Minimizing the Number of Machines in a Unit-Time Scheduling Problem

Svetlana A. Kravchenko [1]
United Institute of Informatics Problems,
Surganova St. 6, 220012 Minsk, Belarus
kravch@newman.bas-net.by

Frank Werner
Otto-von-Guericke-Universität, Fakultät für Mathematik,
39106 Magdeburg, Germany
Frank.Werner@Mathematik.Uni-Magdeburg.De

August 23, 2007

Abstract

In this paper, we give a polynomial algorithm for the problem of minimizing the number of machines in a parallel machine environment with equal processing times of all jobs and arbitrarily given release dates and deadlines.

Keywords: parallel machine scheduling, linear programming

1 Introduction

The problem considered can be stated as follows. 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 the jobs, a release date r_j, and a deadline D_j are given. Each machine can process only one job at a time. 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 number

[1]Supported by the Alexander von Humboldt Foundation

of machines is minimal. In this paper, we present a polynomial time algorithm for the above problem.

Note that, if the number of machines is given, then the problem of finding a feasible schedule was considered in [4]. The criterion of minimizing the number of machines is very important in practice, see e.g. [3]. The algorithm, presented in this note, is based on the approach given in [1] and [2].

The remainder of this note is organized as follows. In Section 2, we give a linear programming (LP) formulation for a relaxation of the scheduling problem under consideration. A polynomial algorithm for the scheduling problem is presented in Section 3. Finally, we give the correctness proof for this algorithm in Section 4.

2 LP formulation

A feasible schedule for the problem considered can be found in the class of schedules, where each job is processed in one of the following intervals of length p :

$$\{[r_j + kp, r_j + kp + p \; [\; | \; k \in \{0, 1, \ldots\}\}.$$

On the other hand, if for some r_k and for any index $j < k$ the inequality $r_j + \sum_{r_j \le r_i < r_k} p_i \le r_k$ holds, then the problem can be decomposed into subproblems that can be solved independently from each other. Therefore, without loss of generality, we can suppose that in an optimal schedule any job is processed within the time interval $[\min_i\{r_i\}, \min_i\{r_i\} + 2np\,[$. Thus, we will restrict the possible positions of the jobs by the set of intervals

$$\{[r_j + kp, r_j + kp + p \; [\; | \; k \in \mathbb{Z}, r_j + kp \ge \min_i\{r_i\}, r_j + kp + p \le \min_i\{r_i\} + 2np\}.$$

Take all the different intervals from this set and enumerate them in increasing order of their left endpoints. Denote the obtained set by $\{I_i \mid i \in \{1, \ldots, z\}\}$ and for each I_i, denote by $D(I_i)$ the right endpoint of I_i and by $R(I_i)$ the left endpoint of I_i.

One can see that there exists some q such that $I_{i+1} \cap \ldots \cap I_{i+q} \neq \emptyset$ for any $i \in \{0, \ldots, z - q\}$. Set $y = \max\{q \mid I_{i+1} \cap \ldots \cap I_{i+q} \neq \emptyset, i \in \{0, \ldots, z - q\}\}$, i.e. y is the number of intersecting intervals.

Consider the following LP problem:

$$\text{minimize } M \tag{2.1}$$

subject to

$$\sum_{i=1}^{z} x_{ji} = p, \quad j = 1, \ldots, n \tag{2.2}$$

$$\sum_{j=1}^{n} x_{j,i+1} + \ldots + \sum_{j=1}^{n} x_{j,i+y} \le Mp, \quad i = 0, \ldots, z - y \tag{2.3}$$

$$x_{ji} = 0 \text{ if } R(I_i) < r_j \quad i = 1, \ldots, z, \quad j = 1, \ldots, n \tag{2.4}$$

$$x_{ji} = 0 \text{ if } D_j < D(I_i) \quad i = 1, \ldots, z, \quad j = 1, \ldots, n \tag{2.5}$$

$$x_{ji} \geq 0, \quad i = 1, \ldots, z, \quad j = 1, \ldots, n \tag{2.6}$$

If we set x_{ji} equal to the amount of job J_j processed in the interval I_i and set $m = \lceil M \rceil$ to be the number of machines, then any feasible schedule for the scheduling problem under consideration can be described as a feasible solution of problem (2.1) – (2.6). On the other hand, if the obtained solution of problem (2.1) – (2.6) is integer, then it is a solution of the scheduling problem considered.

In the following, we will suppose that problem (2.1) – (2.6) has been solved and that the obtained solution x^* is not integer. Parallel to the notation x_{ji}, we will use $x_{j,i}$ to avoid a confusion between the first index and the second one.

3 Algorithm

In this section, we present a polynomial algorithm for the scheduling problem under consideration. Using x^* and m, we determine all intervals where jobs are processed in a feasible schedule. With this purpose, we calculate the value $v(I_k)$ for each interval I_k as follows:

$$v(I_k) = \sum_{i=1}^{k} \sum_{j=1}^{n} x^*_{ji}, \qquad k = 1, \ldots, z.$$

Note that $v(I_z) = \sum_{i=1}^{z} \sum_{j=1}^{n} x^*_{ji} = np$ holds if the solution is feasible.

Since in any interval at most m jobs can be scheduled, we define the values:

$$v_1 = v(I_1), \ldots, v_m = v(I_1),$$

$$v_{m+1} = v(I_2), \ldots, v_{2m} = v(I_2),$$

$$\ldots$$

$$v_{(z-1)m+1} = v(I_z), \ldots, v_{zm} = v(I_z),$$

i.e. we take m copies of each interval enumerated in nondecreasing order of their left endpoints.

Now, using $v_1, \ldots, v_{zm}$, we mark all intervals where some jobs are processed in an optimal schedule. The marking procedure is as follows:

Step 1: To select the first marked interval, one moves from v_1 to v_{zm} and takes the first v_j such that $v_j > 0 \cdot p$ holds. Denote the corresponding interval by I^*_1, i.e. $v_j = v(I^*_1)$.

Step 2: To select the second marked interval, one moves from $v_j = v(I^*_1)$ to v_{zm} and takes the first v_g such that $v_g > 1 \cdot p$ holds. Denote the corresponding interval by I^*_2, i.e. $v_g = v(I^*_2)$.

...

Step n: To select the n-th marked interval, one moves from $v(I^*_{n-1})$ to v_{zm} and takes the first v_h such that $v_h > (n-1) \cdot p$ holds. Denote the corresponding interval by I^*_n, i.e. $v_h = v(I^*_n)$.

The marked intervals determine the places where the jobs are processed in an optimal schedule. Now, we use all marked intervals $I^*_1, \ldots, I^*_n$ and apply the following

Algorithm 1

`Consider the marked intervals` $I^*_1, \ldots, I^*_n$ `in the given order and schedule in each of these intervals a currently available job with the earliest deadline.`

4 Feasibility proof

Before proving that the schedule constructed by Algorithm 1 is a feasible one, we investigate some properties of the vector v. For a solution x^*, let $I_{i_1}, \ldots, I_{i_n}$ be all marked intervals and denote by e_{i_j} the number of marked copies of the interval $I_{i_j}, j = 1, \ldots, n$.

Lemma 1 *For any index $j \in \{1, \ldots, n\}$, inequalities*

$$(e_{i_1} + e_{i_2} + \ldots + e_{i_j})p \geq v(I_{i_j}) > (e_{i_1} + e_{i_2} + \ldots + e_{i_j} - 1)p$$

hold.

Proof: The proof is done by induction. By Algorithm 1, we have the following properties for $j = 1$:

- the first copy of I_{i_1} is marked since $v(I_{i_1}) > 0 \cdot p$,
- the second copy of I_{i_1} is marked since $v(I_{i_1}) > 1 \cdot p$,
- ...
- the $e_{i_1}-$th copy of I_{i_1} is marked since $v(I_{i_1}) > (e_{i_1} - 1)p$.

Thus, $v(I_{i_1}) > (e_{i_1} - 1)p$ holds. Moreover, if $e_{i_1} < m$, then by definition of e_{i_1}, the inequalities $e_{i_1}p \geq v(I_{i_1}) > (e_{i_1} - 1)p$ hold. On the other hand, $v(I_{i_1}) \leq mp$ holds since for all intervals I_i with $i < i_1$, the equality $x^*_{ji} = 0$ holds. Hence, if $e_{i_1} = m$, then $v(I_{i_1}) \leq e_{i_1}p$ holds. Thus, Lemma 1 is true for i_1.

Suppose that Lemma 1 holds for any $j \leq k$, where $k \leq n-1$, i.e.

$$(e_{i_1} + e_{i_2} + \ldots + e_{i_j})p \geq v(I_{i_j}) > (e_{i_1} + e_{i_2} + \ldots + e_{i_j} - 1)p.$$

We want to show that

$$(e_{i_1} + e_{i_2} + \ldots + e_{i_{k+1}})p \geq v(I_{i_{k+1}}) > (e_{i_1} + e_{i_2} + \ldots + e_{i_{k+1}} - 1)p$$

holds.

By Algorithm 1, we have the following properties:

- the first copy of $I_{i_{k+1}}$ is marked since $v(I_{i_{k+1}}) > (e_{i_1} + e_{i_2} + \ldots + e_{i_k})p$;
- the second copy of $I_{i_{k+1}}$ is marked since $v(I_{i_{k+1}}) > (e_{i_1} + e_{i_2} + \ldots + e_{i_k} + 1)p$;
- ...
- the $e_{i_{k+1}}-$th copy of $I_{i_{k+1}}$ is marked since $v(I_{i_{k+1}}) > (e_{i_1} + e_{i_2} + \ldots + e_{i_{k+1}} - 1)p$,

i.e. if $e_{i_{k+1}} < m$, then $v(I_{i_{k+1}}) \leq (e_{i_1} + e_{i_2} + \ldots + e_{i_{k+1}})p$ by definition of $e_{i_{k+1}}$.

Inequality

$$v(I_{i_k}) \leq (e_{i_1} + e_{i_2} + \ldots + e_{i_k})p \tag{4.1}$$

holds by the inductive assumption, and

$$v(I_{i_{k+1}}) = v(I_{i_k}) + \sum_{l=1}^{n} x^*_{l,i_{k+1}} \tag{4.2}$$

holds by definition. Moreover, inequality

$$\sum_{l=1}^{n} x^*_{l,i_{k+1}} \leq mp \tag{4.3}$$

holds by constraint (2.2). Combining (4.1) – (4.3), we have $v(I_{i_{k+1}}) \leq (e_{i_1}+e_{i_2}+\ldots+e_{i_k})p+mp$, i.e. if $e_{i_{k+1}} = m$, then $v(I_{i_{k+1}}) \leq (e_{i_1} + e_{i_2} + \ldots + e_{i_{k+1}})p$ holds. □

Theorem 1 *Algorithm 1 constructs a feasible schedule.*

Proof: To prove Theorem 1, it is sufficient to show that there is a feasible schedule in which all marked intervals are occupied.

Using vector x^*, we can calculate the restricted release date r^*_j and the restricted deadline D^*_j for each job J_j in the following way: $r^*_j = \min\{r(I_i) \mid x^*_{ji} \neq 0\}$, $D^*_j = \max\{D(I_i) \mid x^*_{ji} \neq 0\}$. It is clear that $[r^*_j, D^*_j[\subseteq [r_j, D_j[$ holds. Now we schedule in each marked interval the job with the minimal **restricted** deadline among all jobs that are currently available for processing, denote this schedule by $\tilde{s}$.

To prove Theorem 1, it is sufficient to show that $\tilde{s}$ is a feasible schedule. Determine $\tilde{x}$ in the following way: $\tilde{x}_{ji} = p$ if job J_j is scheduled in the interval I_i in $\tilde{s}$, and $\tilde{x}_{ji} = 0$ otherwise.

Now, we want to prove that $\tilde{x}$ is a feasible solution of problem (2.2) – (2.6). It is clear that $0 \leq \tilde{x}_{ji} \leq p$ for any $i = 1, \ldots, z$ and for any $j = 1, \ldots, n$. Thus, we have to prove (2.2), (2.3), (2.4), and (2.5).

First, we prove (2.2). We show that $\sum_{i=1}^{z} \tilde{x}_{ji} = p$ for any $j = 1, \ldots, n$. This means that any job is scheduled in some interval. Since we choose only jobs which are currently available for processing, this will also prove (2.4) and (2.5). Take any job J_k which is not scheduled. Let $r_k^* = r(I_\gamma)$, $D_k^* = D(I_\beta)$, see Figure 1. In $[r_k^*, D_k^*[$, there is at least one marked interval since $v(I_\beta) - v(I_\gamma) \geq p$. For all jobs J_j scheduled in these marked intervals, we have $D_j^* \leq D_k^*$ since otherwise job J_k would be scheduled but not job J_j. Besides all these marked intervals are occupied by some jobs.

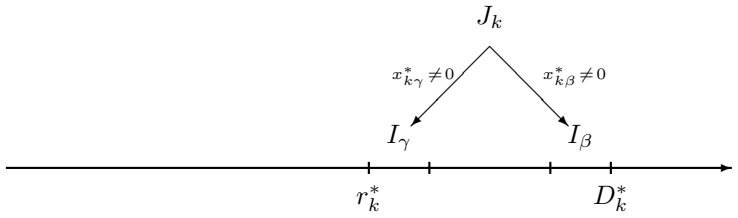

Figure 1: $r_k^* = r(I_\gamma), D_k^* = D(I_\beta)$, and job J_k is not scheduled

Take

$$r_u^* = \min\{r_j^* \mid J_j \text{ is scheduled in some marked interval within } [r_k^*, D_k^*[\}.$$

Consider $[r_u^*, r_k^*[$ and all marked intervals in it. For any job g scheduled in these marked intervals, inequality $D_g^* \leq D_k^*$ holds since otherwise job J_u would be scheduled but not job J_g. Again, all these marked intervals are occupied by some jobs because otherwise J_u could have been scheduled earlier. Take

$$r_w^* = \min\{r_j^* \mid J_j \text{ is scheduled in some marked interval within } [r_u^*, r_k^*[\}.$$

Now, we consider $[r_w^*, r_u^*[$ and all marked intervals in it, and so on. At the end of this process, we find an interval $I^* = [r_\tau^*, D_k^*[$ such that

- all marked intervals in I^* are occupied by some jobs;
- for all, say N, jobs scheduled in I^*, we have $[r_j^*, D_j^*[\subseteq [r_\tau^*, D_k^*[$;
- job J_k is not scheduled in I^* but inclusion $[r_k^*, D_k^*[\subseteq [r_\tau^*, D_k^*[$ holds.

Denote all marked intervals in $[r_\tau^*, D_k^*[$ by $I_{i_{q+1}}, I_{i_{q+2}}, \ldots, I_{i_{q+f}}$ and the corresponding numbers of marked copies by $e_{i_{q+1}}, e_{i_{q+2}}, \ldots, e_{i_{q+f}}$. We also consider I_{i_q}, i.e. the last marked interval before $I_{i_{q+1}}$.

By Lemma 1, we have

$$(e_{i_1} + e_{i_2} + \ldots + e_{i_q})p \geq v(I_{i_q}) > (e_{i_1} + e_{i_2} + \ldots + e_{i_q} - 1)p.$$

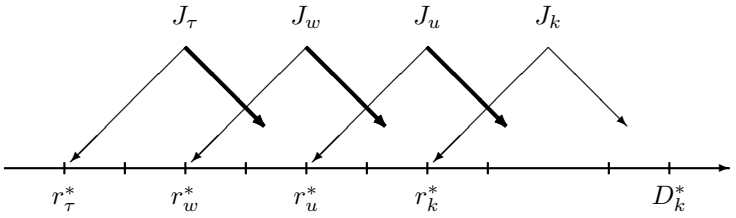

Figure 2: Jobs J_τ, J_w, J_u are scheduled in $[r^*_\tau, D^*_k[$

Let us assume that $v(I_{i_q})$ is equal to $(e_{i_1} + e_{i_2} + \ldots + e_{i_q} - 1)p + \delta$, where $0 < \delta \leq p$, and

$$(e_{i_1} + e_{i_2} + \ldots + e_{i_{q+f}})p \geq v(I_{i_{q+f}}) > (e_{i_1} + e_{i_2} + \ldots + e_{i_{q+f}} - 1)p.$$

Moreover, let $v(I_{i_{q+f}}) = (e_{i_1} + e_{i_2} + \ldots + e_{i_{q+f}} - 1)p + \epsilon$, where $0 < \epsilon \leq p$.

Then, for the intervals $I_{i_{q+1}}, I_{i_{q+2}}, \ldots, I_{i_{q+f}}$, we obtain

$$\sum_{j=1}^{n} x^*_{j,i_{q+1}} + \sum_{j=1}^{n} x^*_{j,i_{q+2}} + \ldots + \sum_{j=1}^{n} x^*_{j,i_{q+f}} = (N+1)p$$

$$= v(I_{i_{q+f}}) - v(I_{i_q}) = (e_{i_{q+1}} + e_{i_{q+2}} + \ldots + e_{i_{q+f}})p + \epsilon - \delta,$$

since $e_{i_{q+1}} + e_{i_{q+2}} + \ldots + e_{i_{q+f}} = N$. Hence, we have $\epsilon - \delta = p$, i.e. $\epsilon = p$ and $\delta = 0$ but δ has to be positive by our assumption. Thus, we get a contradiction with the statement that J_k is not scheduled by Algorithm 1.

Now we want to prove (2.3), i.e. we show that

$$\sum_{j=1}^{n} \tilde{x}_{j,i+1} + \sum_{j=1}^{n} \tilde{x}_{j,i+2} + \ldots + \sum_{j=1}^{n} \tilde{x}_{j,i+y} \leq mp$$

holds for any $i = 1, \ldots, z - y$. Suppose that among the intervals $I_{i+1}, I_{i+2}, \ldots, I_{i+y}$, the marked intervals are the following ones: I_{i_c}, $I_{i_{c+1}}, \ldots, I_{i_g}$ with the numbers of marked copies $e_{i_c}, e_{i_{c+1}}, \ldots, e_{i_g}$ correspondingly. Thus, we have to show that $(e_{i_c} + e_{i_{c+1}} + \ldots + e_{i_g})p \leq mp$.

By Lemma 1, we have $(e_{i_1} + e_{i_2} + \ldots + e_{i_{c-1}})p \geq v(I_{i_{c-1}})$ and $v(I_{i_g}) > (e_{i_1} + e_{i_2} + \ldots + e_{i_g} - 1)p$. Thus,

$$(e_{i_c} + e_{i_{c+1}} + \ldots + e_{i_g})p = (e_{i_1} + e_{i_2} + \ldots + e_{i_g})p - (e_{i_1} + e_{i_2} + \ldots + e_{i_{c-1}})p < v(I_{i_g}) - v(I_{i_{c-1}}) + p.$$

Furthermore, we obtain

$$v(I_{i_g}) - v(I_{i_{c-1}}) = \sum_{j=1}^{n} x^*_{j,i_c} + \sum_{j=1}^{n} x^*_{j,i_{c+1}} + \ldots + \sum_{j=1}^{n} x^*_{j,i_g} \leq mp$$

by condition (2.2). Therefore,

$$v(I_{i_g}) - v(I_{i_{c-1}}) + p \leq mp + p = (m+1)p,$$

i.e.

$$e_{i_c} + e_{i_{c+1}} + \ldots + e_{i_g} < m + 1,$$

which gives

$$e_{i_c} + e_{i_{c+1}} + \ldots + e_{i_g} \leq m.$$

□

ACKNOWLEDGEMENTS

The results presented in this paper were attained during a visit of S.A. Kravchenko at the Otto-von-Guericke-Universität of Magdeburg which was supported by a fellowship of the Alexander von Humboldt Foundation.

References

[1] P. Brucker, S.A. Kravchenko, Scheduling jobs with equal processing times and time windows on identical parallel machines, OSM Reihe P, Heft 257, Universität Osnabrück, Fachbereich Mathematik/Informatik, 2004.

[2] P. Brucker, S.A. Kravchenko, Scheduling jobs with release times on parallel machines to minimize total tardiness, OSM Reihe P, Heft 258, Universität Osnabrück, Fachbereich Mathematik/Informatik, 2005.

[3] G.B. Dantzig, D.R. Fulkerson, Minimizing the number of tankers to meet a fixed schedule, Naval Res. Logist. Quart. 1 (1954) 217-222.

[4] B. Simons, Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines, SIAM J. Comput. 12 (1983) 294-299.