LETTER

Maximizing the Total Weight of Just-In-Time Jobs under Multi-Slot Conditions Is NP-Hard*

Eishi CHIBA[†a)], ***Member and*** **Shinji IMAHORI**[††], ***Nonmember***

SUMMARY A job is called just-in-time if it is completed exactly on its due date. Under multi-slot conditions, each job has one due date per time slot and has to be completed just-in-time on one of its due dates. Moreover, each job has a certain weight per time slot. We would like to find a just-in-time schedule that maximizes the total weight under multi-slot conditions. In this paper, we prove that this problem is NP-hard.

key words: *scheduling, just-in-time, time slot, weight, NP-hard*

1. Introduction

One key feature of just-in-time scheduling is the fact that it plays a crucial role in enabling control systems to observe due dates. Just-in-time scheduling has applications to both manufacturing and computer systems [2]. As a result, such scheduling problems have been receiving growing attention of late.

In the past, the objective function of just-in-time scheduling problems under multi-slot conditions has largely concerned minimizing the number of time slots [3]–[5]. Following traditional scheduling methodology, in this paper we introduce a new class of just-in-time scheduling problems under multi-slot conditions which is related to the weight of jobs. Research in this area of just-in-time scheduling may also prove to be of great value in further improving the efficiency of manufacturing and computer systems.

In the field of just-in-time scheduling, a due date is usually associated with each job to be processed, and each job should be completed by its due date. However, if a job is completed before its due date, the earlier this happens, the greater the storage cost incurred. Here, we focus on scheduling problems that relate to each job finishing exactly on its due date. A job is called *just-in-time* if it is completed exactly on its due date.

A number of known results on just-in-time scheduling exist. In [6], the activity selection problem was presented and it was shown that the problem of maximizing the number of just-in-time jobs for a single machine is solvable in linear time by a simple greedy method. The above problem was extended in [7] to the case of identical parallel machines. It was proven, in [7] that the problem is also solvable in quadratic time by a simple greedy method. Moreover, it was shown in [8], that the problem presented in [7] is solvable in polynomial time even if a non-negative weight is assigned to each job, a non-negative set-up time is assigned to each ordered pair of jobs, and the objective is to maximize the total weight value of just-in-time jobs.

The above book [6] and papers [7], [8] did not consider how to handle jobs that do not finish exactly on their due dates because such jobs do not affect the values of objective functions in these problems. However, considering real-life situations, jobs which are not processed in a certain period of time will be scheduled for the next possible processing opportunity. Such jobs should remain just-in-time for successive opportunities. For example, we can assume that shipping time is fixed on a daily/weekly/monthly, etc. basis. This situation was formulated using multiple time slots as observed in earlier works [3]–[5].

Previous work exists that addresses time slots, where the objective is to minimize the maximum number of periodic time slots required for each machine when scheduling all given jobs. It was shown in [4], that the problem becomes NP-hard (in the strong sense) even for a single machine case, and is solvable in polynomial time for an arbitrary number of identical parallel machines, if set-up times are not considered. Moreover, [5] showed that this problem cannot be approximated, assuming P $\neq$ NP, and also presented a heuristic algorithm for a single machine case. In this heuristic algorithm, the approximation ratio depends on the upper bound of set-up times.

The intractability results for the above-mentioned problem in [4] and [5], i.e. NP-hardness and inapproximability, come from the consideration of the set-up time. In order to counter this, we do not consider the set-up time in this paper. Instead, the weight for each job per time slot is considered. Note that weight is not considered in the above-mentioned problem in [4] and [5]. In this paper, we present a scheduling problem maximizing the total weight of just-in-time jobs under multi-slot conditions. Then, we prove that this problem is NP-hard. The key idea is a reduction from the μ-coloring problem over interval graphs, which is known to be NP-hard [9]. Moreover, it is shown that this problem remains NP-hard even when the number of machines is one, and even when the weight function is non-increasing with increasing time.

Manuscript received August 4, 2015.
Manuscript revised September 29, 2015.
Manuscript publicized October 26, 2015.

[†]The author is with the Department of Industrial and Systems Engineering, Faculty of Science and Engineering, Hosei University, Koganei-shi, 184–8584 Japan.
[††]The author is with the Department of Information and System Engineering, Faculty of Science and Engineering, Chuo University, Tokyo, 112–8551 Japan.

*A preliminary version of this paper was presented in [1].

a) E-mail: e-chiba@hosei.ac.jp

DOI: 10.1587/transinf.2015EDL8171

2. Problem Description

There are m parallel identical machines $M_1, M_2, \ldots, M_m$ and n jobs $J_1, J_2, \ldots, J_n$ with positive processing times $p_1, p_2, \ldots, p_n$. The operation time on all machines is divided into time slots of length L, and each job is assigned a set of periodically repeating due dates, one per time slot. Namely, due dates of a job J_j are $d_j, L + d_j, 2L + d_j, \ldots$, where d_j ($\geq p_j$) is the due date of the job J_j in the first time slot, $L + d_j$ is the due date of the job J_j in the second time slot, and so on, and where L is the length of each time slot. Every job must be completed exactly on one of its due dates. Note that $p_j \leq d_j \leq L$ holds since one due date of a job exists per time slot. Moreover, a non-negative weight function $w_j(l)$ is associated with each job J_j upon its completion in the l-th time slot, where the number associated with l refers to the *slot number*, i.e. $l = 1$ refers to time slot 1, $l = 2$ refers to time slot 2, etc.

We assume that all jobs are completed by the $\lceil n/m \rceil$-th time slot. Thus, the slot number $l \leq \lceil n/m \rceil$ holds. Note that it is possible to complete all jobs if $\lceil n/m \rceil$ time slots are considered.

A *schedule* is a mapping $S : J_j \mapsto (M^S_{[j]}, C^S_j)$, where $M^S_{[j]}$ is the machine on which job J_j is processed and C^S_j is the time instant when job J_j is finished on machine $M^S_{[j]}$. A schedule S is deemed *feasible* if

(i) for each job J_j, there exists a non-negative integer r^S_j such that $C^S_j = r^S_j \cdot L + d_j$, and
(ii) for every pair of jobs J_j and J_k, if $M^S_{[j]} = M^S_{[k]}$, then $C^S_k - p_k \geq C^S_j$ or $C^S_j - p_j \geq C^S_k$.

We refer to the constraints (i) as *just-in-time constraints* and (ii) as *compatible constraints*.

The problem is to find a feasible non-preemptive schedule that maximizes the total weight of just-in-time jobs when processing all jobs. Namely, a *Just-In-Time Scheduling Problem (JITS)* is stated as follows. Given number of jobs n, number of machines m, processing times p_j, due dates d_j, weights $w_j(l)$, and length of time slot L, maximize the value of the sum

$$R(S) = \sum_{j=1}^{n} w_j \left(\left\lfloor \frac{C^S_j}{L} \right\rfloor \right)$$

called the *objective function* over the set of all feasible non-preemptive schedules.

An *optimal schedule* for an input of the problem is a feasible schedule that achieves the largest objective function value, known as the *optimal value*. If the number of machines is no less than the number of jobs, then there is a feasible schedule in which no machine executes more than one job. Such a problem is trivial and so, from now on, we assume that $m < n$.

Example 1: We consider the following input of problem JITS, i.e. $m = 2$, $n = 4$, $L = 6$, processing times p_j, due dates d_j, and weights $w_j(l)$ as in Table 1.

Table 1 The left table shows the processing time p_j and due date d_j for each job J_j. The right table shows the weight $w_j(l)$ for each job J_j and each slot number l.

j	p_j	d_j
1	2	5
2	2	3
3	1	4
4	3	5

$j \backslash l$	1	2
1	4	3
2	6	5
3	3	1
4	5	2

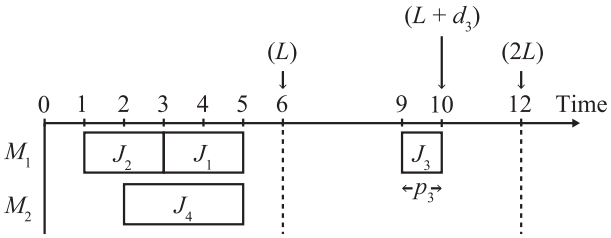

Fig. 1 A feasible schedule by the greedy method for Example 1.

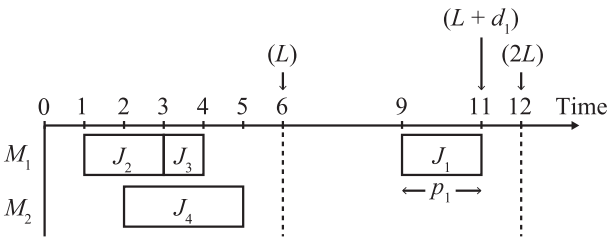

Fig. 2 An optimal schedule for Example 1.

We consider a greedy method, in which the total weight value of just-in-time jobs is maximized in each time slot from the first time slot onwards. Note that, when focusing on only one time slot, a polynomial-time algorithm which maximizes the total weight value of just-in-time jobs exists (see [8]). Figure 1 shows a feasible schedule obtained from the greedy method. The value of $R(S)$ is 16. On the other hand, Fig. 2 shows an optimal schedule that achieves the largest objective function value. The optimal value is 17.

3. NP-Hardness

In this section, we show that problem JITS is NP-hard. We use, for the reduction, the μ-coloring problem over interval graphs, which was proven to be NP-complete in [9].

An undirected graph G is called an *interval graph* if its vertices can be put in a one-to-one correspondence with a family of intervals on a real line such that two vertices are adjacent in G if and only if the corresponding intervals intersect [10]. In other words, an undirected graph $G = (V, E)$ is an interval graph if there exists a set $\{I_v | v \in V\}$ of real intervals such that $I_u \cap I_v \neq \emptyset$ if and only if $\{u, v\} \in E$.

Given an interval graph, open intervals $I_v = (\alpha_v, \beta_v)$ corresponding to every vertex v of the interval graph can be computed in linear time by the algorithm presented in [11]. Therefore, given an interval graph, we can efficiently obtain the corresponding intervals on a real line.

A *coloring* of an undirected graph $G = (V, E)$ is a function $f : V \to \mathbb{N}$ such that $f(u) \neq f(v)$ whenever $\{u, v\} \in E$. Given an undirected graph $G = (V, E)$ and a function $\mu : V \to \mathbb{N}$, G is *μ-colorable* if a coloring f of G exists such that $f(v) \leq \mu(v)$ for every $v \in V$ [12].

μ-coloring problem over interval graphs
Input: An interval graph $G = (V, E)$, a function $\mu : V \to \mathbb{N}$.
Question: Is G μ-colorable?

This problem is NP-complete [9]. In the following, we prove the NP-hardness of problem JITS.

Given an input of the μ-coloring problem over interval graphs, the corresponding intervals on a real line can be translated effectively. Then, we construct an input of the decision version of problem JITS, so that this input has a solution if and only if the input of the μ-coloring problem over interval graphs has a solution.

DECISION JITS
Input: $m = 1$, $n = |V|$ (job J_v is identified with interval $I_v = (\alpha_v, \beta_v)$ corresponding to each vertex $v \in E$), $p_v = \beta_v - \alpha_v$ for all $v \in V$, $d_v = \beta_v$ for all $v \in V$, $L = \max_{v \in V}\{\beta_v\}$,

$$w_v(l) = \begin{cases} 1 & \text{if } l \leq \mu(v), \\ 0 & \text{otherwise,} \end{cases}$$

for all $v \in V$.

Question: Is there a feasible schedule S with an objective function that is at least n, i.e. such that $R(S) \geq n$?

Note that, from the way of constructing an input of DECISION JITS, if two vertices in the interval graph are non-adjacent, the two jobs corresponding to the two vertices are compatible.

Lemma 1: An input of μ-coloring problem over interval graphs has a solution if and only if the constructed input of DECISION JITS has a solution.

Proof: If the μ-coloring problem over interval graphs has a solution, then we can easily construct a solution of DECISION JITS by scheduling each job J_v in the $f(v)$-th time slot using the coloring function. For each $v \in V$, $f(v) \leq \mu(v)$ holds, and thus the value of the objective function for the constructed schedule is exactly n.

On the other hand, if DECISION JITS has a solution (in such a case, the value of the objective function is exactly n because the weight of each job is, at most, one), then the designated slot number for each job gives a coloring such that the input graph is μ-colorable, since any two jobs, which cannot be scheduled in the same time slot, must be, in such a case, scheduled in different time slots. Thus the slot numbers associated with each job give a coloring in the input graph. □

Since the above-stated reduction from the μ-coloring problem over interval graphs to DECISION JITS is clearly polynomial, we can obtain the following theorem.

Theorem 1: Problem JITS is NP-hard.

Remark 1: Since the number of machines is one in the reduction, problem JITS remains NP-hard even if $m = 1$.

Remark 2: In the reduction, since the weight function has a 0/1 value and is non-increasing with increasing slot number for each job, problem JITS remains NP-hard even for such a weight function case.

If a job is processed in the next or later time slots, it may be possible to consider that penalties for values of objective function increase proportionally with the length of delay. The assumption that a weight (in other words, profit) for each job is non-increasing with time was considered in [13]. Although it was open as to whether the restricted problem is NP-hard or not in [13], from Remark 2, we can see that the restricted problem is indeed NP-hard.

4. Conclusions

We presented the scheduling problem maximizing the total weight of just-in-time jobs under multi-slot conditions. We proved that this problem is NP-hard. Moreover, even if the weight function only takes a value of 0/1 and is non-increasing, the problem is still NP-hard.

A study of solvable cases, in which problem JITS can be solved in polynomial time, is one possible area of future investigation. Developing good heuristics is another area worthy of further research. In many practical situations, both the maximization of the total weight and the minimization of the number of necessary time slots might also be considered; consequently, a study of schedules balancing the total weight against the number of necessary time slots is another potential area of future work.

References

[1] E. Chiba and S. Imahori, "Maximizing the total weight of just-in-time jobs under multi-slot conditions is NP-hard," Proc. International Symposium on Scheduling, pp.65–67, 2013.
[2] J. Józefowska, Just-in-Time Scheduling: Models and Algorithms for Computer and Manufacturing Systems, Springer, 2007.
[3] K. Hiraishi, "Just-in-time scheduling of parallel identical machines with multiple time slots," Proc. 4th Czech-Japan Seminar on Data Analysis and Decision Making under Uncertainty, pp.33–39, 2001.
[4] O. Čepek and S.C. Sung, "Just-in-time scheduling with periodic time slots," Scientiae Mathematicae Japonicae, vol.60, no.2, pp.295–301, 2004.
[5] E. Chiba and K. Hiraishi, "A heuristic algorithm for one-machine just-in-time scheduling problem with periodic time slots," IEICE Trans. Fundamentals, vol.E88-A, no.5, pp.1192–1199, 2005.
[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms, Second ed., MIT Press, 2001.
[7] O. Čepek and S.C. Sung, "A quadratic time algorithm to maximize the number of just-in-time jobs on identical parallel machines," Comput. Oper. Res., vol.32, no.12, pp.3265–3271, 2005.
[8] K. Hiraishi, E. Levner, and M. Vlach, "Scheduling of parallel identical machines to maximize the weighted number of just-in-time jobs," Comput. Oper. Res., vol.29, no.7, pp.841–848, 2002.
[9] F. Bonomo, G. Durán, and J. Marenco, "Exploring the complexity boundary between coloring and list-coloring," Annals of Operations Research, vol.169, no.1, pp.3–16, 2009.
[10] S.R. Arikati and C.P. Rangan, "Linear algorithm for optimal path

cover problem on interval graphs," Inf. Process. Lett., vol.35, no.3, pp.149–153, 1990.

[11] D.G. Corneil, S. Olariu, and L. Stewart, "The ultimate interval graph recognition algorithm?," Proc. Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pp.175–180, 1998.

[12] F. Bonomo and M. Cecowski, "Between coloring and list-coloring: μ-coloring," Electronic Notes in Discrete Mathematics, vol.19, pp.117–123, 2005.

[13] E. Chiba, T. Kageyama, Y. Karuno, and H. Goto, "Maximizing the total weight value of just-in-time jobs in identical parallel machines with periodic time slots," Proc. IEEE International Conference on Industrial Engineering and Engineering Management, pp.1349–1353, 2012.