다수의 경쟁이 존재하는 환경에서 적시 스케줄링에 관한 연구*

정대영** · 최병천***†

Just-in-time Scheduling with Multiple Competing Agents

Dae-Young Chung** · Byung-Cheon Choi***

■ Abstract ■

We consider a multi-agent scheduling problem such that each agent tries to maximize the weighted number of just-in-time jobs. Two objectives are considered : the first is to find the optimal solution for one agent with constraints on the other agents' weight functions, and the second is to find the largest set of efficient schedules of which corresponding objective vectors are different for the case with identical weights. We show that when the number of agents is fixed, the single machine case with the first objective is NP-hard in the ordinary sense, and present the polynomial-time algorithm for the two-machine flow shop case with the second objective and identical weights.

Keyword : Multi-Agent Scheduling, Just-in-Time job, Efficient Schedule

1. Introduction

We consider a multi-agent scheduling problem, where each agent is responsible for his own set of non-preemptive jobs. All jobs share the common resources (machines) and each agent

논문접수일 : 2011년 08월 16일 논문게재확정일 : 2011년 11월 17일
논문수정일(1차 : 2011년 10월 13일, 2차 : 2011년 11월 16일)
* This study was financially supported by the research fund of Chungnam National University in 2011.
** JDA 소프트웨어
*** 충남대학교 경영학부
† 교신저자

wishes to optimize his own objective applied to his own set of jobs. This situation can be found in various real-world applications. In a preventive maintenance planning problem, maintenance periods and processing periods can be viewed as the jobs in different agents [14]. Thus, each agent has one's objective or constraint depending on the schedule of the jobs. Hall and Potts [9] consider a rescheduling problem, in which a set of jobs is scheduled before and newly coming jobs are to be scheduled. All jobs have to share machines for their processing. Assume that while the jobs in the first set try to minimize the deviation from the original schedule, the jobs in the second set may have a different cost function. This situation can be regarded as a two-agent scheduling problem. In multimedia services environment, limited radio resources have to be assigned to various types of data requests. Each request type is regarded as a agent and it can have a different criteria from others [4].

In this paper, the objective of each agent is to maximize the number of just-in-time (JIT) jobs which should be completed exactly on their due dates. This type of the objective describes the business environment where through JIT production, companies try to attain benefits including better quality products, higher inventory turnover, higher productivity, etc.

The remainder of this paper is organized as follows. Section 2 provides a literature review related to our problem. Section 3 defines two problems formally. Section 4 shows that the first model is NP-hard in the ordinary sense when the number of agents is fixed. Section 5 shows that the second model with the fixed number of agents is polynomially solvable. Finally, Section 6 draws our conclusions.

2. Literature Review

Scheduling with multiple agents has already received considerable attention in the literature. Agnetis et al. [1, 2] considered a single machine scheduling problems with two agents in which each agent wants to minimize a certain objective function, such as maximum of regular functions, number of tardy jobs, and total weighted completion times. They established the computational complexity of all combinations of objectives except the case that the objective is to minimize the total completion time of the first agent while the number of tardy jobs of the other agent is less than or equal to the given value. Ng et al. [15] and Leung et al. [14] considered that case. Ng et al. [15] showed that the problem is NP-hard under high multiplicity encoding and can be solved in pseudo-polynomial time under binary coding. Later, Leung et al. [14] showed that the problem is NP-hard without high multiplicity encoding. Cheng et al. [6] considered the problems with more than two agents and each agent having as objective the total weighted number of tardy jobs. They showed that the problem is strongly NP-hard. Lee et al. [12] considered a multi-agent scheduling problem on a single machine in which each agent tries to minimize the total weighted completion time. They showed that the problem has fully polynomial time approximation scheme and provided an efficient approximation algorithm with a reasonable good worst-case ratio. The results of computational complexity for single machine problems with two agents are presented in <Table 1>.

〈Table 1〉 Summary of Complexity Results

(Agent 1, Agent 2)	Computational Complexity	References
(f_{max}, f_{max})	$O(n^2)$	[1, 2]
$(\sum w_jC_j,\ f_{max})$	NP-hard in the ordinary sense	[1, 2]
$(\sum C_j,\ f_{max})$	$O(n \log n)$	[1, 2]
$(\sum U_j,\ f_{max})$	$O(n \log n)$	[1, 2]
$(\sum U_j, \sum U_j)$	$O(n^3)$	[1, 2]
$(\sum C_j, \sum U_j)$	NP-hard in the ordinary sense	[14]
$(\sum w_jC_j, \sum U_j)$	NP-hard in the ordinary sense	[1, 2]
$(\sum C_j, \sum C_j)$	NP-hard in the ordinary sense	[1, 2]
$(\sum C_j, f_{max})$	NP-hard in the ordinary sense	[1, 2]
$(\sum w_jU_j,\ \sum w_jU_j)$	NP-hard in the strong sense	[6]

Note that f_{max}, $\sum U_j$ and $\sum w_jC_j$ mean maximum of regular functions, number of tardy jobs, and total weighted completion times, respectively.

Recently, Leung et al. [14] considered two-agent scheduling problem on multiple parallel machines. To the best of our knowledge, however, multi-agent scheduling problems have not been discussed in the flow shop environment except Agnetis et al. [1] and Lee et al. [13] They considered a two-machine flow shop problem with two agents. In Agnetis et al. [1], the objective is to minimize the makespan of the first agent while that of the other agent is less than or equal to the given value. They showed that this case is NP-hard. In Lee et al. [13] the objective is to minimize the total completion time of the first agent with no tardy jobs for the other agent. They developed a branch-and-bound algorithm and simulated annealing heuristic.

Since Cormen and Leiserson [8] and Lann and Mosheiov [11] which analyzed the problem of minimizing the weighted number of early and tardy jobs on a single machine, many researchers have studied JIT scheduling problems under various machine environments. Hiraishi et al. [10] considered an identical machine scheduling problem in which the objective is to maximize the weighted number of JIT jobs. They showed that the problem with sequence dependent setup times is polynomially solvable. Cepek and Sung [5] proposed an improved efficient algorithm for the same problem. Sung and Vlach [16] considered the problem of maximizing the weighted number of JIT jobs on unrelated parallel machines. They showed that the problem is polynomially solvable when the number of machines is fixed, and it is strongly NP-hard when the number of machines is a part of input. Choi and Yoon [7] considered the maximization of the weighted number of JIT jobs in a flow shop problem. They showed that the two-machine problem is NP-hard, while the problem with identical weights is polynomially solvable. Furthermore, they showed that the problem with identical weights is strongly NP-hard if the number of machines is three or more.

3. Problem Definition and Notations

In this section, we first define two problems under consideration, and then introduce the nota-

tions and the formal definition of the problems.

3.1 Problem Definition

Two different optimization models considered by this paper are as follows :

Problem P$_0$1 : Constrained Optimization Model
Problem P$_0$2 : Pareto Optimization Model

Problem P$_0$1 can be described as the model in which one of the objectives has to be maximized while each value of the remaining objectives has to be kept larger than or equal to a fixed value. In this paper, we prove that Problem P1 on a single machine is NP-hard in the ordinary sense when the number of agents is fixed. Let Problem P$_0$1 on a single machine be referred to as *Problem P1*.

Problem P$_0$2 can be described as a model to generate an efficient set, which is defined as follows.

Definition : Let $x=(x_1, \cdots, x_g)$ and $y=(y_1, \cdots, y_g)$ be two vectors of real numbers. Vector y *dominates* vector x if $x_k \le y_k$, $k=1,\cdots, g$. Let Y and Y^* be two sets of vectors of real numbers. If each vector in Y is dominated by some vector in Y^*, and no vector in Y dominates any other vector in Y^*, then Y^* is an *efficient* set with respect to Y.

The goal of this model is to find a set of efficient schedules. However, consider the instance of Problem P$_0$2 with two agents, in which the processing time and due date of job j of agent h are $p_j^h=1$ and $d_j^h=j$, respectively, $j=1, 2, \cdots, n_h$, $h=1, 2$ and $n_1=n_2=n$. It is easy to see that there exist 2^n efficient schedules in this instance. Thus, it is observed that the number of efficient schedules of Problem P$_0$2 can be an exponential function of the size of the input, even for two-agent case with identical weights.

Thus, we focus on the case with identical weights in which the objective is to find the largest set of efficient schedules of which objective vectors are different from each other. In this paper, we show that this case in two-machine flow shop is polynomially solvable when the number of agents is fixed. Let Problem P$_0$2 with identical weights in two-machine flow shop be referred to as *Problem P2*.

3.2 Notations

Our problem can be stated formally as follows. Agent h has his own set of n_h jobs $J^h=\{J_1^h, J_2^h, \cdots, J_{n_h}^h\}$, $h=1, 2, \cdots, g$. Associated with job J_j^h is a due date d_j^h a weight w_j^h and a processing time of machine i $p_{i,j}^h$. Note that in the single machine case, the processing time of job J_j^h is denoted by p_j^h. The objective of agent h is to maximize the number of job J_j^h which is exactly completed on d_j^h. Let $\sigma=(E^1(\sigma), \cdots, E^g(\sigma))$ be a feasible schedule, where $E^h(\sigma)$ is the set of JIT jobs of agent h in schedule σ, $h=1, 2, \cdots, g$, and $\sigma(j)$ be the j-th JIT job in schedule σ. Let $C_{i,j}^h(\sigma)$ be the completion time of job J_j^h on machine i in schedule σ. Note that in the single machine case, the completion time of job J_j^h in schedule σ is denoted by $C_j^h(\sigma)$. Let $z(\sigma)=(|E^1(\sigma)|, \cdots, |E^g(\sigma)|)$ be the objective vector corresponding to schedule σ, where $|E^h(\sigma)|$ is the cardinality of $E^h(\sigma)$, $h=1, 2, \cdots, g$. Let U_j^h be an indicator variable that is defined below : For $h=1, 2, \cdots, g$,

$$U_j^h = \begin{cases} 1 & \text{if job } j \text{ of agent } h \text{ is completed on } d_j^h \\ 0 & \text{otherwise.} \end{cases}$$

The objective of Problem P1 is to maximize the weighted number of JIT jobs of agent 1 while each values of the objectives of agents in $\{2, 3, \cdots, g\}$ are larger than or equal to a fixed value. That is,

$$\max \sum_{j=1}^{n_1} w_j^1 U_j^1 \; s.t. \; \sum_{j=1}^{n_h} w_j^h U_j^h \geq \omega_h \text{ for } h = 2, 3, \cdots, g.$$

The objective of Problem P2 is to construct the largest set of efficient schedules of which objective vectors are different, which is denoted by E^*.

Without loss of generality, let $d_1^h \leq d_2^h \leq \cdots \leq d_{n_h}^h$; $h = 1, 2, \cdots, g$. It is observed from the definition of a JIT job that there exist an optimal schedule (Problem P1) and efficient schedules (Problem P2) such that

- JIT jobs are processed in increasing order of their due dates;
- Tardy jobs follow all the JIT jobs.

Throughout the paper, we only consider these schedules.

4. Problem P1

In this section, we show that the problem is NP-hard in the ordinary sense when the number of agents is fixed.

In order to prove the NP-hardness of Problem P1, we introduce the partition problem, which is stated as follows : Given n integers $a_1, a_2, \cdots, a_n$ such that $\sum_{j=1}^{n} a_j = 2A$, is there set B such that $\sum_{j \in B} a_j = A$?

Lemma 1 : Even when the number of agents is two, Problem P1 is NP-hard.

Proof : The decision version of Problem 1 with two agents can be stated as follows : Given thresholds λ_1 and λ_2, is there a feasible schedule σ such that $\sum_{j \in E^1(\sigma)} w_j^1 \geq \lambda_1$ and $\sum_{j \in E^2(\sigma)} w_j^2 \geq \lambda_2$?

Given an instance of the partition problem, we can construct an instance of Problem P1 as follows. Consider the set of n jobs for each agent $J^1 = \{J_1^1, \cdots, J_n^1\}$ and $J^2 = \{J_1^2, \cdots, J_n^2\}$ such that for $j = 1, 2, \cdots, n$,

$$p_j^1 = p_j^2 = 1, \quad d_j^1 = d_j^2 = j, \quad w_j^1 = w_j^2 = a_j,$$

and $\lambda_1 = \lambda_2 = A$. This reduction can be carried out in polynomial time.

Suppose that there is a set $\bar{B}$ to the partition problem such that $\sum_{j \in \bar{B}} a_j = A$. We can construct a schedule $\sigma = (E^1(\sigma), E^2(\sigma))$, where $E^1(\sigma) = \bar{B}$ and $E^2(\sigma) = \{1, 2, \cdots, n\} \setminus \bar{B}$. It is observed that since $p_j^1 = p_j^2 = 1$ and $d_j^1 = d_j^2 = j$, $j = 1, 2, \cdots, n$, schedule σ is feasible, and

$$\sum_{j \in E^1(\sigma)} w_j^1 = \sum_{j \in \bar{B}} a_j = A \text{ and } \sum_{j \in E^2(\sigma)} w_j^2 = \sum_{j \in \{1, 2, \cdots, n\} \setminus \bar{B}} a_j = A.$$

Suppose that there exists a feasible schedule $\bar{\sigma} = (E^1(\bar{\sigma}), E^2(\sigma))$ for Problem P1 such that

$$\sum_{j \in E^1(\bar{\sigma})} w_j^1 \geq A \text{ and } \sum_{j \in E^2(\bar{\sigma})} w_j^2 \geq A. \qquad (1)$$

Since $\bar{\sigma}$ is a feasible schedule, $d_j^1 = d_j^2 = j$ and

$w_j^1 = w_j^2 = a_j,\ j=1,\ 2,\ \cdots,\ n,$

$$\sum_{j\in E^1(\bar{\sigma})} w_j^1 + \sum_{j\in E^2(\bar{\sigma})} w_j^2 = \sum_{j=1}^{n} a_j = 2A. \qquad (2)$$

By inequalities (1) and (2), $\sum_{j\in E^1(\bar{\sigma})} w_j^1 = \sum_{j\in E^2(\bar{\sigma})} w_j^2 = A$. Thus, since $w_j^1 = w_j^2 = a_j,\ j=1,\ 2,\cdots,\ n,$ sets $E^1(\bar{\sigma})$ and $E^2(\bar{\sigma})$ are the solutions to the partition problem. The proof is complete. ■

Lemma 2 : There exists a pseudo-polynomial-time algorithm for Problem P1 with the fixed number of agents.

Proof : We show that if g is fixed, Problem P1 can be reduced to the shortest path problem in pseudo-polynomial time. Let $N(k_1,\ l_1,\ k_2,\ l_2,\ w_2,\ \cdots,\ k_g,\ l_g,\ w_g)$ denote a node that represents the following :

- Jobs in $\cup_{h=1}^{g}\{1,\cdots,\ k_h\}$ have been determined to be processed just-in-time or tardy;
- Job l_h is the last JIT job in $\{J_1^h,\ \cdots,\ J_{k_h}^h\}$, $h=1,\ 2,\ \cdots,\ g$;
- w_h is the total weighted number of jobs which are determined to be processed just-in-time among jobs in $\{1,\cdots,\ k_h\},\ h.=2,\cdots,\ g$

Let $N(0,\ 0,\ 0,\ 0,\ 0,\ \cdots,\ 0,\ 0,\ 0)$ and $N(n_1+1,\ end,\ n_2+1,\ end,\ end,\ \cdots,\ n_g+1,\ end,\ end)$ be the source node and sink node, respectively. For $(k_1,,\ \cdots,\ k_g) \neq (n_1,\cdots,\ n_g)$ let $N(k_1,\ l_1,\ k_2,\ l_2,\ w_2,\ \cdots,\ k_g,\ l_g\ w_g)$ with $d_{k_{h'}+1}^{h'} = \min\{d_{k_1+1}^1,\ \cdots,\ d_{k_g+1}^g\}$ and $d_{\max} = \max\{d_{k_1}^1,\ \cdots,\ d_{k_g}^g\}$ be connected to the following nodes : (For the consistency of notations, let $d_{k_h+1}^h = \infty$ if $k_h = n_h,\ h=1,\ 2,\ \cdots,\ g$)

(1) $N(k_1+1,\ k_1+1,\ k_2,\ l_2,\ w_2,\ \cdots,\ k_g,\ l_g,\ w_g)$ if $h'=1$ and $d_{\max}+p_{k_1+1}^1 \le d_{k_1+1}^1$,

(2) $N(k_1+1,\ l_1,\ k_2,\ l_2,\ w_2,\ \cdots,\ k_g,\ l_g,\ w_g)$ if $h'=1$,

(3) $N(k_1,\ l_1,\ k_2,\ l_2,\ w_2,\ \cdots,\ k_{h'}+1,\ k_{h'}+1,\ w_{h'}+w_{k_{h'}+1}^{h'},\ \cdots,\ k_g,\ l_g,\ w_g)$ if $h' \ge 2$ and $d_{\max}+p_{k_{h'}+1}^{h'} \le d_{k_{h'}+1}^{h'}$,

(4) $N(k_1,\ l_1,\ k_2,\ l_2,\ w_2,\ \cdots,\ k_{h'}+1,\ l_{h'},\ w_{h'},\cdots,\ k_g,\ l_g,\ w_g)$ if $h' \ge 2$.

The distances of the edges between $N(k_1,\ l_1,\ k_2,\ l_2,\ w_2,\ \cdots,\ k_g,\ l_g,\ w_g)$ and the first nodes is $-w_{k_1+1}^1$, while the distance of the edge between the remaining nodes are zeros.

For $(k_1,\ \cdots,\ k_g) = (n_1,\ \cdots,\ n_g)$, let $N(k_1,\ l_1,\ k_2,\ l_2,\ w_2,\ \cdots,\ k_g,\ l_g,\ w_g)$ be connected to $N(n_1+1, end,\ n_2+1,\ end,\ end,\ \cdots,\ n_g+1,\ end,\ end)$. If $w_h \le \omega_h,\ h=1,\ 2,\ \cdots,\ g$, then the distance of the edge between $N(k_1,\ l_1,\ k_2,\ l_2,\ w_2,\ \cdots,\ k_g,\ l_g,\ w_g)$ and the sink node is zero while otherwise, that distance is ∞. Note that the reduced graph is acyclic. The objective is to find the shortest path between the source and the sink nodes. It is clear that the shortest path of the reduced graph is equivalent to the optimal solution of Problem P1.

The number of nodes in the reduced graph is $O\left(n^{2g}\prod_{h=1}^{g}\sum_{j=1}^{n_h} w_j^h\right)$. Since the number of edges emanating from each node is $O(g)$, the total number of edges is $O\left(gn^{2g}\prod_{h=1}^{g}\sum_{j=1}^{n_h} w_j^h\right)$. Since the reduced graph is acyclic, we can now use the algorithm of Ahuja et al. [3] to obtain the shortest path between the source and the sink nodes. The running time of the algorithm becomes $O\left(gn^{2g}\prod_{h=1}^{g}\sum_{j=1}^{n_h} w_j^h\right)$. The proof is complete. ■

Theorem 1 : *Problem P1 with the fixed number of agents is NP-hard in the ordinary sense.*

Proof : This holds immediately from Lemmas 1 and 2. ■

5. Problem P2

In this section, we show that Problem P2 with the fixed number of agents is polynomially solvable. Note that since the weights are identical in Problem P2, without loss of generality, let $w_j^h = 1,\ j = 1,\ 2,\ \cdots,\ n_h,\ h = 1,\ 2,\ \cdots,\ g$.

Let Problem P2′ be the problem identical to Problem P2 except the due dates as follows :

$$\bar{d}_j^h = d_j^h + j \cdot \sum_{j=1}^{h} \epsilon^j,\ j = 1,\ 2,\ \cdots,\ n_h,\ h = 1,\ 2,\ \cdots,\ g,$$

where $\epsilon > 0$ is sufficiently small and ϵ^j is ϵ to the power of j. From the definition of a JIT job, it is easy to show that σ is an efficient schedule in Problem P2′ if and only if it is an efficient one in Problem P2. Thus, henceforth, we assume that all due dates are different.

Let a $(k_1,\ l_1,\ \cdots,\ k_g,\ l_g)$ schedule be defined as follows :

- Jobs belonging to $\cup_{h=1}^{g}\left\{J_1^h, \ldots, J_{k_h}^h\right\}$ have been considered;
- Job l_h is the last JIT job in $\left\{J_1^h,\ \cdots,\ J_{k_h}^h\right\}$, $h = 1,\ 2,\ \cdots,\ g$.

Definition : Let $\bar{\sigma} = (E^1(\bar{\sigma}),\ \cdots,\ E^g(\bar{\sigma}))$ and $\hat{\sigma} = (E^1(\hat{\sigma}),\ \cdots,\ E^g(\hat{\sigma}))$ be two feasible $(k_1,\ l_1,\ \cdots,\ k_g,\ l_g)$ schedules. We say that $\bar{\sigma}$ $(k_1,\ l_1,\ \cdots,\ k_g,\ l_g)$-*dominates* $\hat{\sigma}$ if one of the following conditions is satisfied :

- $\sum_{h=1}^{q} k_h = n$ and $|E^h(\bar{\sigma})| \geq |E^h(\hat{\sigma})|,\ h = 1,\ 2,\ \cdots,\ g$;
- $\sum_{h=1}^{q} k_h < n$,

  $\sum_{h=1}^{q} \sum_{j \in E^h(\bar{\sigma})} p_{1,\,j}^h \leq \sum_{h=1}^{q} \sum_{j \in E^h(\hat{\sigma})} p_{1,\,j}^h$ and

  $|E^h(\bar{\sigma})| \geq |E^h(\hat{\sigma})|,\ h = 1,\ 2,\ \cdots,\ g$; (3)
- $\sum_{h=1}^{q} k_h < n$, equality holds in inequalities (3) and

  $d_{\bar{\sigma}(j')} < d_{\hat{\sigma}(j')}$ and $d_{\bar{\sigma}(j)} = d_{\hat{\sigma}(j)}$,

  $j = j' + 1,\ \cdots,\ \sum_{h=1}^{q} |E^h(\bar{\sigma})|$.

It is easy to see that the definition above does not reduce the cardinality of set E^*.

Definition : $\bar{\sigma}$ is a $(k_1,\ l_1,\ \cdots,\ k_g,\ l_g)$-*efficient* schedule if any $(k_1,\ l_1,\ \cdots,\ k_g,\ l_g)$ schedules cannot $(k_1,\ l_1,\ \cdots,\ k_g,\ l_g)$-dominate $\bar{\sigma}$.

From the above definitions, the following observation can be obtained.

Observation 1 : For each $(k_1,\ l_1,\ \cdots,\ k_g,\ l_g)$, the number of $(k_1,\ l_1,\ \cdots,\ k_g,\ l_g)$-efficient schedules is at most one.

Lemma 3 : In a $(k_1,\ l_1,\ \cdots,\ k_g,\ l_g)$-efficient schedule,

i) If $d_{k_h}^h = \max\left\{d_{k_1}^1,\ \cdots,\ d_{k_g}^g\right\}$ and $k_{h'} = l_{h'}$, then a $(k_1,\ l_1, \cdots,\ k_{h'-1},\ l_{h'-1},\ k_{h'} - 1,\ j_{h'},\ k_{h'+1},\ l_{h'+1},\ \cdots,\ k_g,\ l_g)$-efficient schedule can be obtained by eliminating job $J_{k_h}^{h'}$ from $(k_1,\ l_1,\ \cdots,\ k_g,\ l_g)$-efficient schedule, where $j_{h'}$ is the index of the JIT job of agent h' with the second largest due date in the $(k_1,\ l_1,\ \cdots,\ k_g,\ l_g)$-effi-

cient schedule.

ii) If $d^h_{k_h} = \max\{d^1_{k_1}, \cdots, d^g_{k_g}\}$ and $k_{h'} \neq l_{h'}$, then a $(k_1, l_1, \cdots, k_{h'-1}, l_{h'-1}, k_{h'}-1, l_{h'}, k_{h'+1}, l_{h'+1}, \cdots, k_g, l_g)$-efficient schedule can be obtained by eliminating job $J^{h'}_{k_{h'}}$ from a $(k_1, l_1, \cdots, k_g, l_g)$-efficient schedule.

Proof : This holds immediately from the definition of a $(k_1, l_1, \cdots, k_g, l_g)$-efficient schedule. ■

Based on Observation 1 and Lemma 3, we can construct the following algorithm. The idea behind this algorithm is to construct the $(k'_1, l'_1, \cdots, k'_g, l'_g)$-efficient schedules with $\sum_{h=1}^{g} k'_h = k+1$ from the $(k_1, l_1, \cdots, k_g, l_g)$-efficient schedules with $\sum_{h=1}^{g} k_h = k$.

Algorithm for efficient schedules (Algorithm ES)

- For simplicity, let $(E^1, \cdots, E^g)_{k_1, \cdots, k_g}$ be the schedule such that jobs belonging to $\cup_{h=1}^{g} \{J^h_1, \cdots, J^h_{k_h}\}$ have been considered and E^h is the set of JIT jobs of agent h, $h=1, 2, \cdots, g$.

Step 1 : Set $k=1$ and $H_0 = \{(\varnothing, \cdots, \varnothing)_{0, \cdots, 0}\}$. Set $H_i = T_i = \varnothing$, $i=1, 2, \cdots, n$ and $d_{\max} = \max\{d^1_{n_1}, \cdots, d^g_{n_g}\}$

Step 2 : Set $w=0$, and select an arbitrary $(E^1, \cdots, E^g)_{k_1, \cdots, k_g}$ from H_{k-1}, and set $H_{k-1} = H_{k-1} \setminus \{(E^1, \cdots, E^g)_{k_1, \cdots, k_g}\}$.

Step 3 : Set $w=w+1$. If $w=g+1$, then go to Step 5.

Step 4 : If $k_w = n_w$, then go to Step 3.

Step 4-1 : If $J^w_{k_w+1}$ can become a JIT job without making JIT jobs in $(E^1, \cdots, E^g)_{k_1, \cdots, k_g}$ non-JIT, then construct two schedules: one is constructed by letting $J^w_{k_w+1}$ be completed exactly at $d^w_{k_w+1}$ and the other by processing $J^w_{k_w+1}$ after $d_{\max}$ (For simplicity, let the two schedules be denoted σ_1 and σ_2, respectively). Set $T_k = T_k \cup \{\sigma_1, \sigma_2\}$ and go to Step 3.

Step 4-2 : Otherwise, set $T_k = T_k \cup \{\sigma_2\}$ and go to Step 3.

Step 5 : If $H_{k-1} \neq \varnothing$, then go to Step 2, whereas otherwise, construct H_k by selecting $(k_1, l_1, \cdots, k_g, l_g)$-efficient schedules from T_k.

Step 6 : If $k=n$, then $E^* = H_n$ and stop, whereas otherwise, set $k=k+1$ and go to Step 2.

Lemma 4 : Algorithm ES produces all $(k_1, l_1, \cdots, k_g, l_g)$-efficient schedules for $0 \le l_h \le k_h \le n_h$, $h=1, 2, \cdots, g$.

Proof : By induction. If $\sum_{h=1}^{g} k_h = 1$, then Algorithm ES finds $(k_1, l_1, \cdots, k_g, l_g)$-efficient schedules. Suppose that Lemma 4 holds for the case with $\sum_{h=1}^{g} k_h = k$. By Lemma 3 and Steps 4 and 5 of Algorithm ES, Lemma 4 holds for the case with $\sum_{h=1}^{g} k_h = k+1$. This completes proof. ■

Theorem 2 : *Problem P2 with the fixed number of agents is polynomially solvable.*

Proof : Since $|H_k| \le k^{2g}$ and $|T_k| \le 4k^{2g}$, and Step

5 is done in $O(k^{2g+1})$, $k=1, 2, \cdots, n$, Algorithm ES is done in $O(n^{2g+2})$. Thus, Theorem 2 holds from Lemma 4. ■

Numerical Example : Consider two-agent, two-machine flow shop problem where each agent has three jobs, presented in <Table 2>.

〈Table 2〉 Job Parameter Values for the Numerical Example

Agent 1	1	2	3
$p^1_{1,j}$	2	4	3
$p^1_{2,j}$	3	2	3
d^1_j	6	8	11

Agent 2	1	2
$p^2_{1,j}$	4	2
$p^2_{2,j}$	2	4
d^2_j	7	10

This example can be solved as follows :

Step 1 $k=1$, $H_0=\{(\varnothing,\varnothing)_{0,0}\}$, and $H_i=T_i=\varnothing$, $i=1, \cdots, 5$.

Stage 1 : Find (k_1, l_1, k_2, l_2)-efficient schedules with $k_1+k_2=1$.

Step 2 $w=0$ and $H_0=\varnothing$.

Step 3 $w=1$.

Step 4 Go to Step 4-1.

Step 4-1 $T_1=\{(\{1\}, \varnothing)_{1,0}, (\varnothing, \varnothing)_{1,0}\}$ and go to Step 3.

Step 3 $w=2$.

Step 4 Go to Step 4-1.

Step 4-1 $T_1=\{(\{1\},\varnothing)_{1,0}, (\varnothing, \varnothing)_{1,0}, (\varnothing, \{1\})_{0,1}, (\varnothing, \varnothing)_{0,1}\}$ and go to Step 3.

Step 3 $w=3$ and go to Step 5.

Step 5 $H_1=\{(\{1\}, \varnothing)_{1,0}, (\varnothing, \varnothing)_{1,0}, (\varnothing, \{1\})_{0,1}, (\varnothing, \varnothing)_{0,1}\}$.

Step 6 $k=2$, and go to Step 2.

(Note : For simplicity, henceforth, we will omit the precise procedures to find the efficient schedules.)

Stage 2 : The set of (k_1, l_1, k_2, l_2)-efficient schedules with $k_1+k_2=2$ is as follows :

$H_2=\{(\{1, 2\}, \varnothing)_{2,0}, (\{1\}, \varnothing)_{2,0}, (\{1\}, \varnothing)_{1,1}, (\{2\}, \varnothing)_{2,0}, (\varnothing, \varnothing)_{2,0}, (\varnothing, \{1\})_{1,1}, (\varnothing, \varnothing)_{1,1}, (\varnothing, \{1\})_{0,2}, (\varnothing, \{2\})_{0,2}, (\varnothing, \varnothing)_{0,2}\}$

Stage 3 : The set of (k_1, l_1, k_2, l_2)-efficient schedules with $k_1+k_2=3$ is as follows :

$H_3=\{(\{1, 2\}, \varnothing)_{3,0}, (\{1, 2\}, \varnothing)_{2,1}, (\{1, 3\}, \varnothing)_{3,0}, (\{1\}, \varnothing)_{3,0}, (\{1\}, \varnothing)_{2,1}, (\{1\}, \{2\})_{1,2}, (\{1\}, \varnothing)_{1,2}, (\{2, 3\}, \varnothing)_{3,0}, (\{2\}, \varnothing)_{3,0}, (\{2\}, \varnothing)_{2,1}, (\{3\}, \varnothing)_{3,0}, (\varnothing, \varnothing)_{3,0}, (\varnothing, \{1\})_{2,1}, (\varnothing, \varnothing)_{2,1}, (\varnothing, \{1\})_{1,2}, (\varnothing, \{2\})_{1,2}, (\varnothing, \varnothing)_{1,2}\}$

Stage 4 : The set of (k_1, l_1, k_2, l_2)-efficient schedules with $k_1+k_2=4$ is as follows :

$H_4=\{(1, 2\}, \varnothing)_{3,1}, (\{1, 2\}, \varnothing)_{2,2}, (\{1, 3\}, \varnothing)_{3,1}, (\{1\}, \varnothing)_{3,1}, (\{1\}, \{2\})_{2,2}, (\{1\}, \varnothing)_{2,2}, (\{2\}, \varnothing)_{3,1}, (\{2\}, \varnothing)_{2,2}, (\{3\}, \{1\})_{3,1}, (\{3\}, \varnothing)_{3,1}, (\varnothing, \{1\})_{3,1}, (\{3\}, \{1\})_{3,1}, (\varnothing, \{1\})_{3,1}, (\varnothing, \{1\})_{2,2}, (\varnothing, \{2\})_{2,2}\}$.

Stage 5 : The set of (k_1, l_1, k_2, l_2)-efficient schedules with $k_1+k_2=5$ is as follows :

$H_5=\{(\{1, 2\}, \varnothing)_{3,2}, (\{1\}, \{2\})_{3,2}\}$.

6. Conclusions

We consider two versions of a multi-agent scheduling problem in order to maximize the weighted number of JIT jobs of each agent. The first is the constrained optimization model while the second is the Pareto optimization model in two-machine flow shop. We show that when the number of agents is fixed, the first model is

NP-hard in the ordinary sense and the second one with identical weights is polynomially solvable.

For future research, it would be interesting to determine the computational complexity of the case with the arbitrary number of agents.

References

[1] Agnetis, A., P.B. Mirchandani, D. Pacciarelli, and A. Pacifici, "Scheduling problems with two competing agents," *Operations Research*, Vol.52, No.2(2004), pp.229-242.

[2] Agnetis, A., P.B. Mirchandani, D. Pacciarelli, and A. Pacifici, "Multi-agent single machie sheduling," *Annal of Operations Research*, Vol.150(2007), pp.3-15.

[3] Ahuja, R.A., K. Mehlhorn, J.B. Orlin, and R.E. Tarjan, "Faster algorithms for the shortest path problem," *Journal of the Association for Computing Machinery*, Vol.37(1990), pp.213-223.

[4] Arbib, C., S. Smriglio, and M. Servilio, "A competitive scheduling problem and its relevances to UMTS channel assignment," *Networks*, Vol.44, No.2(2004), pp.132-141.

[5] Cepek, R.L. and S.C. Sung, "A quadratic time algorithm to maximize the number of just-in-time jobs on identical parallel machines," *Computers and Operations Research*, Vol.32(2005), pp.3265-3271.

[6] Cheng, T.C.E., C.T. Ng, and J.J. Yuan, "Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs," *Theoretical Computer Science*, Vol.362(2006), pp.273-281.

[7] Choi, B.C. and S.H. Yoon, "Maximizing the weighted number of just-in-time jobs in flow shop scheduling," *Journal of Scheduling*, Vol.10(2007), pp.237-243.

[8] Cormen, T.H. and C.E. Leiserson, "Introduction to algorithm, Cambridge : MIT press, 1996.

[9] Hall, N.G. and C.N. Potts, "Rescheduling for New Orders," *Operations Research*, Vol.52 (2004), pp.440-453.

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

[11] Lann, A. and G. Mosheiov, "Single machine scheduling to minimize the number of early and tardy jobs," *Computers and Operations Research*, Vol.23(1996), pp.769-781.

[12] Lee, K.B., B.C. Choi, J.Y.T. Leung, and M.L. Pinedo, "Approximation algorithms for multi-agent scheduling to minimize total weight completion time," *Information Processing Letters*, Vol.109(2009), pp.913-917.

[13] Lee, W.C., S.K. Chen, C.W. Chen, and C.C. Wu, "A two-machine flowshop problem with two agents," *Computers and Operations* Research, Vol.109(2009), pp.913-917.

[14] Leung, J.Y.T., M.L. Pinedo, and G. Wan, "Competitive two-agent scheduling and its applications," *Operations Research*, Vol.58 (2010), pp.458-469.

[15] Ng, C.T., T.C.E. Cheng, and J.J. Yuan, "A note on the complexity of the problem of two-agent scheduling on a single machine," *Journal of Combinatorial Optimization*, Vol.12 (2006), pp.387-394.

[16] Sung, S.C. and M. Vlach, "Maximizing weighted number of just-in-time jobs on unrelated parallel machines," *Journal of Scheduling*, Vol.8(2005), pp.453-460.