Journal of Scheduling **8:** 453–460, 2005.

MAXIMIZING WEIGHTED NUMBER OF JUST-IN-TIME JOBS ON UNRELATED PARALLEL MACHINES

SHAO CHIN SUNG[1] AND MILAN VLACH[2]

[1] *Department of Industrial and Systems Engineering, College of Science and Engineering, Aoyama Gakuin University, 5-10-1 Fuchinobe, Sagamihara City, Kanagawa, 229-8551, Japan*
[2] *Kyoto College of Graduate Studies for Informatics, 7 Monzen-cho, Tanaka, Sakyo-ku, Kyoto, 606-8225, Japan (On leave from School of Mathematics and Physics, Charles University, Prague, Czech Republic)*

ABSTRACT

We are concerned with problems of scheduling jobs non-preemptively with the objective to maximize the weighted number of jobs that are completed exactly at their due dates. It has been shown that the problems for single machine and identical parallel machines are polynomial time solvable. The purpose of this paper is to establish the complexity status of the problem for unrelated parallel machine, which was left open. First, we present a polynomial time algorithm for solving the problem when the number of machine is fixed. Second, we show that when the number of machine is a part of input, the problem becomes NP-hard in the strong sense.

KEY WORDS: just-in-time jobs, unrelated parallel machines, polynomial time algorithm, NP-hardness

1. INTRODUCTION

We are concerned with problems of scheduling jobs non-preemptively with the objective to maximize the weighted number of jobs that are completed exactly at their due dates. In other words, our objective is to minimize the weighted number of jobs which are early or tardy. The jobs that are completed exactly at their due dates in some schedule are called *just-in-time jobs* in that schedule. Just-in-time is a production-control method with benefits including better quality products, higher inventory turnover, higher productivity, etc.

It has been shown by Lann and Mosheiov (1996) the problem for single machine can be solved in polynomial time (also see Gavril (1972)). By Hiraishi, Levner and Vlach (2002), the problem for identical parallel machines has been shown to be polynomial time solvable, even if a non-negative set-up time between every two consecutive jobs on the same machine is required. Recently, a quadratic time algorithm for solving the problem for identical parallel machines with unit weights is presented by Čepek and Sung (2004). Observe that, in these two cases, the objective is equivalent to minimize the weighted number of late jobs, by assuming that each job J_i has release date $r_i = d_i - p_i$, where d_i and p_i are respectively the due date and the processing time of job J_i. Without any assumption on release dates, the problem of minimizing the weighted number of late jobs with release dates becomes NP-hard in the strong sense even in single machine case (see Garey and Johnson (1979)).

In this paper, we consider a more general case, namely, the case of *unrelated parallel machines*. That is, we allow jobs to have different processing times on different machines. This case

Correspondence to: Shao Chin Sung, Aoyama Gakuin University, 5-10-1 Fuchinobe, Sagamihara City, Kanagawa, 229-8551, Japan. Tel: +81-42-759-6307, Fax: +81-42-759-6559, E-mail: son@ise.aoyama.ac.jp

corresponds to the situation that there are more than one production facilities with different performances. Our purpose is to establish the complexity status of the problem for unrelated parallel machines, which was left open. We present a polynomial time algorithm for solving the problem when the number of machine is fixed. Furthermore, we show that when the number of machine is a part of input, the problem becomes NP-hard in the strong sense.

The rest of this paper is organized as follows. In Section 2, the problem is formulated. In Section 3, a polynomial time algorithm is presented for solving the problem when the number of machine is fixed. In Section 4, a proof of NP-hardness of the problem when the number of machine is a part of input is presented. Finally, our results are summarized and some open problems are given in Section 5.

2. PROBLEM FORMULATION

There are n jobs $J_1, J_2, \ldots, J_n$ to be scheduled nonpreemptively on m parallel machines $M_1, M_2, \ldots, M_m$. We denote by $p_{i,j}$ the processing time of job J_i on machine M_j, by d_i the due date of job J_i, and by w_i the weight associate with job J_i. Hence, an instance of the problem is a triplet (P, D, W), where $P = [p_{i,j}]$ is a non-negative $n \times m$ matrix, $D = (d_1, d_2 \ldots, d_n)$ is a non-negative n-vector, and $W = (w_1, w_2, \ldots, w_n)$ is a non-negative n-vector.

A schedule is an ordered pair (S, A) such that S is a subset of $\{1, 2, \ldots, n\}$ and A is a function defined on $\{1, 2 \ldots, n\}$ with values in $\{1, 2, \ldots, m\}$. We interpret a schedule (S, A) as follows. For each $i \in S$, job J_i is assigned to be processed on machine $M_{A(i)}$ and is completed exactly at its due date d_i (i.e., J_i is a just-in-time job). Then, the feasibility of schedules can be represented as follows. A schedule (S, A) is said to be *feasible* if, for every $i \in S$,

$$d_i - p_{i,A(i)} \geq 0, \tag{1}$$

and for every $i, k \in S$ with $i \neq k$ and $A(i) = A(k)$,

$$\left[d_i - p_{i,A(i)}, d_i\right) \cap \left[d_k - p_{k,A(k)}, d_k\right) = \emptyset. \tag{2}$$

Observe that (1) guarantees that, for each $i \in S$, job J_i has non-negative starting time, and (2) guarantees that each machine processes at most one job at a time.

The objective is to maximize the weighted number of just-in-time jobs on the set of feasible schedules, that is, to maximize the value of the function JIT defined as follows:

$$\text{JIT}(S, A) = \sum_{i \in S} w_i. \tag{3}$$

Observe that we disregard those jobs J_i with $i \notin S$, and each of such jobs is assumed to be scheduled arbitrarily after their due dates without overlapping with any other job.

Following the standard three-field notation, see e.g. Lawler et al. (1993), we denote the problem by $Rm\,||\,\text{JIT}$ when the number of machines is fixed and is equal to m; when the number of machines is not fixed, that is, it is a part of input, we denote the problem by $R||\text{JIT}$. In the following sections, we show that $Rm\,||\,\text{JIT}$ can be solved in polynomial time, and $R||\text{JIT}$ is NP-hard in the strong sense.

3. POLYNOMIAL TIME ALGORITHM FOR $Rm\,||\,\text{JIT}$

In this section, we show that $Rm||\text{JIT}$ can be solved by dynamic programming, and the running time of the proposed algorithm is a polynomial of n but is exponential of m. Notice that m is a

constant (which is not a part of input) for the problem $Rm||\text{JIT}$. Hence, the proposed algorithm is a polynomial time algorithm. Let $I = (P, D, W)$ be an instance of $Rm||\text{JIT}$ with n jobs. First, we assume that jobs are sorted non-decreasingly by d_is and are renumbered accordingly, i.e., after renumbering we have

$$d_1 \leq d_2 \leq \cdots \leq d_n. \tag{4}$$

Moreover, for simplicity, let $d_0 = 0$.

For each $v \in \{0, 1, \ldots, n\}^m$, a schedule (S, A) is called *v-bounded* if (S, A) is feasible and, for each $j \in \{1, 2, \ldots, m\}$,

$$v_j \geq \max\{i \in S \mid A(i) = j\},$$

and a schedule (S, A) is called *v-optimal* if (S, A) is v-bounded and maximizes JIT (S, A) among all v-bounded schedules. Notice that $Rm||\text{JIT}$ is equivalent to the problem to find an $(n \ldots, n)$-optimal schedule. For each $v \in \{0, 1, \ldots, n\}^m$, we denote by (S^v, A^u) a *v-optimal* schedule, by $\text{OPT}(v)$ the value $\text{JIT}(S^v, A^u)$. Moreover, let

$$\ell(v) = \max\{v_1, v_2, \ldots, v_m\},$$

and

$$L(v) = \left\{ j \in \{1, 2, \ldots, m\} \mid v_j = \ell(v), \quad d_{\ell(v)} - p_{\ell(v),j} \geq 0 \right\}.$$

Now let us consider relation between v-optimal schedules. Let $v \in \{0, 1, \ldots, n\}^m$. If $\ell(v) = 0$ (i.e., $v = (0, \ldots, 0)$), then it is obvious that $S^v = \emptyset$. Now suppose $\ell(v) > 0$. Let $v^0 \in \{0, 1 \ldots, n\}^m$ is such that for each $j \in \{1, 2, \ldots, m\}$,

$$v_j^0 = \min\{v_j, \ell(v) - 1\},$$

and, for each $k \in \{1, 2, \ldots, m\}$, let $v^k \in \{0, 1, \ldots, n\}^m$ is such that for each $j \in \{1, 2, \ldots, m\}$,

$$v_j^k = \begin{cases} \max\left\{ i \in \{0, 1, \ldots, \ell(v) - 1\} \mid d_i \leq d_{\ell(v)} - p_{\ell(v),j} \right\} & \text{if } j = k, \\ \min\{v_j, \ell(v) - 1\} & \text{otherwise.} \end{cases}$$

Then, if $\ell(v) \notin S^v$, we have

$$\text{OPT}(v) = \text{OPT}(v^0).$$

and (S^v, A^v) is a v^0-optimal schedule. Otherwise, if $\ell(v) \in S^v$ and $A^v(\ell(v)) = k$, we have $v_k = \ell(v)$ and $d_{\ell(v)} - p_{\ell(v),k} \geq 0$ (i.e., $k \in L(v)$), and moreover,

$$\text{OPT}(v) = \text{OPT}(v^k) + w_{\ell(v)},$$

and $(S^v \backslash \{\ell(v)\}, A^v)$ is a v^k-optimal schedule. Notice that $\ell(v^k) \leq \ell(v) - 1$ for all $k \in \{0\} \cup L(v)$. It turns out that a v-optimal schedule can be obtained if a v'-optimal schedule for all v' satisfying $\ell(v') \leq \ell(v) - 1$ is provided. More precisely, a v-optimal schedule can be found by the following algorithm:

OPTIMAL (v):

If $\ell(v) = 0$, then $S^v := \emptyset$; else

- set $u_0 := \text{OPT}(v^0)$ and set $u_k := \text{OPT}(v^k) + w_{\ell(v)}$ for each $k \in L(v)$;
- find $j \in \{0\} \cup L(v)$ satisfying $u_j = \max\{u_k \mid k \in \{0\} \cup L(v)\}$;

- OPT(v) := u_j
- if $j = 0$, then $(S^v, A^v) := (S^{v^0}, A^{v^0})$; else (S^v, A^v) is defined as follows:

– $S^v := S^{v^j} \cup \{\ell(v)\}$, and
– for each $i \in S^v$,

$$A^v(i) := \begin{cases} A^{v^j}(i) & \text{if } i \in S^{v^j}, \\ j & \text{otherwise (i.e., } i = \ell(v)); \end{cases}$$

Observe that the running time of OPTIMAL(v) is $O(mn)$.

Now an algorithm for finding an $(n, \ldots, n)$-optimal schedule can be described as follows:

Step 1. for $l = 0$ up to $n - 1$

- for each v satisfying $\ell(v) = l$, find an v-optimal schedule (S^v, A^v) by OPTIMAL(v);

Step 2. find $(S^{(n,\ldots,n)}, A^{(n,\ldots,n)})$ by OPTIMAL($n, \ldots, n$);
Step 3. return $(S^{(n,\ldots,n)}, A^{(n,\ldots,n)})$.

Theorem 1. *Rm* ||JIT *is polynomial time solvable.*

Proof. Observe that the running time for renumbering jobs for satisfying (4) is $O(n \log n)$. Then, in Step 1, there are n^m iterations of OPTIMAL(v), since there are n^m different vs satisfying $\ell(v) \leq n-1$. Thus, the running time of Step 1 is $O(mn^{m+1})$. The total running time of Steps 2 and 3 is $O(mn)$. Therefore, the total running time of the proposed algorithm is $O(n \log n + mn^{m+1} + mn) = O(mn^{m+1})$. Since m is a constant, the total running time of the algorithm is a polynomial of n. ■

4. NP-HARDNESS OF R||JIT

The NP-hardness of R||JIT is proven by reduction from an NP-hard problem called 3-SAT (see Garey and Johnson (1979)). The *3-SAT problem* can be described as follows: Given a collection C of clauses over a set X of boolean variables such that each clause $c \in C$ contains three literals, and decide whether there is a truth assignment $a : X \to \{0, 1\}$ satisfying C, i.e.,

$$\bigwedge_{c \in C} \left(\bigvee_{x \in c} a(x) \vee \bigvee_{\bar{x} \in c} \neg a(x) \right) = 1.$$

Let $C = \{c_1, c_2, \ldots, c_\alpha\}$ be an instance of 3-SAT over $X = \{x_1, x_2, \ldots, x_\beta\}$. Let γ_k for each $k \in \{1, 2, \ldots, \beta\}$ be the number of occurrences of variable x_k in C, and let $\gamma = \max\{\gamma_1, \gamma_2, \ldots, \gamma_\beta\} + 1$. From the instance C of 3-SAT, we construct an instance I_C of R||JIT in which there are $2\beta\gamma$ jobs to be scheduled on $\alpha + \beta$ machines. The instance $I_C = (P, D, W)$ is defined as follows:

- $D = (d_1, d_2, \ldots, d_{2\beta\gamma})$ and $W = (w_1, w_2, \ldots, w_{2\beta\gamma})$ are $(2\beta\gamma)$-dimensional non-negative vectors such that, for each $k \in \{1, 2, \ldots, \beta\}$ and $\ell \in \{1, 2, \ldots, \gamma\}$,

$$d_{(k-1)\gamma+\ell} = \begin{cases} \ell & \text{if } \ell < \gamma, \\ 2\gamma - 1 & \text{if } \ell = \gamma, \end{cases} \quad d_{(\beta+k-1)\gamma+\ell} = \begin{cases} \gamma + \ell & \text{if } \ell < \gamma, \\ \gamma & \text{if } \ell = \gamma, \end{cases}$$

and

$$w_{(k-1)\gamma+\ell} = w_{(\beta+k-1)\gamma+\ell} = \begin{cases} 1 & \text{if} \quad \ell < \gamma, \\ \gamma & \text{if} \quad \ell = \gamma. \end{cases}$$

- $P = [p_{i,j}]$ is a $2\beta\gamma \times (\alpha + \beta)$ non-negative matrix such that, for each $j \in \{1, 2, \ldots, \beta\}$, $k \in \{1, 2, \ldots, \beta\}$, and $\ell \in \{1, 2, \ldots, \gamma\}$,

$$p_{(k-1)\gamma+\ell,j} = p_{(\beta+k-1)\gamma+\ell,j} = \begin{cases} 1 & \text{if } \ell < \gamma \text{ and } j = k, \\ \gamma & \text{if } \ell = \gamma \text{ and } j = k, \\ 2\gamma & \text{otherwise,} \end{cases} \tag{5}$$

and for each $j \in \{\beta + 1, \beta + 2, \ldots, \beta + \alpha\}$, $k \in \{1, 2, \ldots, \beta\}$, and $\ell \in \{1, 2, \ldots, \gamma\}$,

$$\begin{aligned} p_{(k-1)\gamma+\ell,j} &= \begin{cases} d_{(k-1)\gamma+\ell} & \text{if } \ell < \gamma \text{ and } x_k \in c_{j-\beta}, \\ 2\gamma & \text{otherwise,} \end{cases} \\ p_{(\beta+k-1)\gamma+\ell,j} &= \begin{cases} d_{(\beta+k-1)\gamma+\ell} & \text{if } \ell < \gamma \text{ and } \bar{x}_k \in c_{j-\beta}, \\ 2\gamma & \text{otherwise.} \end{cases} \end{aligned} \tag{6}$$

In the following, we show that there exists a truth assignment satisfying C if and only if there exists a feasible schedule (S, A) of I_C satisfying

$$\text{JIT}(S, A) \geq \beta(2\gamma - 1) + \alpha. \tag{7}$$

Notice that each optimal schedule satisfies (7) if there exists a feasible schedule satisfying (7).

Lemma 1. If there exists a truth assignment satisfying C, then there exists a feasible schedule (S,A) of I_C satisfying (7).

Proof. Suppose there exists a truth assignment $a : X \to \{0, 1\}$ satisfying C. Then we can construct a feasible schedule (S, A) satisfying (7) as follows. Start with a feasible schedule (S, A) with $S = \emptyset$.

- For each $k \in \{1, 2, \ldots, \beta\}$, put $(\beta + k - 1)\gamma + \ell$ in S and set $A((\beta + k - 1)\gamma + \ell) = k$ for all $\ell \in \{1, 2, \ldots, \gamma\}$ if $a(x_k) = 1$ (see Figure 1(b)), otherwise put $(k - 1)\gamma + \ell$ in S and set $A((k - 1)\gamma + \ell) = k$ for all $\ell \in \{1, 2, \ldots, \gamma\}$ (see Figure 1 (a)). As a consequence, for each $k \in \{1, 2, \ldots, \beta\}$,

$$\sum_{i \in S: A(i) = k} w_i = 2\gamma - 1.$$

  Notice that (S,A) is still a feasible schedule (see Figure 1).
- For each $j \in \{1, 2, \ldots, \alpha\}$, find either a literal $x_k \in c_j$ with $a(x_k) = 1$ or a literal $\bar{x}_k \in c_j$ with $a(x_k) = 0$. Notice that such a literal always exists because a is a truth assignment satisfying C.

  Suppose we find $x_k \in c_j$ with $a(x_k) = 1$. Then find an $\ell \in \{1, 2, \ldots, \gamma - 1\}$ such that $(k - 1)\gamma + \ell \notin S$, and then put $(k-1)\gamma + \ell$ in S and set $A((k-1)\gamma + \ell) = j + \beta$. Notice that such an ℓ always exists, since the number of occurrences of x_k is at most $\gamma - 1$, i.e., x_k is included in at most $\gamma - 1$ clauses c_js.

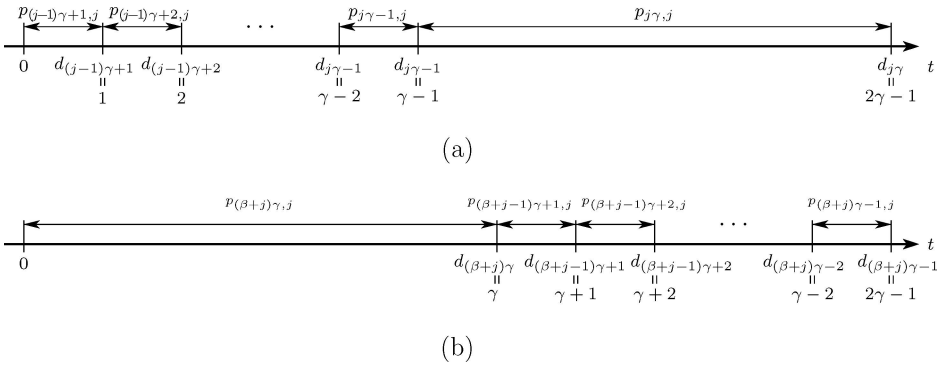

Figure 1. Processing times $p_{i,j}$ and due dates d_i of jobs J_i with $p_{i,j} \le d_i$ on Machine M_j for $j \in \{1, 2, \ldots, \beta\}$.

Suppose we find $\bar{x}_k \in c_j$ with $a(x_k) = 0$. Then find an $\ell \in \{1, 2, \ldots, \gamma - 1\}$ such that $(\beta + k - 1)\gamma + \ell \notin S$, and then put $(\beta + k - 1)\gamma + \ell$ in S and set $A((\beta + k - 1)\gamma + \ell) = j + \beta$.

Again notice that such an ℓ always exists.

As a consequence, we have for each $j \in \{1, 2, \ldots, \alpha\}$,

$$\sum_{i \in S: A(i) = j} w_i = 1.$$

It can easily be verified that (S, A) is a feasible schedule and

$$\text{JIT}(S, A) = \sum_{i \in S: A(i) \le \beta} w_i + \sum_{i \in S: A(i) > \beta} w_i = \beta(2\gamma - 1) + \alpha.$$

■

Before we proceed to show that there exists a truth assignment satisfying C if there exists a feasible schedule (S, A) satisfying (7), let us show some properties of feasible schedules of I_C.

Let (S,A) be a feasible schedule of I_C. From (1), $p_{i,j} \ge 2\gamma$, and $d_i \le 2\gamma - 1$ for each $i \in \{1, 2, \ldots, 2\beta\gamma\}$, we have $A(i) \ne j$ when $i \in S$. Then, for each $k \in \{1, 2, \ldots, \beta\}$,

- $i \in S$ and $A(i) = k$ implies $(k - 1)\gamma + 1 \le i \le k\gamma$ or $(\beta + k - 1)\gamma + 1 \le i \le (\beta + k)\gamma$,
- $k\gamma \in S$ implies $A(i) = k$, and $(\beta + k)\gamma \in S$ implies $A(i) = k$.

From (2) (also see Figure 1), we have

- $k\gamma \in S$ implies $(k - 1)\gamma - 1 \le i \le k\gamma$ for each $i \in S$ with $A(i) = k$, and
- $(\beta + k)\gamma \in S$ implies $(\beta + k - 1)\gamma - 1 \le i \le (\beta + k)\gamma$ for each $i \in S$ with $A(i) = k$.

Thus, for each $k \in \{1, 2, \ldots, \beta\}$,

$$\sum_{i \in S: A(i) = k} w_i \le 2\gamma - 1,$$

and we have $\sum_{i \in S: A(i) = k} w_i = 2\gamma - 1$ only if either

$$\{i \in S \mid A(i) = k\} = \{(k - 1)\gamma - 1, (k - 1)\gamma - 2, \ldots, k\gamma\} \tag{8}$$

or

$$\{i \in S \mid A(i) = k\} = \{(\beta + k - 1)\gamma - 1, (\beta + k - 1)\gamma - 2, \ldots, (\beta + k)\gamma\} \tag{9}$$

is satisfied. Moreover, for each $j \in \{\beta + 1, \ldots, \alpha + \beta\}$, we have $p_{ij} \in \{d_i, 2\gamma\}$, and thus,

$$|\{i \in S \mid A(i) = j\}| \leq 1.$$

Recall that, for each $k \in \{1, 2, \ldots, \beta\}$, $k\gamma \in S$ implies $A(k\gamma) = k$, and $(\beta + k)\gamma \in S$ implies $A(i) = k$. Thus, for each $j \in \{\beta + 1, \ldots, \alpha + \beta\}$,

$$\sum_{i \in S: A(i) = j} w_i \leq 1.$$

Lemma 2. If there exists a feasible schedule (S, A) *of* I_C *satisfying* (7), *then there exists a truth assignment satisfying* C.

Proof. Now suppose there exists a feasible schedule (S,A) satisfying JIT$(S, A) = \beta(2\gamma - 1) + \alpha$. Then, either (8) or (9) is satisfied for each $k \in \{1, 2, \ldots, \beta\}$, and $|\{i \in S \mid A(i) = j\}| = 1$ for each $j \in \{\beta + 1, \beta + 2, \ldots, \alpha + \beta\}$. Let $a : X \to \{0, 1\}$ be a truth assignment such that, for each $k \in \{1, 2, \ldots, \beta\}$, $a(x_k) = 1$ if (9) is satisfied, otherwise $a(x_k) = 0$. Then, it can easily be verified that, for each $j \in \{\beta + 1, \beta + 2, \ldots, \alpha + \beta\}$, the unique $i \in S$ with $A(i) = j$ is such that $x_k \in c_{j-\beta}$ for some x_k with $a(x_k) = 1$ if $i \leq \beta\gamma$, $\overline{x}_k \in c_{j-\beta}$ for some x_k with $a(x_k) = 0$ otherwise. Therefore, a is a truth assignment satisfying C. ■

From Lemmas 1 and 2, for each instance C of 3-SAT, one can decide whether there exists a truth assignment satisfying C by finding an optimal schedule (S,A) of I_C, and then testing whether (S,A) satisfies (7). Hence, the reduction is completed. Moreover, observe that it is a polynomial time reduction, and the size of instance I_C in unary encoding is a polynomial of instance C. Therefore, the following theorem is obtained.

Theorem 2. $R||$ JIT *is NP-hard in the strong sense.*

5. CONCLUSION

In this paper, we have establish the complexity status of the problems $Rm||$ JIT and $R||$ JIT. Namely, we have shown that $Rm||$ JIT is polynomial time solvable by proposing an algorithm with running time $O(mn^{m+1})$, and $R||$ JIT is NP-hard in the strong sense by a reduction from 3-SAT.

The complexity status of related problems are summarized in Table 1. As a restricted variant of $Rm||$ JIT, the problem $Qm||$ JIT, i.e., the case of uniform parallel machines with the number of

Table 1. Complexity status of related problems

Parallel machines	m is fixed	m is a part of input
Identical	Polynomial time	Polynomial time
Uniform	Polynomial time	Open
Unrelated	Polynomial time	Strongly NP-hard

machines fixed, can also be solved in polynomial time by the proposed algorithm. However, the complexity status is still open for the problem $Q||$ JIT, i.e., the case of uniform parallel machines with the number of machines as a part of input. Moreover, the complexity status of the unit-weighted case of $R||$ JIT is also open.

REFERENCES

Čepek, O. and S. C. Sung, "A quadratic time algorithm to maximize the number of just-in-time jobs on identical parallel machines," accepted by *Computers & Operations Research*, 2004.

Garey, M. R. and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.

Gavril, F., "Algorithms for minimizing coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph," *SIAM Journal on Computing*, **1**(2), 180–187 (1972).

Hiraishi, K., E. Levner, and M. Vlach, "Scheduling of parallel identical machines to maximize the weighted number of just-in-time jobs," *Computers & Operations Research*, **29**(7), 841–848 (2002).

Lann, A. and G. Mosheiov, "Single machine scheduling to minimize the number of early and tardy jobs," *Computers & Operations Research* **23**, 765–781 (1996).

Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, "Sequencing and scheduling, Algorithms and complexity," in S. C. Graves, A. H. G. Rinnooy Kan and P. H. Zipkin (eds.), *Handbooks in Operations Research and Management Science*, Vol. 4, Amsterdam, 1993, pp. 455–522.