Optimization Algorithms for a Two-Machine Permutation Flowshop with Limited Waiting Times Constraint and Ready Times of Jobs

Seong-Woo Choi*

Abstract

In this research, we develop and suggest branch and bound algorithms for a two-machine permutation flowshop scheduling problem with the objective of minimizing makespan. In this scheduling problem, after each job is operated on the machine 1 (first machine), the job has to start its second operation on machine 2 (second machine) within its corresponding limited waiting time. In addition, each job has its corresponding ready time at the machine 1. For this scheduling problem, we develop various dominance properties and three lower bounding schemes, which are used for the suggested branch and bound algorithm. In the results of computational tests, the branch and bound algorithms with dominance properties and lower bounding schemes, which are suggested in this paper, can give optimal solution within shorter CPU times than the branch and bound algorithms without those. Therefore, we can say that the suggested dominance properties and lower bounding schemes are efficient.

Keywords : Scheduling, Permutation Flowshop, Branch And Bound, Limited Waiting Time, Ready Times

Received : 2015. 04. 01. Revised : 2015. 04. 20. Final Acceptance : 2015. 04. 20.
※ This work was supported by Kyonggi University Research Grant 2011.
* Department of Business Administration, Kyonggi University, e-mail : swchoi@kyonggi.ac.kr

1. Introduction

This scheduling problem can be denoted by F2/r_i, max-wait/$C_{\max}$ in the three-field notation of Graham et al. [1979], where r_i and max-wait mean that jobs have cannot be operated on the machine 1 earlier than their ready times and have to be operated on the machine 2 within a certain limited time after those jobs are operated on the machine 1, respectively, and $C_{\max}$ is the makespan.

We can found the above scheduling problem in a sub workstation of semiconductor fab line. At the workstation (process section) consisting of continuous clean and diffusion processes in semiconductor wafer fabrication line, after a chemical treatment process for a wafer lot is completed on a clean machine, the next process for the wafer lot must be started on a diffusion machine within a pre-determined time period, and if the next process for the wafer lots is delayed, it must be abandoned or re-processed because the chemical treatment is no longer effective after the time period [Joo and Kim, 2009]. Such a time period between the two processes is called the limited waiting time in scheduling research and the lengths of these time periods may differ for different wafer lots according to their chemical characteristics [Joo and Kim, 2009]. In addition, since the above subsystem is not the first operation of wafer fabrication in general, jobs arrive at this workstation dynamically, that is, jobs (processing of wafer lots) may have different ready times in the scheduling problem for the first machine [Choi et al., 2010]. In this study, we develop scheduling algorithms for subsystems of a wafer fabrication system, and hence we (need to) consider the limited waiting time constraint and ready times.

There are many researches for typical flowshop scheduling problem [Chen et al., 2000; Framinan et al., 2004; Gupta and Stafford, 2006]. However, there are few researches for the flowshop scheduling problem with limited waiting time as follows. Yang and Chern [1995] and Bouquard and Lente [2006] suggested branch and bound algorithms with upper bounds and lower bounds. Also, Joo and Kim [2009] developed several dominance properties and lower bounds for a branch and bound algorithm, and Attar et al. [2013] dealt with hybrid flexible flowshop scheduling problem with unrelated parallel machine and limited waiting times. In addition, there are many researches for the flowshop scheduling problems with ready times or release dates of jobs. Specially, Tadei et al. [1998] and Potts [1985] suggested branch and bound algorithm and investigated the worst-case performance of five approximation algorithms for minimizing makespan on the two-machine flowshop with release date, respectively. Also, Hall [1994] and Chu [1992] proposed a polynomial approximation scheme and a branch and bound algorithm for minimizing makespan and total tardiness on the two-machine flow-shop with release date, respectively. Choi [2014] suggested several heuristic algorithms for the two- machine permutation flowshop scheduling problem with both of limited waiting time and ready times. However, there is no research on branch and bound algorithm for the two-machine permutation flowshop scheduling problem with both of limited waiting time and ready times to the best of our knowledge.

This scheduling problem is NP-hard in the strong sense, since the two-machine permutation flowshop scheduling problem with limited waiting time is NP-hard in the strong sense [Joo and Kim, 2009; Yang and Chern, 1995], which is a special case (ready times of jobs are 0) of this scheduling problem. To develop an efficient branchand bound algorithm, we develop various dominance properties and three lower bounding schemes, and we use three existing heuristic algorithms [Choi, 2015] to get the initial upper bounds of the branch and bound algorithms.

2. Problem Description

In this scheduling problem, n jobs should be operated on two-machine flowshop in the order of machine 1 and then machine 2, and the following assumptions are made in this study.

1) In the two-machine flowshop scheduling problem, we have a given set of jobs with different ready times, that is, each job can be operated on the machine 1 at its ready time.
2) The operation times and limited waiting times of the jobs are known and different from each other.
3) Each job should start its operation on machine 2 within its corresponding limited waiting time after the job is operated on the machine 1.

In this research, only permutation schedules are considered, that is, operation order of jobs is same on the both machines. Permutation schedules are dominant in the ordinary two-machine flowshop scheduling problem with the objective of minimizing makespan [Baker, 1974; Choi and Kim, 2007]. However, in the cases of two-machine scheduling problem with the limited waiting times and ready times of jobs, permutation schedules are not dominant. The following example may be the case that a non-permutation schedule is better than the best among permutation schedules.

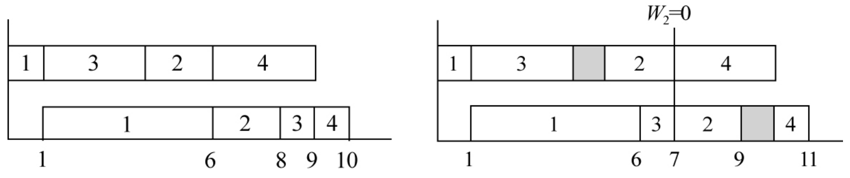
Counter example for the dominance of permutations schedules

We assume that four jobs should be scheduled, and their processing times, limited waiting times, and ready times are given in <Table 1>. As shown in <Figure 1>, the best permutation schedule can be obtained by sequence (1, 3, 2, 4), while there is a better non-permutation schedule, in which the sequence on machine 1 is (1, 3, 2, 4) and the sequence on machine 2 is (1, 2, 3, 4).

<Table 1> Processing Times, Limited Waiting Times and Ready Times for the Example

	j = 1	j = 2	j = 3	j = 4
p_{j1}	1	2	3	3
p_{j2}	5	2	1	1
w_j	1	0	6	1
r_j	0	2	1	6

However, Choi and Kim [2007] described that permutation schedules are preferred to non-permutation schedules in most real systems because of the ease of implementation or material flow management, and non-permutation schedules are not feasible in many cases because of



(a) Non-permutation schedule (b) Best permutation schedule

<Figure 1> An Example Showing Non-Dominance of Permutation Schedules

technical constraints of material handling systems. Therefore, in this research, we consider only permutation schedules.

The following symbols are used in this research.

i, j	indices of jobs
k	index of machines k = 1, 2
p_{ik}	operation time of job i on machine k
w_i	limited waiting time of job i
r_i	ready time of job i
σ	partial sequence
$\sigma_{ij\cdots m}$	partial schedule obtained with σ followed by jobs $i, j, \cdots, m$ in this order
U	set of unscheduled jobs, that is, those are not included in σ
$c_{[m]k}$	completion time of the job at the m-th position in a (partial) schedule on machine k
$C_k(\sigma)$	completion time of the last positioned job in a partial schedule σ on machine k

With the above symbols, completion time of each job in a given schedule (sequence) can be expressed as follows.

$$c_{[1]1} = r_{[1]} + p_{[1]1} \quad \text{(E.1)}$$

$$c_{[1]2} = c_{[1]1} + p_{[1]2} = r_{[1]} + p_{[1]1} + p_{[1]2} \quad \text{(E.2)}$$

$$c_{[j]1} = \max\{\max(r_{[j]}, c_{[j-1]1}) + p_{[j]1}, c_{[j-1]2} - w_{[j]}\} \text{ for } j = 2, \cdots, n \quad \text{(E.3)}$$

$$c_{[j]2} = \max\{c_{[j]1}, c_{[j-1]2}\} + p_{[j]2} \text{ for } j = 2, \cdots, n \quad \text{(E.4)}$$

3. Dominance Properties

In this section, we develop dominance properties to reduce the number of sub-problems to be considered in the suggested branch and bound algorithms, that is, we can eliminate partial schedules that are dominated by others in the branch and bound algorithm [Choi and Kim, 2007].

The developed dominance properties are developed based on the Johnson's algorithm [Johnson, 1954] for two-machine permutation flowshop scheduling problems (in the Johnson's algorithm, job i precedes job j in an optimal sequence if min{p_{i1}, p_{j2}} min{p_{i2}, p_{j1}}) and properties suggested by Joo and Kim [2009] and Tadei et al. [1998]. Joo and Kim [2009] suggested dominance properties for a two-machine flowshop with the limited waiting times of jobs and Tadei et al. [1998] suggested dominance properties for a two-machine flowshop with the ready times of jobs.

Proposition 1 : *There is a partial schedule σ, and if we have two jobs i and j ($i \notin \sigma$, $j \notin \sigma$) satisfying the three conditions such as* (C.1) $C_2(\sigma) \le r_i \le r_j$, (C.2) min{$p_{i1}$, p_{j2}} $\le$ min{p_{i2}, p_{j1}} *and*

(C.3) $p_{i2} \le p_{j1} + w_j$, *there is an optimal solution among schedules that start with* σij (that is, σij *dominates* σji).

Proof : We can complete this proof by showing that $C_k(\sigma ij) \le C_k(\sigma ji)$ for $k = 1, 2$. The following equalities [(A.1)-(A.8)] are the obtained completion times (of jobs i and j in schedules σij and σji) from equalities (E.1)-(E.4) at the end of section 2 :

$$C_1(\sigma i) = \max\{\max(r_i, C_1(\sigma))+p_{i1}, C_2(\sigma)-w_i\} \quad (A.1)$$
$$C_2(\sigma i) = \max\{C_1(\sigma i), C_2(\sigma)\}+p_{i2} \quad (A.2)$$
$$C_1(\sigma ij) = \max\{\max(r_j, C_1(\sigma i))+p_{j1}, C_2(\sigma i)-w_j\} \quad (A.3)$$
$$C_2(\sigma ij) = \max\{C_1(\sigma ij), C_2(\sigma i)\}+p_{j2} \quad (A.4)$$
$$C_1(\sigma j) = \max\{\max(r_j, C_1(\sigma))+p_{j1}, C_2(\sigma)-w_j\} \quad (A.5)$$
$$C_2(\sigma j) = \max\{C_1(\sigma j), C_2(\sigma)\}+p_{j2} \quad (A.6)$$
$$C_1(\sigma ji) = \max\{\max(r_i, C_1(\sigma j))+p_{i1}, C_2(\sigma j)-w_i\} \quad (A.7)$$
$$C_2(\sigma ji) = \max\{C_1(\sigma ji), C_2(\sigma j)\}+p_{i2} \quad (A.8)$$

In the first phase, we show $C_1(\sigma ij) \le C_1(\sigma ji)$.

Since the condition (C.1) $C_2(\sigma) \le r_i \le r_j$ of this proposition, (A.1), (A.2), (A.5) and (A.6) can be reduced to $C_1(\sigma i) = r_i+p_{i1}$, $C_2(\sigma i) = r_i+p_{i1}+p_{i2}$, $C_1(\sigma j) = r_j+p_{j1}$ and $C_2(\sigma j) = r_j+p_{j1}+p_{j2}$, respectively.

In addition, from (A.3), we have

$$\begin{aligned} C_1(\sigma ij) &= \max\{\max(r_j, r_i+p_{i1})+p_{j1}, r_i+p_{i1}+p_{i2}-w_j\} \\ &= \max\{r_j+p_{j1}, r_i+p_{i1}+p_{j1}, r_i+p_{i1}+p_{i2}-w_j\}. \end{aligned}$$

Here, from the condition (C.3) $p_{i2} \le p_{j1}+w_j$ of this proposition, we know $r_i+p_{i1}+p_{i2}-w_j \le r_i+p_{i1}+p_{j1}$, and therefore, we have

$$C_1(\sigma ij) = \max\{r_j+p_{j1}, r_i+p_{i1}+p_{j1}\} \quad (A.9)$$

Also, from the reduced (A.5), the reduced (A.6) and (A.7), we have

$$\begin{aligned} C_1(\sigma ji) &= \max\{\max(r_i, C_1(\sigma j))+p_{j1}, \\ &\quad C_2(\sigma j)-w_i\} \\ &= \max\{r_j+p_{j1}+p_{i1}, r_j+p_{j1}+p_{j2}-w_i\} \quad (A.10) \end{aligned}$$

From (A.9) and (A.10), we have $C_1(\sigma ij) \le C_1(\sigma ji)$.

In the second phase, we show $C_2(\sigma ij) \le C_2(\sigma ji)$.

From the reduced (A.2), (A.4), and (A.9), we have

$$\begin{aligned} C_2(\sigma ij) &= \max\{C_1(\sigma ij), C_2(\sigma i)\}+p_{j2} \quad (A.11) \\ &= \max\{\max\{r_j+p_{j1}, r_i+p_{i1}+p_{j1}\}, \\ &\quad r_i+p_{i1}+p_{i2}\}+p_{j2}, \\ &= \max\{r_j+p_{j1}+p_{j2}, r_i+p_{i1}+p_{j1}+p_{j2}, \\ &\quad r_i+p_{i1}+p_{i2}+p_{j2}\} \\ &\equiv \max(A_1, A_2, A_3), \end{aligned}$$

where $A_1 = r_j+p_{j1}+p_{j2}$, $A_2 = r_i+p_{i1}+p_{j1}+p_{j2}$ and $A_3 = r_i+p_{i1}+p_{i2}+p_{j2}$.

From the reduced (A.6), (A.8), and (A.10), we have

$$\begin{aligned} C_2(\sigma ji) &= \max\{C_1(\sigma ji), C_2(\sigma j)\}+p_{i2} \\ &= \max\{\max\{r_j+p_{j1}+p_{i1}, r_j+p_{j1}+p_{j2}- w_i\}, \\ &\quad r_j+p_{j1}+p_{j2}\}+p_{i2} \\ &= \max\{r_j+p_{j1}+p_{i1}+p_{i2}, r_j+p_{j1}+p_{j2}-w_i+p_{i2}, \\ &\quad r_j+p_{j1}+p_{j2}+p_{i2}\} \\ &\equiv \max\{B_1, B_2, B_3\}, \quad (A.12) \end{aligned}$$

where $B_1 = r_j+p_{j1}+p_{i1}+p_{i2}$, $B_2 = r_j+p_{j1}+p_{j2}-w_i+p_{i2}$ and $B_3 = r_j+p_{j1}+p_{j2}+p_{i2}$.

First, we have $A_1 \le B_3$, easily.

Second, we consider two cases to show max $(A_2, A_3) \le \max\{B_1, B_2, B_3\}$.

Case 1 : If $\min\{p_{i1}, p_{j2}\} = p_{i1}$, that is, we have $p_{i1} \le \min\{p_{i2}, p_{j1}\}$ from the condition (C.2) $\min\{p_{i1}, p_{j2}\} \le \min\{p_{i2}, p_{j1}\}$ of this proposition, in this case, we have $A_2 \le B_3$ and $A_3 \le B_3$ since the conditions of $p_{i1} \le \min\{p_{i2}, p_{j1}\}$ and $r_i \le r_j$.

Case 2 : If $\min\{p_{i1}, p_{j2}\} = p_{j2}$, that is, we have $p_{j2} \le \min\{p_{i2}, p_{j1}\}$ from the condition (C.2) $\min\{p_{i1}, p_{j2}\} \le \min\{p_{i2}, p_{j1}\}$ of this proposition, in this case, we have $A_2 \le B_1$ and $A_3 \le B_1$ since the conditions of $p_{j2} \le \min\{p_{i2}, p_{j1}\}$ and $r_i \le r_j$.

In conclusion, by showing $C_k(\sigma ij) \le C_k(\sigma ji)$ for $k = 1, 2$, the proof is completed. ■

Proposition 2 : *There is a partial schedule* σ, *and if we have two jobs i and j* ($i \notin \sigma, j \notin \sigma$) *satisfying the four conditions such as* (C.4) $r_i \le C_1(\sigma)$, (C.2) $\min\{p_{i1}, p_{j2}\} \le \min\{p_{i2}, p_{j1}\}$, (C.5) $C_2(\sigma) - C_1(\sigma) \le p_{i1}$ *and* (C.3) $p_{i2} \le p_{j1}+w_j$, *there is an optimal solution among schedules that start with* σij (that is, σij *dominates* σji).

Proof : If we can show that $C_k(\sigma ij) \le C_k(\sigma ji)$ for $k = 1, 2$, this completes the proof. We use equalities (A.1)-(A.8) obtained in the proof of Proposition 1.

In the first phase, we show $C_1(\sigma ij) \le C_1(\sigma ji)$.

Since the conditions (C.4) $r_i \le C_1(\sigma)$ and (C.5) $C_2(\sigma) - C_1(\sigma) \le p_{i1}$ of this proposition, (A.1) and (A.2) can be reduced to $C_1(\sigma i) = C_1(\sigma)+p_{i1}$ and $C_2(\sigma i) = C_1(\sigma)+p_{i1}+p_{i2}$, respectively.

From the condition (C.3) $p_{i2} \le p_{j1}+w_j$ of this proposition and (A.3), we have

$$\begin{aligned} C_1(\sigma ij) &= \max\{\max(r_j, C_1(\sigma)+p_{i1})+p_{j1}, \\ &\quad C_1(\sigma)+p_{i1}+p_{i2}-w_j\} \\ &= \max\{r_j+p_{j1}, C_1(\sigma)+p_{i1}+p_{j1}, \\ &\quad C_1(\sigma)+p_{i1}+p_{i2}-w_j\} \\ &= \max\{\text{ⓐ}r_j+p_{j1}, \text{ⓑ}C_1(\sigma)+p_{i1}+p_{j1}\} \end{aligned} \tag{A.13}$$

Also, from the condition (C.4) $r_i \le C_1(\sigma)$ of this proposition, (A.5), (A.6) and (A.7), we have

$$\begin{aligned} C_1(\sigma ji) &= \max\{\max(r_i, C_1(\sigma j))+p_{i1}, C_2(\sigma j)-w_i\} \\ &= \max\{C_1(\sigma j)+p_{i1}, C_1(\sigma j)+p_{j2}-w_i, \\ &\quad C_2(\sigma)+p_{j2}-w_i\} \\ &= \max\{\text{ⓒ}r_j+p_{j1}+p_{i1}, \text{ⓓ}C_1(\sigma)+p_{j1}+p_{i1}, \\ &\quad C_2(\sigma)-w_j+p_{i1}, r_j+p_{j1}+p_{j2}-w_i, \\ &\quad C_1(\sigma)+p_{j1}+p_{j2}-w_i, \\ &\quad C_2(\sigma)-w_j+p_{j2}-w_i, \\ &\quad C_2(\sigma)+p_{j2}-w_i\} \end{aligned} \tag{A.14}$$

From (A.13) and (A.14), we have $C_1(\sigma ij) \le C_1(\sigma ji)$ since ⓐ $r_j+p_{j1} <$ ⓒ $r_j+p_{j1}+p_{i1}$ and ⓑ $C_1(\sigma)+p_{i1}+p_{j1} =$ ⓓ $C_1(\sigma)+p_{j1}+p_{i1}$.

In the second phase, we show $C_2(\sigma ij) \le C_2(\sigma ji)$. From the reduced (A.2), (A.4), and (A.14), we have

$$\begin{aligned} C_2(\sigma ij) &= \max\{C_1(\sigma ij), C_2(\sigma i)\}+p_{j2} \\ &= \max\{r_j+p_{j1}, C_1(\sigma)+p_{i1}+p_{j1}, \\ &\quad C_1(\sigma)+p_{i1}+p_{i2}\}+p_{j2}, \\ &= \max\{r_j+p_{j1}+p_{j2}, C_1(\sigma)+p_{i1}+p_{j1}+p_{j2}, \\ &\quad C_1(\sigma)+p_{i1}+p_{i2}+p_{j2}\} \\ &\equiv \max(A_1, A_2, A_3) \end{aligned} \tag{A.15}$$

where $A_1 = r_j+p_{j1}+p_{j2}$, $A_2 = C_1(\sigma)+p_{i1}+p_{j1}+p_{j2}$ and $A_3 = C_1(\sigma)+p_{i1}+p_{i2}+p_{j2}$.

From (A.5), (A.6), (A.8), and (A.14), we have

$$\begin{aligned} C_2(\sigma j) &= \max\{C_1(\sigma j),\ C_2(\sigma)\}+p_{j2} \qquad \text{(A.16)} \\ &= \max\{\max\{\max(r_j,\ C_1(\sigma))+p_{j1}, \\ &\quad C_2(\sigma)-w_j\},\ C_2(\sigma)\}+p_{j2} \\ &= \max\{r_j+p_{j1}+p_{j2},\ C_1(\sigma)+p_{j1}+p_{j2}, \\ &\quad C_2(\sigma)-w_j+p_{j2},\ C_2(\sigma)+p_{j2}\} \end{aligned}$$

From (A.8), (A.14) and (A.16), we have

$$\begin{aligned} C_2(\sigma ji) &= \max\{C_1(\sigma ji),\ C_2(\sigma j)\}+p_{i2} \qquad \text{(A.17)} \\ &= \max\{r_j+p_{j1}+p_{i1}+p_{i2}, \\ &\quad C_1(\sigma)+p_{j1}+p_{i1}+p_{i2}, \\ &\quad C_2(\sigma)-w_j+p_{i1}+p_{i2},\ r_j+p_{j1}+p_{j2}-w_i+p_{i2}, \\ &\quad C_1(\sigma)+p_{j1}+p_{j2}-w_i+p_{i2}, \\ &\quad C_2(\sigma)-w_j+p_{j2}-w_i+p_{i2}, \\ &\quad C_2(\sigma)+p_{j2}-w_i+p_{i2},\ r_j+p_{j1}+p_{j2}+p_{i2}, \\ &\quad C_1(\sigma)+p_{j1}+p_{j2}+p_{i2}, \\ &\quad C_2(\sigma)-w_j+p_{j2}+p_{i2}, \\ &\quad C_2(\sigma)+p_{j2}+p_{i2}\} \\ &\equiv \max\{B_1,\ B_2,\ B_3,\ B_4,\ B_5,\ B_6,\ B_7,\ B_8, \\ &\quad B_9,\ B_{10},\ B_{11}\} \\ &\equiv \max\{B_1,\ B_2,\ B_3,\ B_4,\ B_5,\ B_6,\ B_7,\ B_8, \\ &\quad B_9,\ B_{11}\} \end{aligned}$$

where $B_2 = C_1(\sigma)+p_{j1}+p_{i1}+p_{i2}$, $B_8 = r_j+p_{j1}+p_{j2}+p_{i2}$ and $B_9 = C_1(\sigma)+p_{j1}+p_{j2}+ p_{i2}$. Here, B_{10} is eliminated since B_{10} cannot be larger than B_{11}.

First, we have $A_1 \leq B_8$, easily.

Second, we consider two cases to show max $(A_2, A_3) \leq$ max $B_2, B_9\}$.

Case 1 : If $\min\{p_{i1}, p_{j2}\} = p_{i1}$, that is, we have $p_{i1} \leq \min\{p_{i2}, p_{j1}\}$ from the condition (C.2) $\min\{p_{i1}, p_{j2}\} \leq \min\{p_{i2}, p_{j1}\}$ of this proposition, in this case, we have $A_2 \leq B_9$ and $A_3 \leq B_9$.

Case 2 : If $\min\{p_{i1}, p_{j2}\} = p_{j2}$, that is, we have $p_{j2} \leq \min\{p_{i2}, p_{j1}\}$ from the condition (C.2) $\min\{p_{i1}, p_{j2}\} \leq \min\{p_{i2}, p_{j1}\}$ of this proposition, in this case, we have $A_2 \leq B_2$ and $A_3 \leq B_2$.

As a result, we have $C_k(\sigma ij) \leq C_k(\sigma ji)$ for k = 1, 2, and the proof is completed. ■

Proofs for the following propositions 3~5 are not shown in this paper, since their proofs are similar to those for propositions 1~2.

Proposition 3 : *There is a partial schedule* σ, *and if we have two jobs i and j* ($i \not\in \sigma, j \not\in \sigma$) *satisfying the four conditions such as* (C.4) $r_i \leq C_1(\sigma)$, (C.2) $\min\{p_{i1}, p_{j2}\} \leq \min\{p_{i2}, p_{j1}\}$, (C.6) $C_2(\sigma)-C_1(\sigma) \leq p_{i1}+w_j$ *and* (C.7) $C_2(\sigma)+p_{i2}-C_1(\sigma)-p_{i1} \leq p_{j1}+w_j$, *there is an optimal solution among schedules that start with* σij (that is, σij *dominates* σji).

Proposition 4 : *There is a partial schedule* σ, *and if we have two jobs i and j* ($i \not\in \sigma, j \not\in \sigma$) *satisfying the five conditions such as* (C.8) $C_1(\sigma) \leq r_i \leq C_2(\sigma)$, (C.9) $r_i \leq r_j$, (C.2) $\min\{p_{i1}, p_{j2}\} \leq \min\{p_{i2}, p_{j1}\}$, (C.10) $C_2(\sigma) \leq r_i+p_{i1}$ *and* (C.11) $p_{i2} \leq p_{j1}+w_j$, *there is an optimal solution among schedules that start with* σij (that is, σij *dominates* σji).

Proposition 5 : *There is a partial schedule* σ, *and if we have two jobs i and j* ($i \not\in \sigma$, $j \not\in \sigma$) *satisfying the five conditions such as* (C.8) $C_1(\sigma) \le r_i \le C_2(\sigma)$, (C.9) $r_i \le r_j$, (C.2) $\min\{p_{i1}, p_{j2}\} \le \min\{p_{i2}, p_{j1}\}$, (C.12) $C_2(\sigma) \le r_i + p_{i1} + w_j$ *and* (C.13) $C_2(\sigma) + p_{i2} - r_i - p_{i1} \le p_{j1} + w_j$, *there is an optimal solution among schedules that start with* σij (that is, σij *dominates* σji).

Proposition 6 : *If* $p_{j*1} \le p_{j*2} \le \min_{i \ne j*}\{p_{i1}\}$, *where* $j*$ is the job with the minimum processing time on machine 1 ($j* = \arg\min_i\{p_{i1}\}$) and the minimum ready time ($j* = \arg\min_i\{r_i\}$) among all jobs, *we have an optimal schedule with job* $j*$ *positioned in the first order.*

Proof : First, let's assume that we have an optimal schedule without job $j*$ positioned in the first order. Then, we can express the optimal schedule as $\sigma_1 j* \sigma_2$, where σ_1 and σ_2 are partial schedules, and σ_1 is non-empty partial sequence. We can prove this proposition by showing $C_k(j*\sigma_1\sigma_2) \le C_k(\sigma_1 j*\sigma_2)$ for any arbitrary σ_1 and σ_2 for $k = 1, 2$. The completion times of job $j*$ (the last job) in schedule $\sigma_1 j*$ on machine 1 and 2 can be expressed as the following equations.

Since $r_{j*} \le C_1(\sigma_1)$ from the condition of $j* = \arg\min_i\{r_i\}$,

$$\begin{aligned} C_1(\sigma_1 j*) &= \max\{\max(r_{j*}, C_1(\sigma_1)) + p_{j*1}, \\ &\qquad C_2(\sigma_1) - w_{j*}\} \\ &= \max\{C_1(\sigma_1) + p_{j*1}, C_2(\sigma_1) - w_{j*}\} \end{aligned} \tag{A.18}$$

and

$$C_2(\sigma_1 j*) = \max\{C_1(\sigma_1 j*), C_2(\sigma_1)\} + p_{j*2} \tag{A.19}$$

On the other hand, since the condition of $p_{j*2} \le \min_{i \ne j*}\{p_{i1}\}$, the second job in $j*\sigma_1$ (that is, first job in σ_1 of $j*\sigma_1$) can be started on machine 2 instantly after its first operation (on machine 1) is completed. Therefore, the completion times of jobs in σ_1 of $j*\sigma_1$ on each machine can be delayed at most to p_{j*1} than those in σ_1 because of the condition of $j* = \arg\min_i\{r_i\}$. That is, we have

$$C_1(j*\sigma_1) \le p_{j*1} + C_1(\sigma_1) \tag{A.20}$$

and

$$C_2(j*\sigma_1) \le C_2(\sigma_1) + p_{j*1} \tag{A.21}$$

From (A.18) and (A.20), we have

$$\begin{aligned} &C_1(j*\sigma_1) \le C_1(\sigma_1 j*) \text{ since } p_{j*1} + C_1(\sigma_1) \\ &\le \max\{C_1(\sigma_1) + p_{j*1}, C_2(\sigma_1) - w_{j*}\} \end{aligned} \tag{A.22}$$

In addition, From (A.19), (A.21) and the condition of $p_{j*1} \le p_{j*2}$, we have

$$\begin{aligned} &C_2(j*\sigma_1) \le C_2(\sigma_1 j*) \text{ since } C_2(\sigma_1) + p_{j*1} \\ &\le \max\{C_1(\sigma_1 j*), C_2(\sigma_1)\} + p_{j*2} \end{aligned} \tag{A.23}$$

From (A.22) and (A.23), $C_k(j*\sigma_1\sigma_2) \le C_k(\sigma_1 j*\sigma_2)$ for $k = 1, 2$. This proof is completed. ■

From proposition 6, the following Corollary holds :

Corollary 1 : *Given a partial schedule* σ, *If* $p_{j*1} \le p_{j*2} \le \min_{i \ne j*}\{p_{i1}\}$, *where* $j* = \arg\min_i\{p_{i1}\}$, $j* = \arg\min_i\{r_i\}$, $i \not\in \sigma$, $j \not\in \sigma$, *then schedules that start with* $\sigma j*$ *dominate the others.*

Proposition 7 : *If* $p_{j*2} \le p_{j*1} \le \min_{i \ne j*}\{p_{i2}\}$, *where* $j*$ = arg $\min_i\{p_{i2}\}$ and $j*$ = arg $\max_i\{r_i\}$, *we have an optimal schedule with job* $j*$ *positioned in the last order.*

Proof : First, let's assume that we have an optimal schedule without job $j*$ positioned in the last order. Then, we can express the optimal schedule as $\sigma_1 j*\sigma_2$, where σ_1 and σ_2 are partial schedules, and σ_1 is non-empty partial sequence. We can prove this proposition by showing $C_2(\sigma_1\sigma_2 j*) \le C_2(\sigma_1 j*\sigma_2)$ for any arbitrary σ_1 and σ_2.

From $C_1(\sigma_1) \le C_1(\sigma_1 j*)$ and $C_2(\sigma_1) \le C_2(\sigma_1 j*)$, we know $C_k(\sigma_1\sigma_2) \le C_k(\sigma_1 j*\sigma_2)$ for k = 1, 2.
In addition, the completion times of job $j*$ (the last job) in schedule $\sigma_1\sigma_2 j*$ on the machine 1 and 2 can be expressed as the following equations.

$$C_1(\sigma_1\sigma_2 j*) = \max\{\max(r_{j*}, C_1(\sigma_1\sigma_2)) + p_{j*1}, C_2(\sigma_1\sigma_2) - w_{j*}\} \quad \text{(A.24)}$$

and

$$C_2(\sigma_1\sigma_2 j*) = \max\{C_1(\sigma_1\sigma_2 j*), C_2(\sigma_1\sigma_2)\} + p_{j*2} \quad \text{(A.25)}$$

From (A.24) and (A.25), we have

$$\begin{aligned} C_2(\sigma_1\sigma_2 j*) &= \max\{r_{j*}+p_{j*1}, C_1(\sigma_1\sigma_2)+p_{j*1}, \\ &\quad C_2(\sigma_1\sigma_2)-w_{j*}, \\ &\quad C_2(\sigma_1\sigma_2)\}+p_{j*2} \\ &= \max\{r_{j*}+p_{j*1}, C_1(\sigma_1\sigma_2)+p_{j*1}, \\ &\quad C_2(\sigma_1\sigma_2)\}+p_{j*2} \end{aligned}$$

Here, from the condition of $p_{j*1} \le \min_{i \ne j*}\{p_{i2}\}$, we have $C_1(\sigma_1\sigma_2)+p_{j*1} \le C_2(\sigma_1\sigma_2)$.
Therefore, $C_2(\sigma_1\sigma_2 j*) = \max\{r_{j*}+p_{j*1}, C_2(\sigma_1\sigma_2)\}+p_{j*2}$

Case 1 : If $r_{j*}+p_{j*1} \le C_1(\sigma_1\sigma_2)$, $C_2(\sigma_1\sigma_2 j*) = C_2(\sigma_1\sigma_2)+p_{j*2}$
Case 2 : Otherwise($C_1(\sigma_1\sigma_2) < r_{j*}+p_{j*1}$), $C_2(\sigma_1\sigma_2 j*) = r_{j*}+p_{j*1}+p_{j*2}$

In the case 1, the completion times of jobs in σ_2 of $\sigma_1 j*\sigma_2$ on the second machine can be delayed at least to p_{j*2} than those in σ_2 of $\sigma_1\sigma_2 j*$ because of the condition of $j*$ = arg $\max_i\{r_i\}$ and $p_{j*2} \le p_{j*1}$. That is, we have

$$C_2(\sigma_1\sigma_2 j*) = C_2(\sigma_1\sigma_2)+p_{j*2} \le C_2(\sigma_1 j*\sigma_2) \quad \text{(A.26)}$$

On the other hand, in the case 2, the completion times of jobs in σ_2 of $\sigma_1 j*\sigma_2$ on the first machine can be started at the time of $r_{j*}+p_{j*1}$. Therefore, jobs in σ_2 of $\sigma_1 j*\sigma_2$ can be completed later than the time of $r_{j*}+p_{j*1}+p_{j*2}$, obviously. That is, we have

$$C_2(\sigma_1\sigma_2 j*) = r_{j*}+p_{j*1}+p_{j*2} < C_2(\sigma_1 j*\sigma_2) \quad \text{(A.27)}$$

From (A.26) and (A.27), we have $C_2(\sigma_1\sigma_2 j*) \le C_2(\sigma_1 j*\sigma_2)$. This proof is completed. ■

4. Lower Bounds

In this paper, we present several lower bounding schemes on the makespan of a partial schedule to be used in the suggested branch and bound algorithms. The following two lower bounds are developed based on a machine-based lower bound showed in Baker [1974], and in those, in addition, limited waiting time and ready times are considered.

The first is $LB_1 = \max\left[\min_{j\in U}[\max\{C_1(\sigma), r_j\} + p_{j1}] C_2(\sigma)\right] + \sum_{j\in U} p_{j2}$.

The front formula means the minimum time when the operations of unscheduled jobs can be started on the second machine, and the back formula means the total processing times (the sum of processing times) of unscheduled jobs on the second machine.

The second is $LB_2 = \max\left[C_1(\sigma), \min_{j\in U}[\max\{C_2(\sigma) - w_j - p_{j1}, r_j\}]\right] + \sum_{j\in U} p_{j1} + \min_{j\in U}\{p_{j2}\}$.

The front formula means the minimum time when the operations of unscheduled jobs can be started on the first machine, and the back formula means the total processing time of unscheduled jobs on machine 1. In this paper, we describe this lower bound, max(LB_1, LB_2) as LB_M.

We use another lower bound based on the following proposition 8 suggested by Choi and Kim [2007] for the two-machine permutation flowshop. In the proposition 8, ***σπ*** is (partial) schedule obtained with σ followed by partial schedule ***π***.

Proposition 8 [Choi and Kim 2007]. *Let* σ *be a partial schedule to be placed at the front of a complete schedule for the two-machine permutation flowshop scheduling problem with the objective of minimizing makespan. Then the makespan of any complete schedule starting with* σ *cannot be less than* $C_2(\sigma\pi*)$, *where* π* *is a partial schedule obtained by applying the Johnson's rule* (Johnson 1954) *to the set of jobs that are included in U.*

Here, a lower bound can be obtained by using the proposition 8 and assuming that there are no both limited waiting time constraints and ready times of jobs in *U*, and this lower bound is described as $LB_{C\&K}$ in this paper.

5. Branch and Bound Algorithm

The main objective of the research is presenting efficient branch and bound algorithms for the considered scheduling problem in this paper. The architecture of this branch and bound algorithm is based on the typical branch and bound algorithm of Baker [1974]. That is, in the suggested branch and bound algorithms, a branch and bound tree is constructed to generate all possible schedules (sequences), that is, a node in the branch and bound tree means its corresponding partial sequence (schedule), and a node at the *k*th depth in the branch and bound tree means its corresponding partial schedule for the first-positioned *k* jobs in a complete schedule [Choi and Kim 2007].

In these branch and bound algorithms, for calculating initial upper bounds in the branch and bound algorithms, we use several existing heuristic algorithms, that is, modified Johnson's algorithm (which is develop algorithm based on the Johnson's algorithm [Johnson, 1954] for twp-machine permutation flowshop scheduling problem with the objective of minimizing makespan), modified NEH algorithm (which is developed algorithm based on the algorithm of Nawaz, Enscore and Ham [1983] for m-machine permutation flowshop scheduling problems and called as *MNEH*) and Greed Local Search algorithm (which is developed algorithm based on an extended neighborhood search algorithm of Kim [1993] for m-machine permutation flowshop scheduling problem with the objective of minimizing mean tardiness and called as *GLS*) suggested by Choi [2015]. Choi [2015] suggested these heuristic algorithm for the two-machine permutation flowshop scheduling problem with limited waiting times and ready times, which is same with the scheduling problem of this research. We use the best one among solutions of the above heuristic algorithms as the initial upper bounds in the branch and bound algorithms.

In addition, when we generate some child nodes from a selected parent node, based on the developed dominance properties in the section 3, we prune the child nodes dominated by others. Also, for the nodes that are not pruned by the dominance properties, we compute the presented two bounds (LB_{M} and $LB_{\mathrm{C\&K}}$), and we fathom the corresponding nodes without lower bounds less than the current upper bound. Finally, when we construct the branch and bound tree, the depth and first rule is adopted, in which we select a node with the biggest number of jobs among the considered nodes (partial schedules) for the next branching.

6. Computational Tests

To evaluate efficiencies of the developed dominance properties and lower bounding schemes and the suggested branch and bound algorithms, we performed the experiments on sets of randomly generated test problems. The tests were done on a personal computer with the CPU of 2.6 GHz clock speed, and the algorithms were coded by using C++ programming language.

For the computational tests, the processing times of jobs were generated from $DU(1, 50)$, where $DU(a, b)$ denotes the discrete uniform distribution with a range of $[a, b]$, and limited waiting times of the jobs were generated from $DU(1, 100)$. Also, ready times of the jobs were generated from $DU(0, LB_{\mathrm{C\&K}})$, in which $LB_{\mathrm{C\&K}}$ is described in the section 4, and here, when $LB_{\mathrm{C\&K}}$ is calculated for generating ready times, we assume that σ is empty sequence ($\sigma = \phi$). In addition, the utilization of a bottleneck workstation (process section) is expected to be close to 100%, and note that there are always job to be processed at the workstation (which is one of bottleneck process sections in general) if the ready times are generated by the above way. Here, note that the ratios of processing times, limited waiting times and ready times of jobs reflect the real situation at the two-machine flowshop consisting of continuous clean and diffusion processes in a real fab. Computation time required for branch and bound algorithms (we tested several versions of the branch and bound algorithm) was limited to 3600 seconds (1 hour) of CPU time for each test problem to avoid excessive computation times required for the tests. Therefore, in some instances, optimal solutions could not be found within the CPU time limit of 1 hour.

For evaluations of branch and bound algorithms, 100 problems were randomly generated, 20 problems for each of four levels (10, 20, 30, 40 and 50) for the number of jobs. In tables 3-6, for an instance, the CPU time for a branch and bound algorithm means the duration (in seconds counter) to find optimal solution, and if a branch and bound algorithm cannot find the optimal solution within 3,600 seconds, its duration (CPU time) is 3,600 seconds. For a branch and bound algorithm, as ACPUT† and MCPUT‡ are increased, it means that performance of the branch and bound algorithm gets worse. In addition, for an instance, the ratio of CPU time for a branch and bound algorithm means its relative ratio of CPU time in comparison with that of another B&B algorithm (BB3), and the average ratio of CPU time for a branch and bound algorithm is mean value of the ratio values of CPU times for 20 instances for each of four levels (10, 20, 30, 40 and 50)for the number of jobs. For example, for an instance, the ratio of CPU time of "BB1/BB3" is the relative ratio of CPU time of BB1 in comparison with that of BB3 (CPU time of BB1/CPU time of BB3). That is, as ARCPUT* of "BB1/BB3" is increased, it means the performance of BB1 gets worse in comparison with BB3.

Firstly, we compared three branch and bound algorithms (BB1, BB2 and BB3) with different lower bounds. For fathoming the branch and bound trees, BB1, BB2 and BB3 use LB_{M}, $LB_{\mathrm{C\&K}}$ and max(LB_{M}, $LB_{\mathrm{C\&K}}$) as lower bounding schemes, respectively. In BB1, BB2 and BB3, we use all the suggested dominance properties (propositions 1~7) for reducing the number of generated nodes, and we use the best one among solutions of the three heuristic algorithms [Choi, 2015] as initial upper bounds. As can be seen from Table 2, BB3 worked better than BB1 and BB2 in terms

<Table 2> Effectiveness of the Lower Bounds

n	ACPUT† (MCPUT‡)			ARCPUT*	
	BB1	BB2	BB3	BB1/BB3	BB2/BB3
10	0.01 (0.01)	0.01 (0.01)	0.01 (0.01)	1.00	1.00
20	184.45 (0.07)	183.80 (0.05)	183.80 (0.05)	2.05	1.00
30	1329.44 (77.86)	1253.62 (63.52)	1158.08 (49.68)	3.30	1.48
40	1694.15 (474.51)	1347.74 (118.50)	1309.69 (76.32)	278.93	1.28
50	1854.48 (2072.93)	(1786.80 (1621.37)	1676.55 (547.87)	215.26	1.54

† average CPU time or lower bound on the average CPU time, which was computed assuming that the CPU time for an instance that has not been solved to optimality within 3600 seconds is 3600 seconds [Joo and Kim, 2009].
‡ median CPU time.
* average ratio of CPU times of a branch and bound algorithm in comparison with those of BB3.

of both average CPU time and median CPU time. In addition, as seen in term of average ratio of CPU time, there are less significant differences between BB2 and BB3 than those between BB1 and BB3, and it means that $LB_{C\&K}$ outperforms LB_{M}.

Next, to show the effectiveness of the dominance properties, we solve the given instances with two branch and bound algorithms, BB3 with all dominance properties and BB4 (BB3-DPs), in which none of the dominance properties is used. The results of the test are given in <Table 3>. When the suggested dominance properties were not used, CPU time was increased significantly.

Thirdly, we test the effectiveness of the initial upper bound from the heuristic algorithms proposed by Choi [2015]. For this test, we solve the given problems with two branch and bound algorithms, BB3, in which the heuristic algorithms of Choi [2015] are used for initial upper bound

<Table 3> Effectiveness of the Dominance Properties

n	ACPUT† (MCPUT‡)		ARCPUT*
	BB3	BB4	BB4/BB3
10	0.01 0.01	0.01 0.01	1.000
20	183.80 0.05	200.53 0.05	1.564
30	1158.08 49.68	1289.98 16.07	1.402
40	1309.69 76.32	1350.04 96.60	1.329
50	1676.55 547.87	1735.26 1130.00	1.018

†, ‡, * See the footnotes of <Table 4>.

<Table 4> Effectiveness of the initial upper bound

n	ACPUT† (MCPUT‡)		ARCPUT*
	BB3	BB4	BB5/BB3
10	0.01 0.01	0.01 0.01	1
20	183.80 0.05	374.72 1.24	785.1
30	1158.08 49.68	1375.44 180.94	674.7
40	1309.69 76.32	2460.27 3600.01	6647.4
50	1676.55	2409.15	5446.6

†, ‡, * See the footnotes of <Table 4>.

<Table 5> Summary of the Tests on the Branch and Bound Algorithms

n	ACPUT† (MCPUT‡)					NINS#				
	BB1	BB2	BB3	BB4	BB5	BB1	BB2	BB3	BB4	BB5
10	0.01† (0.01)‡	0.01 (0.01)	0.01 (0.01)	0.01 (0.01)	0.01 (0.01)	0	0	0	0	0
20	184.45 (0.07)	183.80 (0.05)	183.80 (0.05)	200.53 (0.05)	374.72 (1.24)	1	1	1	1	2
30	1329.4 (77.8)	1253.6 (63.5)	1158.0 (49.6)	1289.9 (16.1)	1375.4 (180.9)	7	6	6	6	7
40	1694.1 (474.5)	1347.7 (118.5)	1309.6 (76.32)	1350.0 (96.60)	2460.2 (3600)	9	7	7	7	13
50	1854.4 (2072)	1786.8 (1621)	1676.5 (547.8)	1735.2 (1130)	2409.1 (3600)	10	9	9	9	13

†, ‡, * See the footnotes of <Table 4>.
# number of problems among 20 problems that optimal solution cannot be obtained within 3600s.

<Table 6> Analysis of Variance (for CPU times)

Source of variation	Degrees of freedom	Sum of squared error	Mean squared error	F**
Algorithm (A)	4	13746175.9	3436544.0	2.6**
Number of jobs (N)	4	287521494.1	71880373.5	55.3**
A×N	16	314726374.2	19670398.4	15.1**
Error	475	617277227.8	1299531.0	
Total	499	1233271272		

**Different effects with the significance level of 0.05.

as stated above, and BB5 (BB3-initial upper bound), in which none of initial upper bounds is used, that is, a random solution is generated for initial upper bound in each instance. The results of the test are given in <Table 4>. When the initial upper bound were not used, CPU time was increased very much since BB5 cannot give optimal solutions in many instances as seen in <Table 6>.

In summary, we shows the performances of five branch and bound algorithms (BB1-BB5) in <Table 5>. In addition, in the table, we add the number of problems among 20 problems that optimal solution cannot be obtained within 3600s. As you see in <Table 6>, the lower bounding schemes, the dominance properties (proposed in this research) and initial upper bounds are effective, and when the branch and bound algorithm does not use the dominance properties, lower bounding schemes and initial upper bounds, the CPU times are increased significantly.

Finally, to see the differences in the performance of the algorithms, we performed an ANOVA (analysis of variance) for CPU times of five branch and bound algorithms (BB1-BB5). Results are given in <Table 6>. Note that performances of algorithms (BB1-BB5) are statistically different at the significance level of 0.05. In addition, CPU times of five branch and bound algorithms are affected very much by problem size (number of jobs), and it is very natural situation since the CPU times of algorithms are increased as the problem size becomes large.

7. Conclusion

In this research, we suggest branch and bound algorithms for a two-machine permutation flowshop scheduling problem with the objective of minimizing makespan. In this permutation flowshop, after each job is operated on the machine 1 (first machine), the job has to start its second operation on machine 2 (second machine) within its corresponding limited waiting time. In addition, each job has its corresponding ready time at the machine 1. For this scheduling problem, we develop various dominance properties and three lower bounding schemes, which are used for the suggested branch and bound algorithm. In the results of computational tests, we can show that the branch and bound algorithm is efficient and can give optimal solutions (for problem size of 30～50 jobs) in a reasonable amount of CPU times.

We can extended this research in several directions. For example, we can modify the suggested lower bounding schemes and dominance properties for the m-machine permutation flowshop problem with limited waiting times and ready times. In addition, more general cases, that is, parallel machines in each stage of m-machine flowshop with limited waiting times and ready times, can be considered.

References

[1] Attar, S. F., Mohammadi, M., and Tavakkoli-Moghaddam, R., "Hybrid flexible flowshop scheduling problem with unrelated parallel machines and limited waiting times", *International Journal of Advanced Manufacturing technology*, Vol. 68, 2013, pp. 1583-1599.

[2] Baker, K. R., Introduction to sequencing and scheduling, New York : John Wiley and Sons, 1974.

[3] Bouquard, J. L. and Lent´e, C., "Two-machine flow shop scheduling problems with minimal and maximal delays", *A Quarterly Journal of Operations Research*, Vol. 4, 2006, pp. 15-28.

[4] Chen, T. C. E., Gupta, J. N. D., and Wang, G., "A review of flowshop scheduling research with setup times", *Production and Operations Management*, Vol. 9, 2000, pp. 283-302.

[5] Choi, S. W., "Heuristics for two-machine re-entrant flowshop with limited waiting times and ready times", Submitted to Korea Contents Spring Conference, May, 2015.

[6] Choi, S. W. and Kim, Y. D., "Minimizing makespan on a two-machine re-entrant flowshop", *Journal of the Operational Research Society*, Vol. 58, 2007, pp. 972-981.

[7] Choi, S. W., Lim, T. K., and Kim, Y. D., "Heuristics for scheduling wafer lots at the deposition workstation in a fab", *Journal of the Korean Institute of Industrial Engineers*, Vol. 36, 2010, pp. 125-137.

[8] Chu, C., "A branch and bound algorithm to minimize total tardiness with different release times", *Naval Research Logistics*, Vol. 39, 1992, pp. 265-283.

[9] Framinan, J. M., Gupta, J. N. D., and Leisten, R., "A review and Classification of heuristics for permutation flow-shop scheduling with makespan objective", *Journal of Operational Research Society*, Vol. 55, 2004, pp. 1243-1255.

[10] Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G., "Optimization and approximation in deterministic sequencing and scheduling : A survey", *Annals of Discrete Mathematics*, Vol. 5, 1979, pp. 287-326.

[11] Gupta, J. N. D. and Stafford, J. E. F., "Flowshop scheduling research after five decades", *European Journal of Operational Research*, Vol. 169, 2006, pp. 699-711.

[12] Hall, L. A., "A polynomial approximation scheme for a constrained flow shop scheduling problem", *Mathematics of Operations research*, Vol. 19, 1994, pp. 68-85.

[13] Johnson, S. M., "Optimal two- and three-stage production schedules with setup times included", *Naval Research Logistics Quarterly*, Vol. 1, 1954, pp. 61-68.

[14] Joo, B. J. and Kim, Y. D., "A branch-and-bound algorithm for a two-machine flow-

shop scheduling problem with limited waiting time constraints", *Journal of the Operational Research Society*, Vol. 60, 2009, pp. 572-582.

[15] Kim, Y. D., "Heuristics for flowshop scheduling problems minimizing mean tardiness", *Journal of Operational Research Society*, Vol. 44, 1993, pp. 19-28.

[16] Nawaz, M., Enscore, E. E., and Ham, I., "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem", *Omega*, Vol. 11, 1983, pp. 91-95.

[17] Potts, C. N., "Analysis of heuristics for two-machine flow-shop sequencing subject to release dates", *Mathematics of Operations research*, Vol. 10, 1985, pp. 576-584.

[18] Tadei, R., Gupta, J. N. D., Della, C. F., and Cortesi, M., "Minimizing makespan in the two-machine flow-shop with release times", *Journal of Operational Research Society*, Vol. 49, 1998, pp. 77-85.

[19] Yang, D. L. and Chern, M. S., "A two-machine flowshop sequencing problem with limited waiting time constraints", *Computers and Industrial Engineering*, Vol. 28, 1995, pp. 63-70.

Author Profile

Seong-Woo Choi

Seong-Woo Choi is an assistant professor at the Department of Business and Administration, Kyonggi University. He received the B.S., M.S. and Ph.D. degrees in Industrial Engineering from KAIST. His research areas include design and operation of manufacturing systems and operations scheduling. Before joining the faculty at Kyonggi University, he had worked for Samsung Electronics Company as a senior engineer at the systems technology group in the semiconductor business division and for Hoseo University.