Scheduling Jobs with Equal Processing Times

Svetlana A. Kravchenko [*] **Frank Werner** [**]

[*] *United Institute of Informatics Problems, Minsk, 220012 Belarus (Tel: 375-17-2842125; e-mail: kravch@newman.bas-net.by).*
[**] *Fakultät für Mathematik, Otto-von-Guericke-Universität, Magdeburg, 39106 Germany (Tel: +49 391 6712025; frank.werner@mathematik.uni-magdeburg.de)*

Abstract: Whereas the overwhelming majority of scheduling problems appears to be NP-hard, models with equal processing time jobs form a remarkable case which is still open for most problems but it intuitively looks polynomially solvable. The basic scheduling problem we are dealing with is the following. There are n jobs, each requiring an identical execution time p. There are associated a release time r_j and a deadline D_j with each job. All data are assumed to be integers. The aim is to construct a feasible schedule so as to minimize a given criterion. In this paper, we survey existing approaches for the problem considered, and for various machine environments.

1. INTRODUCTION

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. The basic scheduling problem we are dealing with is the following. There are n jobs, each requiring an identical execution time p. With each job, there are associated a release time r_j and a deadline D_j. All data are assumed to be integers. The objective is to construct a feasible schedule so as to minimize a given criterion for various machine environments.

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 monotonous criterion. Nevertheless, since prime conflicts are caused by overlapping intervals most of the problems have an open complexity status.

In this paper, we survey existing approaches for the considered model and expose problems with 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, r_j, and D_j since, in the case of $p = 1$, the main conflicts among overlapping places for processing jobs disappear. We refer the reader to Baptiste and Brucker [2004] for a survey.

2. CLASSIC CRITERIA

If the processing times can be arbitrary, problem $1|r_j, D_j|-$ is NP-hard in the strong sense by a polynomial reduction from the 3-partition problem, see Lenstra et al. [1977].

In Garey et al. [1981], an $O(n \log n)$ algorithm has been proposed for problem $1|r_j, p_j = p, D_j, prec|C_{max}$. More precisely, they 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 $1 \mid r_j, p_j = p, D_j \mid C_{max}$, Garey et al. [1981] proposed the concept of forbidden regions (intervals), i.e., open intervals where no job can start. The set of all forbidden regions is formed iteratively. The main used observation 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 ignoring all the values $r_1, \ldots, r_k$ and $D_1, \ldots, D_k$, i.e., supposing that all $r_1, \ldots, r_k$ are equal to r_i and all $D_1, \ldots, D_k$ are equal to D_j, we 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 the forbidden intervals. In other words, we schedule k jobs of duration p in $[e, D_j]$ under the condition that some intervals in $[e, D_j]$ are forbidden for starting the jobs and try to maximize the value of e. This can be done in $O(n)$ steps. Now, if $e < r_i$, then there is no feasible schedule for the original problem $1 \mid r_j, p_j = p, D_j \mid C_{max}$. However, if $r_i \le e < r_i + 1$, 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 Ullman [1975], it has been shown that problem $P|p_j = 1, D_j, prec|-$ is NP-hard in the strong sense by a polynomial reduction from the 3-satisfiability problem.

Lenstra and Rinnooy Kan [1978] have shown that even problem $P \mid p_j = 1, D_j = 3, prec \mid -$ is NP-hard in the strong sense. 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 problem $P2 \mid p_j = 1, prec \mid C_{max}$ was solved in $O(n^3)$ time by Fujii et al. [1969] by a reduction of the problem to the maximum matching problem.

In Brucker et al. [1977], 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 NP-hard in the strong sense.

In Simons [1983], 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 $(1, t_1)$, …, (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 Simons [1978], 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 Simons and Warmuth [1989], 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 Simons [1983] and Garey et al. [1981].

Let x_{ji} be equal to the amount of job J_j processed in the interval I_i, where

$$\{I_i \mid i \in \{1, \ldots, z\}\}$$
$$= \{[r_j + kp, r_j + kp + p\ [\ \mid k \in \{-n, \ldots, -1, 0, 1, \ldots, n\}\},$$

and

$$y = \max\{k \mid I_{i+1} \cap \ldots \cap I_{i+k} \neq \emptyset, i \in \{0, \ldots, z-k\}\}.$$

Then, see Brucker and Kravchenko [2008], the following linear programming formulation can be proposed for problem $P|r_j, p_j = p, D_j|-$:

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

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

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

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

In the above system, the polyhedron (1) and for each $i = 0, \ldots, z-y$, with e such that $i + ey \leq z$ holds, the polyhedron

$$\sum_{j=1}^{n} x_{j,i+1} + \ldots + \sum_{j=1}^{n} x_{j,i+y} \leq mp$$
$$\sum_{j=1}^{n} x_{j,i+y+1} + \ldots + \sum_{j=1}^{n} x_{j,i+2y} \leq mp$$
$$\cdots$$
$$\sum_{j=1}^{n} x_{j,i+(e-1)y+1} + \ldots + \sum_{j=1}^{n} x_{j,i+ey} \leq mp$$

is integer, since the corresponding matrices are network matrices and therefore, they are totally unimodular. Nevertheless, their intersection is not an integer polyhedron and therefore, the obtained solution is not necessarily integer.

Using an obtained solution x^*_{ji}, one can construct an optimal schedule in two equivalent ways, namely:

1. With the help of x^*_{ji}, 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 x^*_{ji} into the form $x^*_{ji} \in \{0, p\}$ in a straightforward way without marking the occupied intervals. 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.

The proposed algorithm is not so fast as the algorithm given in Simons and Warmuth [1989] but it can be applied to more general problems.

It has been shown 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:

Minimize
$$\sum_{i=1}^{z}\sum_{j=1}^{n} D(I_i)x_{ji}$$
subject to (1), (2), (3), (4).

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

In Dürr and Hurrand [2006], the proposed linear programming formulation was transformed by means of the substitution
$$g_t = \sum_{s=1}^{t}\sum_{j=1}^{n} x_{js}$$
into the following form:

Minimize
$$\sum_{i=1}^{z} D(I_i)(g_i - g_{i-1})$$
subject to
$$\begin{gathered} g_z - g_0 = n \\ g_i - g_{i-1} \geq 0, \quad i = 1, \ldots, n \\ g_i - g_{i-y+1} \leq m, \quad i = y, \ldots, n \\ g_0 = 0. \end{gathered}$$

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

To solve problem $P|r_j, p_j = p|\sum w_j C_j$, see Brucker and Kravchenko [2008], it is sufficient to solve the following linear program:

Minimize
$$\sum_{i=1}^{z}\sum_{j=1}^{n} w_j D(I_i)x_{ji}$$
subject to (1), (2), (3), (4).

In this case D_j is any large number, for instance $D_j = \max_j\{r_j\} + np$.

In Brucker and Kravchenko [2005], it has been shown that, in order to solve problem $P|r_j, p_j = p|\sum T_j$, it is sufficient to minimize
$$\sum_{i=1}^{z}\sum_{j=1}^{n} \max\{0, D(I_i) - d_j\}\, x_{ji}$$
subject to (1), (2), (3), (4).

In Baptiste et al. [2004], 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 $\{r_j + kp \mid k \in \{0, \ldots, n\}, j \in \{0, \ldots, n\}\}$.
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 $a_i \in \{r_j + kp \mid k \in \{0, \ldots, n\}, j \in \{0, \ldots, n\}\}$, and $\max\{a_1, \ldots, a_m\} - \min\{a_1, \ldots, a_m\} \leq p$. For an optimal schedule, if job J_j starts at t_j, there is a corresponding profile a with $t_j \in \{a_1, \ldots, a_m\}$.
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
$$W_k[a, b] = \max\{W'_k[a, b], W_{k-1}[a, b]\}$$
if
$$\max\{a_1, \ldots, a_m\} - p \leq r_k < \min\{b_1, \ldots, b_m\},$$
and
$$W_k[a, b] = W_{k-1}[a, b]$$
if
$$r_k \notin [\max\{a_1, \ldots, a_m\} - p, \min\{b_1, \ldots, b_m\}[,$$
where
$$W'_k[a, b] = \max\{W_{k-1}[a, x] + w_k + W_{k-1}[x', b]\}.$$
In the last formula the maximum is taken over all such x for which $\min\{x_1, \ldots, x_m\} \in [r_k, d_k - p]$ and the profile x' is obtained from profile x by adding one job.

In Chrobak et al. [2006], an $O(n^5)$ algorithm was proposed for problem $1 \mid r_j, p_j = p \mid \sum U_j$. Their algorithm is analogous to the algorithm of Baptiste et al. [2004], however, they do not use $W_k[a, b]$ but $w_k[a, B]$, i.e., for the given w_k, k, a, they 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 $r_j \geq a$ 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 $[a + p, b]$.

3. SOME GENERALIZATIONS

In Baptiste [2000], 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

1. f_j is non-decreasing,
2. $f_i - f_k$ is monotonic.

Note that both classical criteria $\sum w_j C_j$ and $\sum T_j$ can be described in such a way. In that paper, the following observations and definitions are used:

1. Jobs $J_1, \ldots, J_n$ are ordered in such a way that for any pair $i > j$, function $f_i - f_j$ is non-decreasing.
2. The starting times of the jobs in an optimal schedule belong to the set
$$\{r_j + kp \mid k \in \{0, \ldots, n\}, j \in \{0, \ldots, n\}\}.$$
3. Note that here the profile a indicates the number of machines available at each time point in the interval $[a_1, a_m]$.
4. The set $U_k(a, b)$ is defined as the set
$$\{J_j \mid j \leq k, r_j \in [a_m - p, b_1[\}.$$
5. $F_k(a, b)$ is defined as the minimal value of
$$\sum_{J_j \in U_k(a_m - p, b_1)} f_j(C_j)$$
among all possibilities of scheduling all jobs from the set $U_k(a_m - p, b_1)$ between the two profiles a and b.

Then dynamic programming is realized by the following formulas:

$$F_k(a,d) = F_{k-1}(a,d) \text{ if } r_k \notin [a_m - p, d_1[$$

and

$$F_k(a,d) = \min_b \{F_{k-1}(a,b) + F_{k-1}(c,d) + f_k(b_1+p) \mid r_k < b_1, \quad c = (b_2,\dots,b_m,b_1+p)\}$$

if $r_k \in [a_m - p, d_1[$.

The overall complexity of the proposed algorithm is $O(n^{3m+4})$.

In Kravchenko and Werner [2009], 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 Baptiste [2000]. Here f_j depends on the completion time C_j, such that f_j is non-decreasing, and $f_i - f_k$ is monotonic. For problem $P \mid r_j, p_j = p, D_j \mid \max \varphi_j$, 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 Brucker and Kravchenko [2005] and consists in minimizing

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

subject to

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

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

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

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

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:

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

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

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

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

$$x_{ji} \ge 0, \quad i=1,\dots,z, \quad j=1,\dots,n$$

It is possible to find a solution for the above problem such that $x_{ji} \in \{0,p\}$. At the same time the obtained solution can be considered as a solution for problem $P \mid r_j, p_j = p, D_j \mid \max \varphi_j(C_j) \le F$. 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 Simons and Sipser [1984], the following problem has been considered. There are n jobs, each requiring an identical execution time p without preemptions. With each job, a set of intervals is associated. 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 NP-hard in the strong sense for the case of one machine and two prescribed intervals for each job by a polynomial reduction from the 3-satisfiability problem. From our point of view, this result is surprising since in the case of one prescribed interval the problem is polynomially solvable. We give an NP-hardness proof for a weaker problem, namely when the number of prescribed intervals is not limited.

The 3-satisfiability problem consists of a set of Boolean variables and a set of clauses. It is easy to make the number of occurrences of variable a is equal to the number of occurrences of variable $\bar{a}$ by adding clauses $(a \vee a \vee \bar{a})$ or $(\bar{a} \vee \bar{a} \vee a)$ to the problem.

1. Let $p = 2$ and n_a be the number of occurrences of variable a in the set of clauses, i.e., equality $n_a = n_{\bar{a}}$ holds. For each pair of variables a and $\bar{a}$, we create a set of intervals $\{[t_a, t_a+2[, [t_a+1, t_a+3[, \dots, [t_a + 2n_a - 1, t_a + 2n_a + 1[\}$. Each interval $[t_a + 2k, t_a + 2k + 2[$ is prescribed to some occurrence of variable a. Each interval $[t_a + 2k + 1, t_a + 2k + 2 + 1[$ is prescribed to some occurrence of variable $\bar{a}$.
2. For each clause c, we have a job J_c which can be processed in any of three intervals prescribed to the variables from the clause c.
3. For each set $\{[t_a, t_a+2[, [t_a+1, t_a+3[, \dots, [t_a + 2n_a - 1, t_a + 2n_a + 1[\}$, we create a job J_a which can be processed in one of two intervals, either in $[t_a - 1, t_a + 1[$ or in $[t_a + 2n_a, t_a + 2n_a + 2[$.
4. Besides we have $\sum_a n_a - z$ jobs that can be scheduled in any of the intervals $\{[t_a, t_a+2[, [t_a+1, t_a+3[, \dots, [t_a + 2n_a + 1, t_a + 2n_a[\}$, where z is the number of clauses.

Now choose t_a such that the sets of intervals

$$\{[t_a - 1, t_a + 1[, [t_a, t_a + 2[, \dots, [t_a + 2n_a, t_a + 2n_a + 2[\}$$

do not overlap each other.

Thus, we have $2\sum_a n_a + \sum_a 2$ intervals and $\sum_a n_a + \sum_a 1$ jobs.

One can prove that, if there is a solution to the 3-satisfiability problem, then the scheduling problem under consideration has a solution:

If variable a is true, then job J_a is processed in $[t_a + 2n_a, t_a + 2n_a + 2[$, otherwise job J_a is processed in $[t_a - 1, t_a + 1[$. For each clause c, job J_c is processed in one of the intervals prescribed to some true variable. All other jobs are scheduled in any feasible way.

However, if there is a solution to the scheduling problem considered, then for each variable a either intervals prescribed to a or intervals prescribed to $\bar{a}$ will be occupied. Moreover, for each clause, there is at least one interval corresponding to the true variable. Therefore, there is a solution to the 3-satisfiability problem.

In Kravchenko and Werner [2007], the following problem has been considered. There are n jobs $J_1, \ldots, J_n$ which have to be processed on a set of identical parallel machines. For each job J_j, $j = 1, \ldots, n$, a processing time $p_j = p$, which is equal for all jobs, a release date r_j, and a deadline D_j are given. Each machine can process only one job at a time. 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 is reduced to the following linear programming problem.

$$\text{minimize } M$$

subject to

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

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

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

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

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

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

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.

In Dessouky et al. [1990], 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

$$\max \varphi_j = \max_{1 \leq j \leq n} \varphi_j(C_j),$$

and $\varphi_j, j = 1, \ldots, n$, are non-decreasing functions in the job completion times. They also indicated how to reduce the complexity for classic 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.

4. CONCLUSIONS

In the following table, we give an overview on the results for classic criteria.

$1\|r_j, D_j\|-$ Lenstra et al. [1977]	NP-hard in the strong sense
$1\|r_j, p_j = p, D_j, prec\|C_{max}$ Garey et al. [1981]	$O(n \log n)$
$P\|p_j = 1, D_j, prec\|-$ Ullman [1975]	NP-hard in the strong sense
$P2\|p_j = 1, prec\|C_{max}$ Fujii et al. [1969]	$O(n^3)$
$P \mid p_j = 1, D_j = 3, prec \mid -$ Lenstra and Rinnooy Kan [1978]	NP-hard in the strong sense
$P \mid p_j = 1, D_j = 2, prec \mid -$ Lenstra and Rinnooy Kan [1978]	$O(n)$
$P \mid r_j, p_j = p, outtree \mid C_{max}$ Brucker et al. [1977]	$O(n \log n)$
$P \mid r_j, p_j = p, intree \mid C_{max}$ Brucker et al. [1977]	NP-hard in the strong sense
$P\|r_j, p_j = p, D_j\|-$ Simons [1983]	$O(n^3 \log\log n)$
$P\|r_j, p_j = p, D_j\|L_{max}$ Simons [1978]	$O(mn^4)$
$P\|r_j, p_j = p, D_j\|-$ Simons and Warmuth [1989]	$O(mn^2)$
$P\|r_j, p_j = p, D_j\|C_{max}$ Simons and Warmuth [1989]	$O(mn^2)$
$P\|r_j, p_j = p, D_j\|\sum C_j$ Simons and Warmuth [1989]	$O(mn^2)$
$P\|r_j, p_j = p, D_j\|-$ Brucker and Kravchenko [2008]	LP

$P\|r_j, p_j = p, D_j\| \sum C_j$ Brucker and Kravchenko [2008]	LP
$P\|r_j, p_j = p, D_j\| \sum C_j$ Dürr and Hurrand [2006]	$O(n^4)$
$P\|r_j, p_j = p\| \sum w_j C_j$ Brucker and Kravchenko [2008]	LP
$P\|r_j, p_j = p\| \sum T_j$ Brucker and Kravchenko [2005]	LP
$Pm \mid r_j, p_j = p \mid \sum w_j U_j$ Baptiste et al. [2004]	$O(n^{6m+1})$
$1 \mid r_j, p_j = p \mid \sum U_j$ Chrobak et al. [2006]	$O(n^5)$
$Q\|r_j, p_j = p\|C_{max}$ Dessouky et al. [1990]	$O(n \log n)$
$Q\|r_j, p_j = p\| \sum C_j$ Dessouky et al. [1990]	$O(mn^{2m+1})$
$Q\|p_j = p\|C_{max}$ and $Q\|p_j = p\| \sum C_j$ Dessouky et al. [1990]	$O(n + m \log m)$
$Q\|p_j = p\| \sum w_j C_j$, $Q\|p_j = p\|L_{max}$ $Q\|p_j = p\| \sum T_j$, $Q\|p_j = p\| \sum w_j U_j$ Dessouky et al. [1990]	$O(n \log n)$
$Q\|p_j = p\| \max w_j T_j$ Dessouky et al. [1990]	$O(n \log^2 n)$

The most interesting open problems are the following ones:

- $P3 \mid p_j = 1, \text{ prec} \mid C_{max}$,
- $P \mid r_j, p_j = p \mid \sum U_j$,
- $P \mid r_j, p_j = p, D_j \mid \sum w_j C_j$,
- $Q \mid r_j, p_j = p, D_j \mid -$,
- $Q \mid r_j, p_j = p \mid \sum C_j$ for an arbitrary number of machines.

ACKNOWLEDGEMENTS

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

REFERENCES

P. Baptiste, P. Brucker, S. Knust, and V. G. Timkovsky. Ten notes on equal-processing-time scheduling. *4OR*, 2: 111–127, 2004.

P. Baptiste, and P. Brucker. Scheduling equal processing time jobs: a survey. In Y.T. Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, pages 78–96. CRC Press LLC, Boca Raton, FR, 2004.

P. Baptiste. Scheduling equal-length jobs on identical parallel machines. *Discrete Appl. Math.*, 103:21–32, 2000.

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.

P. Brucker, S.A. Kravchenko. Scheduling jobs with equal processing times and time windows on identical parallel machines. *J. Scheduling*, 11:229–237, 2008.

P. Brucker, S.A. Kravchenko. Scheduling jobs with release times on parallel machines to minimize total tardiness. Universität Osnabrück, *Preprint Heft 258*, 13 p., 2005.

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.

M.I. Dessouky, B.J. Lageweg, J.K. Lenstra, and S.L. van de Velde. Scheduling identical jobs on uniform parallel machines. *Stat. Neerlandica*, 44:115–123, 1990.

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.

M. Fujii, T. Kasami, and K. Ninomiya. Optimal sequencing of two equivalent processors. *SIAM J. Appl. Math.*, 17:234–248, 1969.

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.

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 p., 2007.

S.A. Kravchenko, F. Werner. On a parallel machine scheduling problem with equal processing times. *Discrete Appl. Math.*, 157:848–852, 2009.

J.K. Lenstra, A.G.H. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Ann. Discrete Math.*, 1:343–362, 1977.

J.K. Lenstra, A.H.G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Oper. Res.*, 26:22–35, 1978.

B. Simons. A fast algorithm for single processor scheduling. *Proc. IEEE 19th Annual Symposium on Foundations of Computer Science (FOCS'78)*, 246–252, 1978.

B. Simons. Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines. *SIAM J. Comput.*, 12:7–9, 1983.

B.B. Simons, M. Sipser. On scheduling unit-length jobs with multiple release time/deadline intervals. *Oper. Res.*, 32:80–88, 1984.

B.B. Simons, M.K. Warmuth. A fast algorithm for multiprocessor scheduling of unit-length jobs. *SIAM J. Comput.*, 18:690–710, 1989.

J.D. Ullman. NP-complete scheduling problems. *J. Comput. System Sci.*, 10:384–393, 1975.