_j)$, and $\varphi_j, j = 1,\ldots,n$, are non-decreasing functions in the job completion times. The authors also indicated how to reduce the complexity for classical criteria and solved problems $Q|r_j, p_j = p|C_{max}$ and $Q|r_j, p_j = p|\sum C_j$ in $O(n \log n)$ and $O(mn^{2m+1})$ time, respectively.

Note that currently the most interesting open problems are $P \mid r_j, p_j = p \mid \sum U_j$ and $P \mid r_j, p_j = p, D_j \mid \sum w_j C_j$.

2.2 Some generalizations

In [6], a dynamic programming approach was used to solve polynomially problem $Pm \mid r_j, p_j = p \mid \sum f_j$, where $\sum f_j$ is an objective function depending on the completion times C_j, such that f_j is non-decreasing and $f_i - f_k$ is monotonic. Note that both classical criteria $\sum w_j C_j$ and $\sum T_j$ can be described in such a way.

In [20], a linear programming approach was proposed for problem $P \mid r_j, p_j = p \mid \sum f_j$, where $\sum f_j$ is the objective function from [6], i.e., f_j depends on the completion time C_j, such that f_j is non-decreasing, and $f_i - f_k$ is monotonic, and for problem $P \mid r_j, p_j = p, D_j \mid \max \varphi_j$, where φ_j is any non-decreasing function in the completion time C_j. The classical scheduling criteria described as $\max \varphi_j(C_j)$ are the minimization of maximum lateness $L_{max} = \max_j\{C_j - d_j\}$ and maximum tardiness $\max_j\{T_j\} = \max_j\{L_j, 0\}$.

The approach for problem $P \mid r_j, p_j = p \mid \sum f_j$ is analogous to that from [11] and consists in minimizing

$$\sum_{i=1}^{z} \sum_{j=1}^{n} f_j(D(I_i)) x_{ji}$$

$$\text{subject to (1), (2),}$$

$$x_{ji} = 0 \text{ if } R(I_i) < r_j, \quad i = 1, \ldots, z, \quad j = 1, \ldots, n$$

$$x_{ji} \geq 0, \quad i = 1, \ldots, z, \quad j = 1, \ldots, n.$$

Here $R(I_i)$ is the left endpoint of the interval I_i.

A polynomial algorithm for problem $P \mid r_j, p_j = p, D_j \mid \max \varphi_j(C_j)$ can be briefly described as follows. Note that the number of intervals available for processing can be polynomially bounded. Therefore, the possible number of different values $\varphi_j(D(I_i))$ is polynomially bounded, too. Take any $F = \varphi_j(D(I_i))$ for some i and j. Consider the following feasibility problem:

$$\sum_{i=1}^{z} x_{ji} = p, \quad j = 1, \ldots, n$$

$$\sum_{j=1}^{n} x_{j,i+1} + \ldots + \sum_{j=1}^{n} x_{j,i+y} \leq mp, \quad i = 0, \ldots, z - y$$

$$x_{ji} = 0 \text{ if } R(I_i) < r_j \quad i = 1, \ldots, z, \quad j = 1, \ldots, n$$

$$x_{ji} = 0 \text{ if } \varphi_j(D(I_i)) > F$$

$$x_{ji} \geq 0, \quad i = 1, \ldots, z, \quad j = 1, \ldots, n$$

It is possible to find a solution for the above problem such that $x_{j,i} \in \{0, p\}$. At the same time, the obtained solution can be considered as a solution for problem $P \mid r_j, p_j = p, D_j \mid \max \varphi_j(C_j) \leq F$. Applying the same procedure for all different values of $F = \varphi_j(D(I_i))$, we can choose the minimal value of F. Since the number of different values $\varphi_j(D(I_i))$ is polynomially bounded, the proposed algorithm is polynomial.

In [31], the following problem has been considered. With each of the n jobs, there is associated a set of intervals. Each interval has a starting time and a finishing time. All data are assumed to be integers. The goal is to construct a feasible schedule so that each job is processed only in one of the prescribed intervals. It has been shown that the considered problem is unary NP-hard for the case of one machine and two prescribed intervals for each job by a polynomial reduction from the 3-satisfiability problem.

In [19], the following problem has been considered. As before, for each job J_j, $j = 1, \ldots, n$, a processing time $p_j = p$, a release date r_j, and a deadline D_j are given. Besides we suppose that the time interval $[\min_j\{r_j\}, \max_j\{D_j\}[$ is divided into several intervals $[t_1, t_2[, [t_2, t_3[, \ldots, [t_{T-1}, t_T[$, where $\min_j\{r_j\} = t_1 \leq t_2 \leq \ldots \leq t_T = \max_j\{D_j\}$, such that for each interval $[t_g, t_{g+1}[$ the number of available machines m_{g+1} is known in advance. Note that we do not fix the concrete set of m_{g+1} machines, i.e., at two different points of $[t_g, t_{g+1}[$, one can use different sets of m_{g+1} machines. Preemption of processing is not allowed, i.e., the processing of any job started at time t on one of the identical machines will be completed at time $t + p$ on the same machine. We want to find a feasible schedule such that the maximal number of machines used by $J_1, \ldots, J_n$ is minimal. The problem has been reduced to the following linear programming problem.

$$\text{Minimize } M$$

$$\text{subject to } \sum_{i=1}^{z} x_{ji} = p, \quad j \in \{1, \ldots, n\}$$

$$\sum_{j=1}^{n} x_{j,i+1} + \ldots + \sum_{j=1}^{n} x_{j,i+q} \leq \min\{M, m_{k+1}\}p,$$

$$\begin{aligned} \text{where} \quad & i \in \{0, \ldots, z-q\}, q \in \{1, \ldots, y\}, \\ & k \in \{1, \ldots, T-1\}, \\ & I_{i+1} \cap \ldots \cap I_{i+q} \cap [t_k, t_{k+1}[\neq \emptyset \end{aligned}$$

$$x_{ji} = 0 \text{ if } R(I_i) < r_j, \quad i = 1, \ldots, z, \quad j = 1, \ldots, n$$

$$x_{ji} = 0 \text{ if } D_j < D(I_i), \quad i = 1, \ldots, z, \quad j = 1, \ldots, n$$

$$x_{ji} \geq 0, \quad i = 1, \ldots, z, \quad j = 1, \ldots, n.$$

If we set x_{ji} equal to the amount of job J_j processed in the interval I_i and M is the number of machines we want to minimize, then any feasible schedule for the scheduling problem under consideration can be described as a feasible solution of the above linear programming problem.

On the other hand, the solution (x^*, M^*) obtained for the linear programming problem can be transformed into an optimal solution of the scheduling problem considered in polynomial time.

3 Problems with preemptions

Now we consider the following basic problem. There are m parallel machines and n jobs, each requiring an identical execution time p. With each job J_j, there is associated a release time r_j. The processing of any job may be interrupted arbitrarily often and resumed later on any machine. The objective is to construct a feasible schedule so as to minimize a given criterion.

It has been shown in [34] that problem $P \mid p_j = 1, prec, pmtn \mid \sum C_j$ is unary NP-hard. In [24], the technique based on linear programming has been developed. The proposed approach allows one to solve problem $R \mid r_j, pmtn \mid L_{max}$ in polynomial time. Using a pseudopolynomial reduction from numerical matching with target sums, it has been proved that problem $P \mid p_j = p, pmtn \mid \sum w_j U_j$ is unary NP-hard [9]. It has been noted in [2] that the algorithm developed in [24] for problem $R \mid r_j, pmtn \mid L_{max}$ can be used to solve both problem $P \mid p_j = p, pmtn \mid \sum U_j$ and problem $Q \mid p_j = p, pmtn \mid \sum U_j$ in polynomial time. Problem $P \mid r_j, p_j = p, pmtn \mid \sum U_j$ is unary NP-hard [18]. It is interesting to note that problem $P \mid pmtn \mid \sum U_j$ is binary NP-hard [23], however, its complexity status under an unary encoding is still an open question.

Problem $P \mid r_j, p_j = p, pmtn \mid \sum C_j$ has been solved in [1]. It is possible to prove that for problem $P \mid r_j, p_j = p, pmtn \mid \sum C_j$, an optimal schedule can be found in the class of schedules, where each job J_j is processed on machine m only within an (possibly empty) interval $[S_{j,m}, C_{j,m}[$, such that $C_{j,m} \leq S_{j+1,m}$ for each m and $j < n$, and $C_{j,m} \leq S_{j,m-1}$ for each $m > 1$ and j.

Using such a structure of an optimal schedule, one can formulate the following linear program which will solve problem $P \mid r_j, p_j = p, pmtn \mid \sum C_j$ in polynomial time.

$$\text{Minimize} \quad \sum_{j=1}^{n} C_{j,1}$$

$$\text{subject to} \quad S_{j,m} \geq r_j, \qquad j = 1, \ldots, n$$

$$\sum_{q=1}^{m}(C_{j,m} - S_{j,m}) = p, \qquad j = 1, \ldots, n$$

$$S_{j,m} - C_{j,m} \leq 0, \qquad j = 1, \ldots, n, \quad m = 1, \ldots, m$$

$$C_{j,q} - S_{j,q-1} \leq 0, \qquad j = 1, \ldots, n, \quad q = 2, \ldots, m$$

$$C_{j,q} - S_{j+1,q} \leq 0, \qquad j = 1, \ldots, n-1, \quad q = 1, \ldots, m$$

Another idea was used in [21] for problem $Q \mid r_j, p_j = p, pmtn \mid \sum C_j$. Note that for problem $Q \mid r_j, p_j = p, pmtn \mid \sum C_j$, an optimal schedule can be found in the class of schedules, for which $C_1 \leq C_2 \leq \cdots \leq C_n$ holds, i.e., there exists an optimal solution in which the completion times are in the same order as the release times.

Let s^* be an optimal schedule for problem $Q \mid r_j, p_j = p, pmtn \mid \sum C_j$ with $C_1(s^*) \leq \cdots \leq C_n(s^*)$. Further we set $r_{n+1} = r_n + n \cdot \max_q\{p/s_q\}$, i.e., r_{n+1} is a time point after which no job will be processed. Each $C_j(s^*)$ belongs to some interval $[r_i, r_{i+1}]$. However, if we know for each $C_j(s^*)$ the corresponding interval $[r_i, r_{i+1}]$ such that $C_j(s^*) \in [r_i, r_{i+1}]$, then an optimal schedule can be easily found using a reduction to a network flow problem, see [25]. Thus, the main question is to know the interval $[r_i, r_{i+1}]$ for each $C_j(s^*)$ such that $C_j(s^*) \in [r_i, r_{i+1}]$. However, this difficulty can be avoided due to criterion $\sum C_j$. For any job J_j, let the time interval $[r_i, r_{i+1}]$ be such that $C_j \in [r_i, r_{i+1}]$. Taking into account that $r_1 = 0$, we obtain $C_j = (r_2 - r_1) + (r_3 - r_2) + \cdots + (r_i - r_{i-1}) + (C_j - r_i)$. Due to this decomposition, we introduce the completion time of job J_j for each interval $[r_i, r_{i+1}]$.

For each job J_j with $j = 1, \ldots, n$ and for each interval $[r_i, r_{i+1}]$ with $i = 1, \ldots, n$, we define the value $C(J_j, r_i)$ such that $C(J_j, r_i) = C_j(s^*)$ if $C_j(s^*) \in]r_i, r_{i+1}[$, but if $C_j(s^*) \leq r_i$, then we set $C(J_j, r_i) = r_i$, and if $C_j(s^*) \geq r_{i+1}$, then we set $C(J_j, r_i) = r_{i+1}$. So, for each $i = 1, \ldots, n$, the values $r_i \leq C(J_1, r_i) \leq \cdots \leq C(J_i, r_i) \leq C(J_{i+1}, r_i) = \cdots = C(J_n, r_i) = r_{i+1}$ define a partition of the interval $[r_i, r_{i+1}]$. In turn, each interval $[C(J_{j-1}, r_i), C(J_j, r_i)]$ is completely defined by the jobs processed in it. Thus, we denote by $v(J_k, M_q, J_j, r_i)$ the part (amount) of job J_k processed in the interval $[C(J_{j-1}, r_i), C(J_j, r_i)]$ by machine M_q, i.e., the total processing time of job J_k by machine M_q in the interval $[C(J_{j-1}, r_i), C(J_j, r_i)]$ equals $\frac{v(J_k, M_q, J_j, r_i)}{s_q}$ and for any job J_k, equality $\sum_{q=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n} v(J_k, M_q, J_j, r_i) = p$ holds. The values $C(J_j, r_i)$, where $j = 1, \ldots, n, \quad i = 1, \ldots, n$, and the values $v(J_k, M_q, J_j, r_i)$, where $i, j = 1, \ldots, n, \quad k = j, \ldots, i, \quad q = 1, \ldots, m$, define a feasible solution of the following linear program. For convenience, we introduce $C(J_0, r_i) = r_i$.

$$\text{Minimize } \sum_{i=1}^{n}\Big((C(J_1, r_i) - r_i) + \cdots + (C(J_n, r_i) - r_i)\Big)$$

subject to

$$r_i = C(J_0, r_i) \leq C(J_1, r_i) \leq \cdots \leq C(J_{i+1}, r_i) = \cdots = C(J_n, r_i) = r_{i+1},$$

$$\sum_{q=1}^{m} \frac{v(J_k, M_q, J_j, r_i)}{s_q} \leq C(J_j, r_i) - C(J_{j-1}, r_i),$$

$$\sum_{k=1}^{n} \frac{v(J_k, M_q, J_j, r_i)}{s_q} \leq C(J_j, r_i) - C(J_{j-1}, r_i),$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{q=1}^{m} v(J_k, M_q, J_j, r_i) = p,$$

where $i = 1, \ldots, n, \quad j = 1, \ldots, n, \quad k = 1, \ldots, n, \quad q = 1, \ldots, m,$

$$C(J_j, r_i) \geq 0, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n,$$

and for $q = 1, \ldots, m$

$$v(J_k, M_q, J_j, r_i) = 0, \quad i = 1, \ldots, n-1, \quad j = i+1, \ldots, n, \quad k = 1, \ldots, n$$

$$v(J_k, M_q, J_j, r_i) = 0, \quad i = 1, \ldots, n, \quad j = 1, \ldots, i, \quad k = 1, \ldots, j-1,$$

$$v(J_k, M_q, J_j, r_i) = 0, \quad i = 1, \ldots, n-1, \quad j = 1, \ldots, i, \quad k = i+1, \ldots, n,$$

$$v(J_k, M_q, J_j, r_i) \geq 0 \quad i = 1, \ldots, n, \quad j = 1, \ldots, n, \quad k = 1, \ldots, n.$$

The above formulation includes $O(mn^3)$ variables and constraints, i.e., this problem can be polynomially solved. In [21], it has been shown that an optimal solution of the above linear program can be used to obtain an optimal schedule.

Problem $P \mid r_j, pmtn \mid \sum C_j$ has been proved to be unary NP-hard by a reduction from 3-partition in [1]. Problem $P \mid r_j, p_j = p, pmtn \mid \sum w_j C_j$ is unary NP-hard [28]. Thus, the minimal open problems are $1 \mid r_j, p_j = p, pmtn \mid \sum w_j C_j$ and $P \mid r_j, p_j = p, pmtn \mid \sum T_j$.

4 Summary

In Table 1, we give an overview on the main results for classical criteria.

Acknowledgements This work was partially supported by the Alexander von Humboldt Foundation.

References

1. P. Baptiste, P. Brucker, M. Chrobak, C. Dürr, S.A. Kravchenko, and F. Sourd, The complexity of mean flow time scheduling problems with release times, J. Scheduling, 10, 139–146, 2007.
2. P. Baptiste, P. Brucker, S. Knust, and V. G. Timkovsky, Ten notes on equal-processing-time scheduling, 4OR, 2, 111–127, 2004.
3. P. Baptiste, and P. Brucker, Scheduling equal processing time jobs: a survey, In Y.T. Leung, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, 78–96, CRC Press LLC, Boca Raton, FR, 2004.
4. P. Baptiste, M. Chrobak, C. Dürr, W. Jawor, and N. Vakhania, Preemptive scheduling of equal-length jobs to maximize weighted throughput, Oper. Res. Letters, 32, 258–264, 2004.
5. P. Baptiste, Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine with equal processing times, Journal of Scheduling, 2 (6), 245-252, 1999.
6. P. Baptiste, Scheduling equal-length jobs on identical parallel machines, Discrete Appl. Math., 103, 21–32, 2000.
7. P. Brucker, M. R. Garey, and D. S. Johnson, Scheduling equal-length tasks under tree-like precedence constraints to minimize maximum lateness, Math. Oper. Res., 2, 275–284, 1977.

8. P. Brucker, S. Knust, Complexity results for scheduling problems, http://www.mathematik.uni-osnabrueck.de/research/OR/class/
9. P. Brucker, S.A. Kravchenko, Scheduling equal processing time jobs to minimize the weighted number of late jobs, Journal of Mathematical Modelling and Algorithms, 5(2), 143–165, 2006.
10. P. Brucker, S.A. Kravchenko, Scheduling jobs with equal processing times and time windows on identical parallel machines, J. Scheduling, 11, 229–237, 2008.
11. P. Brucker, S.A. Kravchenko, Scheduling jobs with release times on parallel machines to minimize total tardiness, Universität Osnabrück, Preprint Heft 258, 13 pp., 2005.
12. M. Chrobak, C. Dürr, W. Jawor, Ł. Kowalik, and M. Kurowski, A note on scheduling equal-length jobs to maximize throughput, J. Scheduling, 9, 71–73, 2006.
13. M.I. Dessouky, B.J. Lageweg, J.K. Lenstra, and S.L. van de Velde, Scheduling identical jobs on uniform parallel machines, Statistica Neerlandica, 44, 115–123, 1990.
14. C. Dürr, The scheduling zoo, http://www.lix.polytechnique.fr/∼durr/query/
15. C. Dürr, M. Hurrand, Finding total unimodularity in optimization problems solved by linear programs, *Proc. of the 14th Annual European Symposium on Algorithms (ESA)*, 315–326, 2006.
16. M. Fujii, T. Kasami, and K. Ninomiya, Optimal sequencing of two equivalent processors, SIAM J. Appl. Math., 17, 234–248, 1969.
17. M.R. Garey, D.S. Johnson, B.B. Simons, and R.E. Tarjan, Scheduling unit-time tasks with arbitrary release times and deadlines, J. Comput., 10, 256–269, 1981.
18. S.A. Kravchenko, On the complexity of minimizing the number of late jobs in unit time open shop, Discrete Appl. Math., 100, 127–132, 2000.
19. S.A. Kravchenko, F. Werner, Minimizing the number of machines in a unit-time scheduling problem, Otto-von-Guericke-Universität Magdeburg, FMA, Preprint 25/07, 8 pp., 2007 (to appear in European J. Oper. Res., doi: 10.1016/j.ejor.2008.10.008).
20. S.A. Kravchenko, F. Werner, On a parallel machine scheduling problem with equal processing times, Discrete Appl. Math., 157, 848–852, 2009.
21. S.A. Kravchenko, F. Werner, Preemptive scheduling on uniform machines to minimize mean flow time, Comput. Oper. Res., Vol. 36(10), 2816–2821, 2009.
22. E.L. Lawler, Knapsack-like scheduling problems, the Moore-Hodgson algorithm and the 'tower of set' property, Math. Comput. Modelling, 20(2), 91–106, 1994.
23. E.L. Lawler, Recent results in the theory of machine scheduling, In A. Bachem, *Mathematical Programming: the state of the Art*, 202–234, Springer, Berlin, 1982.
24. E. L. Lawler, J. Labetoulle, On preemptive scheduling of unrelated parallel processors by linear programming, J. Assoc. Comput. Mach., 25(4), 612–619, 1978.
25. J. Labetuolle, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan, Preemptive scheduling of uniform machines subject to release dates, in H.R. Pulleyblank, *Progress in Combinatorial Optimization*, 245–261, Academic Press, New York, 1984.
26. J.K. Lenstra, A.G.H. Rinnooy Kan, and P. Brucker, Complexity of machine scheduling problems, Ann. Discrete Math., 1, 343–362, 1977.
27. J.K. Lenstra, A.H.G. Rinnooy Kan, Complexity of scheduling under precedence constraints, Oper. Res., 26, 22–35, 1978.
28. J.Y.-T. Leung, G.H. Young, Preemptive scheduling to minimize mean weighted flow time, Information Processing Letters, 34, 47–50, 1990.
29. B. Simons, A fast algorithm for single processor scheduling, *Proc. IEEE 19th Annual Symposium on Foundations of Computer Science (FOCS'78)*, 246–252, 1978.
30. B. Simons, Multiprocessor scheduling of unit-time jobs with arbitraty release times and deadlines, SIAM J. Comput., 12, 7–9, 1983.
31. B.B. Simons, M. Sipser, On scheduling unit-length jobs with multiple release time/deadline intervals, Oper. Res., 32, 80–88, 1984.
32. B.B. Simons, M.K. Warmuth, A fast algorithm for multiprocessor scheduling of unit-length jobs, SIAM J. Comput., 18, 690–710, 1989.
33. Z. Tian, C.T. Ng, T.C.E. Cheng, An $O(n^2)$ algorithm for scheduling equal-length preemptive jobs on a single machines to minimize total tardiness, J. Scheduling, 9, 343–364, 2006.
34. J.D. Ullman, Complexity of sequencing problems, in J.L. Bruno, E.G. Coffman, Jr., R.L. Graham, W.H. Kohler, R. Sethi, K. Steiglitz, and J.D. Ullman, *Computer and Job/Shop Scheduling Theory*, John Wiley & Sons Inc., New York, 1976.
35. J.D. Ullman, NP-complete scheduling problems, J. Comput. System Sci., 10, 384–393, 1975.

Table 1: Polynomially solvable and NP-hard problems

Non-preemptive problems	
$1\|r_j, D_j\|-$	unary NP-hard [26]
$P\|p_j = 1, D_j, prec\|-$	unary NP-hard [35]
$P \mid p_j = 1, D_j = 3, prec \mid -$	unary NP-hard [27]
$P \mid p_j = 1, D_j = 2, prec \mid -$	solvable in $O(n)$ [27]
$P\|r_j, p_j = p, D_j\|-$	solvable in $O(mn^2)$ [32]
$1\|r_j, p_j = p, D_j, prec\|C_{max}$	solvable in $O(n \log n)$ [17]
$P2\|p_j = 1, prec\|C_{max}$	solvable in $O(n^3)$ [16]
$P \mid r_j, p_j = p, outtree \mid C_{max}$	solvable in $O(n \log n)$ [7]
$P \mid r_j, p_j = p, intree \mid C_{max}$	unary NP-hard [7]
$P\|r_j, p_j = p, D_j\|C_{max}$	solvable in $O(mn^2)$ [32]
$Q\|r_j, p_j = p\|C_{max}$	solvable in $O(n \log n)$ [13]
$Q\|p_j = p\|C_{max}$	solvable in $O(n + m \log m)$ [13]
$Q\|p_j = p\|L_{max}$	solvable in $O(n \log n)$ [13]
$P\|r_j, p_j = p, D_j\|L_{max}$	solvable in $O(mn^4)$ [29]
$P\|r_j, p_j = p, D_j\|\sum C_j$	solvable in $O(mn^2)$ [32]
$Q\|r_j, p_j = p\|\sum C_j$	solvable in $O(mn^{2m+1})$ [13]
$Q\|p_j = p\|\sum C_j$	solvable in $O(n + m \log m)$ [13]
$P\|r_j, p_j = p\|\sum w_j C_j$	reducible to LP [10]
$Q\|p_j = p\|\sum w_j C_j$	solvable in $O(n \log n)$ [13]
$1\|r_j, p_j = p\|\sum T_j$	solvable in $O(n^7)$ [6]
$P\|r_j, p_j = p\|\sum T_j$	reducible to LP [11]
$Q\|p_j = p\|\sum T_j$	solvable in $O(n \log n)$ [13]
$Q\|p_j = p\|\max w_j T_j$	solvable in $O(n \log^2 n)$ [13]
$1 \mid r_j, p_j = p \mid \sum U_j$	solvable in $O(n^5)$ [12]
$1 \mid r_j, p_j = p \mid \sum w_j U_j$	solvable in $O(n^7)$ [5]
$Pm \mid r_j, p_j = p \mid \sum w_j U_j$	solvable in $O(n^{6m+1})$ [2]
$Q\|p_j = p\|\sum w_j U_j$	solvable in $O(n \log n)$ [13]
Preemptive problems	
$P \mid p_j = 1, prec, pmtn \mid \sum C_j$	unary NP-hard [34]
$P \mid r_j, pmtn \mid \sum C_j$	unary NP-hard [1]
$Q \mid r_j, p_j = p, pmtn \mid \sum C_j$	reducible to LP [21]
$P \mid r_j, p_j = p, pmtn \mid \sum w_j C_j$	unary NP-hard [28]
$R \mid r_j, pmtn \mid L_{max}$	reducible to LP [24]
$1 \mid r_j, p_j = p, pmtn \mid \sum U_j$	solvable in $O(n \log n)$ [22]
$P \mid r_j, p_j = p, pmtn \mid \sum U_j$	unary NP-hard [18]
$P \mid pmtn \mid \sum U_j$	binary NP-hard [23]
$Q \mid p_j = p, pmtn \mid \sum U_j$	solvable in $O(n \log^2 n + mn \log n)$ [2]
$1 \mid r_j, p_j = p, pmtn \mid \sum w_j U_j$	solvable in $O(n^4)$ [4]
$P \mid p_j = p, pmtn \mid \sum w_j U_j$	unary NP-hard [9]
$1 \mid r_j, p_j = p, pmtn \mid \sum T_j$	solvable in $O(n^2)$ [33]