International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 15 (2017) pp. 5101-5109
© Research India Publications. http://www.ripublication.com

Optimal Scheduling Algorithms to Minimize Total Flowtime on a Two-Machine Permutation Flowshop with Limited Waiting Times and Ready Times of Jobs

Seong-Woo Choi [1]

Dept. Of Business Administration, Kyonggi University, Suwon-si, 443-760, South Korea. ,

[1]*Orcid: 0000-0002-7946-2990*

Abstract

In this research, we consider a two-machine permutation flowshop with ready times and limited waiting times of jobs in a semiconductor fab. In the flowshop, each job has its corresponding ready time (arrival time) at machine 1 (for the first operation) and limited waiting time between the completion time of the first operation on machine 1 and the starting time of the second operation on machine 2. That is, the first operation of each job cannot be started earlier than its ready time on the machine 1, and the second operation of each job has to be started within its corresponding limited waiting time after its first operation is completed on machine 1. In this scheduling problem, the objective is minimizing the total flowtime of jobs to be processed on the two-machine permutation flowshop, and we suggested branch and bound algorithms to give optimal solutions. To boost efficiency of the suggested branch and bound algorithms, we developed several dominance properties and three lower bounding schemes, and we used these in the branch and bound algorithms. In addition, we used the solutions of existing heuristic algorithm as the initial upper bounds of the branch and bound algorithms. To evaluate the performances of the branch and bounds, we executed computational simulation test and showed the results.

Keywords: limited waiting time, ready time, total flowtime, branch and bound algorithm

INTRODUCTION

In this research, we consider a two-machine permutation flowshop scheduling problem with ready times and limited waiting times of jobs in a semiconductor fab, and the objective function is minimizing the total flowtime of jobs to be processed. In this scheduling problem, the first operation of each job cannot be started earlier than its ready time on the machine 1 and each job has to start its second operation on machine 2 within its corresponding limited waiting time after its first operation on machine 1 is completed.

This scheduling problem can be denoted as $F2/r_i$, max-wait/$\sum(C_i-r_i)$, which is based on the three-field notation of Graham et al. [1] in which r_i and max-wait mean that jobs cannot be processed on the first machine earlier than their arrival times and have to be processed on the second machine within a certain period of time after those jobs are completed on the first machine, respectively, and $\sum(C_i-r_i)$ is the total flowtime of jobs, and note that the typical two-machine flowshop is a special case of the scheduling problem of this research [2, 3, 4].

Such a two-machine permutation flowshop can be found in a semiconductor manufacturing line (fab). For example, at the workstation with clean and diffusion operations in a semiconductor fabrication line, after a chemical treatment (clean) operation for a wafer lot is completed on the clean machine (machine 1), the next operation (diffusion) for the wafer lot must be started on the diffusion machine (machine 2) within its pre-determined time period called limited waiting time (the length of limited waiting times may be different according to the recipes), and if the next operation 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 [2, 4, 5]. On the other hand, since the workstation above may not be the first operation of semiconductor fabs in general, jobs arrive at this workstation dynamically, that is, jobs have different positive arrival times at the machine 1 (for the clean operation) of this operation section (clean-diffusion) [3, 6]. Many researchers have studied the typical flowshop scheduling problems [7, 8 and 9]. In addition, as the following researches, there are some studies for the flowshop scheduling problem with limited waiting times and ready times of jobs, respectively. Tadei et al. [10] and Potts [11] 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, and in the researches of Hall [12] and Chu [13], a polynomial approximation scheme and a branch and bound algorithm were suggested for minimizing makespan and total tardiness on the two-machine flow-shop with release date, respectively [2, 4]. Choi [2, 3] suggested branch and bound algorithms and heuristic algorithms for the scheduling problem with ready times and limited waiting times of jobs, and the objective was minimizing makespan. In addition, Choi [4] suggested various heuristic algorithms for the same scheduling problem of this research. However, there is no research which suggested branch and bound algorithms, dominance properties and lower bounding schemes for the scheduling problem on the two-machine flowshop with ready time, limited waiting time and objective function of minimizing total flowtime.

PROBLEM DESCRIPTION

In this paper, the notations below are used, and these were used in the research of Choi [2, 3, 4].

i	indices of jobs, i = 1, 2, ..., n
k	indices of machines, k = 1, 2
p_{ik}	processing 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
$\sigma ij...m$	partial schedule obtained with σ followed by jobs i, j, ...,m in this order
[i]	job that is completed i-th in a (partial) schedule
c_{ik}	completion time of job that is completed i-th 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
C_i-r_i	flowtime of job i
$\sum(C_i-r_i)$	total flowtime of all jobs (objective value of this scheduling problem)

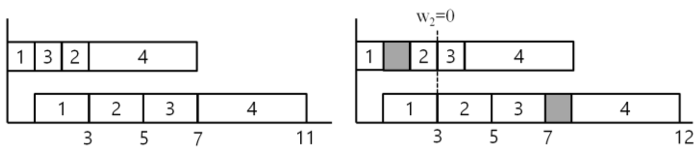
In this paper, only permutation schedules (sequences) are considered, in which operation sequences of jobs are the same on the two machines (machine 1 and machine 2), that is, the operation orders of jobs are not differenct on machine 1 and 2. Although permutation schedules are dominant in the ordinary two-machine flowshop scheduling problem with the objective of minimizing makespan [4, 14, 15], it is not the case with the limited waiting time, the ready times and the objective function of minimizing total flowtime, that is, there may be cases that a non-permutation schedule (sequence) can be better than the best permutation schedule (sequence). The example below is such a case.

Example. (the case that permutation schedules are not dominant)

There are four jobs to be scheduled, and the processing, limited waiting and ready times of jobs are given in Tables 1. In the figure 1, we can confirm that the best permutation schedule (b) can be obtained by the operation sequence (1, 2, 3, 4) of jobs, while there is a better non-permutation schedule, in which the operation sequence of jobs is (1, 3, 2, 4) on machine 1 and the operation sequence of jobs is (1, 2, 3, 4) on machine 2.

Table 1: Processing, limited waiting and ready times of jobs in the example

		i = 1		i = 2		i = 3		i = 4
p_{i1}		1		1		1		4
p_{i2}		2		2		2		4
w_i		10		0		10		10
r_i		0		2		1		3



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

Figure 1: An example that best permutation schedule are dominated by non-permutation schedule

However, in the scheduling problem of this research, only permutation schedules are considered since permutation schedules are preferred to non-permutation schedules in most real systems because of the the ease of material flow management implementation, and non-permutation schedules are not feasible in many cases because of technical constraints of material handling systems (4, 14, 15).

In this two-machine permutation flowshop scheduling problem, there are n jobs to be processed in the machine 1 and then machine 2, and we use the assumptions of Choi [3] as follows.

1) In the two-machine flowshop scheduling problem, we have a given set of jobs with different arrival times, that is, each job can be operated on the machine 1 at its arrival time.

2) The operation times and limited queue 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 queue time after the job is operated on the machine 1.

With the notations above, Choi [2, 3] expressed the completion time of each job in a complete or partial schedules as follows, and we use theseequations in this research [4].

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

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

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

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

Dominance Properties

In this section, seven dominance properties are suggested, and they can be used to eliminate some partial schedules (sequences) that are dominated by other partial schedules. By using the suggested dominance properties, the number of sub-problems to be considered can be reduced in the suggested branch and bound algorithms in this research.

The seven dominance properties can be proved by using the Johnson's algorithm [15] (in which the Johnson's rule is used for sequencing jobs, for two-machine permutation flowshop scheduling problems, and according to the Johnson's rule, job i precedes job j in an optimal sequence if $\min\{p_{i1}, p_{j2}\} \leq \min\{p_{i2}, p_{j1}\}$.), dominance properties developed by Tadei et al. [16] (for a two-machine flowshop scheduling problem with ready times) and dominance properties (for minimization of makespan on a two-machine flowshop with limited waiting time and ready time) developed by Choi [3].

Proposition 1. There is a given partial schedule σ, if we have two jobs i ($i \notin \sigma$) and j ($j \notin \sigma$) satisfying the following four conditions: (C.1) $C_2(\sigma) \leq r_i \leq r_j$, (C.2), $\min\{p_{i1}, p_{j2}\} \leq \min\{p_{i2}, p_{j1}\}$, (C.3), $p_{i2} \leq p_{j1} + w_j$ and (C.4) $p_{i1} + p_{i2} \leq p_{j1} + p_{j2}$, then partial schedule σji is dominated by σij.

Proof. If we can show that (A.1) $C_k(\sigma ij) \leq C_k(\sigma ji)$ and (A.2) $F(\sigma ij) \leq F(\sigma ji)$ for k = 1, 2, the proof of this proposition is completed.

Here, for k = 1, 2, (A.1) $C_k(\sigma ij) \leq C_k(\sigma ji)$ was already proved under the conditions (C1)-(C3) in the research of Choi [3]. Therefore, we need to show only (A.2) $F(\sigma ij) \leq F(\sigma ji)$ for k = 1, 2.

We have the following equalities related to the completion times of jobs i and j in schedules σij and σji from the equalities (E.1)-(E.4) at the end of the previous section [3].

$$C_1(\sigma i) = \max\{\max(r_i, C_1(\sigma)) + p_{i1}, C_2(\sigma) - w_i\} \quad (e.1)$$

$$C_2(\sigma i) = \max\{C_1(\sigma i), C_2(\sigma)\} + p_{i2} \quad (e.2)$$

$$C_1(\sigma ij) = \max\{\max(r_j, C_1(\sigma i)) + p_{j1}, C_2(\sigma i) - w_j\} \quad (e.3)$$

$$C_2(\sigma ij) = \max\{C_1(\sigma ij), C_2(\sigma i)\} + p_{j2} \quad (e.4)$$

$$C_1(\sigma j) = \max\{\max(r_j, C_1(\sigma)) + p_{j1}, C_2(\sigma) - w_j\} \quad (e.5)$$

$$C_2(\sigma j) = \max\{C_1(\sigma j), C_2(\sigma)\} + p_{j2} \quad (e.6)$$

$$C_1(\sigma ji) = \max\{\max(r_i, C_1(\sigma j)) + p_{i1}, C_2(\sigma j) - w_i\} \quad (e.7)$$

$$C_2(\sigma ji) = \max\{C_1(\sigma ji), C_2(\sigma j)\} + p_{i2} \quad (e.8)$$

To show that (A.2) $F(\sigma ij) \leq F(\sigma ji)$ for k = 1, 2, that is, $F(\sigma ij) = C_2(\sigma i) - r_i + C_2(\sigma ij) - r_j \leq F(\sigma ji) = C_2(\sigma j) - r_j + C_2(\sigma ji) - r_i$, we need to show only $C_2(\sigma i) \leq C_2(\sigma j)$ since (A.2) $F(\sigma ij) \leq F(\sigma ji)$ is reduced to $C_2(\sigma i) + C_2(\sigma ij) \leq C_2(\sigma j) + C_2(\sigma ji)$ and $C_2(\sigma ij) \leq C_2(\sigma ji)$ was already satisfied by (A.1) [3].

Here, from the condition (C.1) $C_2(\sigma) \leq r_i \leq r_j$ of this proposition, (e.2) and (e.6) are reduced to $C_2(\sigma i) = r_i + p_{i1} + p_{i2}$ and $C_2(\sigma j) = r_j + p_{j1} + p_{j2}$, recpectively. In addition, from the condition (C.4) $p_{i1} + p_{i2} \leq p_{j1} + p_{j2}$, $C_2(\sigma i) \leq C_2(\sigma j)$ is satisfied.

In conclusion, we have (A.1) $C_k(\sigma ij) \leq C_k(\sigma ji)$ and (A.2) $F(\sigma ij) \leq F(\sigma ji)$ for k = 1, 2. This completes the proof. ■

Proposition 2. There is a given partial schedule σ, if we have two jobs i ($i \notin \sigma$) and j ($j \notin \sigma$) satisfying the following four conditions: (C.5) $r_i \leq C_1(\sigma)$, (C.2) $\min\{p_{i1}, p_{j2}\} \leq \min\{p_{i2}, p_{j1}\}$, (C.6) $C_2(\sigma) - C_1(\sigma) \leq p_{i1}$, (C.3) $p_{i2} \leq p_{j1} + w_j$ and (C.4) $p_{i1} + p_{i2} \leq p_{j1} + p_{j2}$, then partial schedule σji is dominated by σij.

Proof. The proof procedure of this proposition 1 is similar to that of proposition 1. That is, If we can show that (A.1) $C_k(\sigma ij) \leq C_k(\sigma ji)$ and (A.2) $F(\sigma ij) \leq F(\sigma ji)$ for k = 1, 2, the proof of this proposition is completed.

Here, for k = 1, 2, (A.1) $C_k(\sigma ij) \leq C_k(\sigma ji)$ was already proved under the conditions (C2), (C3), (C5) and (C6) in the research of Choi [3]. Therefore, we need to show only (A.2) $F(\sigma ij) \leq F(\sigma ji)$ for k = 1, 2.

To show that (A.2) $F(\sigma ij) \leq F(\sigma ji)$ for k = 1, 2, we need to show $C_2(\sigma i) \leq C_2(\sigma j)$ (refer to the proof procedure of proposition 1).

From the conditions (C.5) $r_i \leq C_1(\sigma)$ and (C.6) $C_2(\sigma) - C_1(\sigma) \leq p_{i1}$, (e.2) can be reduced to $C_2(\sigma i) = C_1(\sigma) + p_{i1} + p_{i2}$.

In addition, from (e.5) and (e.6), we have $C_2(\sigma j) = \max\{C_1(\sigma j), C_2(\sigma)\} + p_{j2} = \max\{\max\{\max(r_j, C_1(\sigma)) + p_{j1}, C_2(\sigma) - w_j\}, C_2(\sigma)\} + p_{j2} = \max\{(1)\ r_j + p_{j1} + p_{j2}, (2)\ C_1(\sigma) + p_{j1} + p_{j2}, (3)\ C_2(\sigma) - w_j + p_{j2}, (4)\ C_2(\sigma) + p_{j2}\}$ [3]. Here, from the condition (C.4) $p_{i1} + p_{i2} \leq p_{j1} + p_{j2}$, we can confirm that (e.2) $= C_2(\sigma i) = C_1(\sigma) + p_{i1} + p_{i2} \leq (2)\ C_1(\sigma) + p_{j1} + p_{j2} = C_2(\sigma j)$ of (e.6) , that is, $C_2(\sigma i) \leq C_2(\sigma j)$ is satisfied.

In conclusion, we have (A.1) $C_k(\sigma ij) \leq C_k(\sigma ji)$ and (A.2) $F(\sigma ij) \leq F(\sigma ji)$ for k = 1, 2. This completes the proof. ■

Proofs for the propositions (3-5) below are skipped in this paper, since it can be proven easily in the similar way with those of propositions 1-2.

Proposition 3. There is a given partial schedule σ, if we have two jobs i ($i \notin \sigma$) and j ($j \notin \sigma$) satisfying the following four conditions: (C.5) $r_i \leq C_1(\sigma)$, (C.2) $\min\{p_{i1}, p_{j2}\} \leq \min\{p_{i2}, p_{j1}\}$, (C.7) $C_2(\sigma) - C_1(\sigma) \leq p_{i1} + w_j$ and (C.8) $C_2(\sigma) + p_{i2} - C_1(\sigma) - p_{i1} \leq p_{j1} + w_j$, (C.4) $p_{i1} + p_{i2} \leq p_{j1} + p_{j2}$ and (C.9) $p_{i2} \leq p_{j2}$, then partial schedule σji is dominated by σij.

Proposition 4. There is a given partial schedule σ, if we have two jobs i ($i \notin \sigma$) and j ($j \notin \sigma$) satisfying the following four conditions: (C.10) $C_1(\sigma) \leq r_i \leq C_2(\sigma)$, (C.11) $r_i \leq r_j$, (C.2) $\min\{p_{i1}, p_{j2}\} \leq \min\{p_{i2}, p_{j1}\}$, (C.12) $C_2(\sigma) \leq r_i + p_{i1}$, (C.3) $p_{i2} \leq p_{j1} + w_j$ and (C.4) $p_{i1} + p_{i2} \leq p_{j1} + p_{j2}$, then partial schedule σji is dominated by σij.

Proposition 5. There is a given partial schedule σ, if we have two jobs i ($i \notin \sigma$) and j ($j \notin \sigma$) satisfying the following four conditions: (C.10) $C_1(\sigma) \leq r_i \leq C_2(\sigma)$, (C.11) $r_i \leq r_j$, (C.2) $\min\{p_{i1}, p_{j2}\} \leq \min\{p_{i2}, p_{j1}\}$, (C.13) $r_i + p_{i1} \leq C_2(\sigma) \leq r_i + p_{i1} + w_i$ and (C.14) $C_2(\sigma) + p_{i2} - r_i - p_{i1} \leq p_{j1} + w_j$ and (C.9) $p_{i2} \leq p_{j2}$, then partial schedule σji is dominated by σij.

Proposition 6. If there is job j*, where $r_{j*} + p_{j*1} + p_{j*2} \leq \min_{i \neq j*}\{r_i\}$, there exists an optimal schedule in which job j* is placed in the first position.

Proof. Let's assume that there is an optimal schedule in which j* is not placed in the first position. Then, the optimal schedule can be denoted as $\sigma_1 j^* \sigma_2$, where σ_1 and σ_2 are partial sequences of the optimal schedule, and σ_1 is a non-empty partial sequence. To prove the proposition, we show that $F(j^*\sigma_1\sigma_2) \leq F(\sigma_1 j^*\sigma_2)$ for any pair of σ_1 and σ_2.

From the condition $r_{j*} + p_{j*1} + p_{j*2} \leq \min_{i \neq j*}\{r_i\}$ of this proposition, all jobs (except for j*) cannot start their first operations earlier than $r_{j*} + p_{j*1} + p_{j*2}$. Therefore, even if j* is scheduled in the first position in a complete schedule ($j^*\sigma_1\sigma_2$), the starting times and the completion times of the jobs (in σ_1 of $j^*\sigma_1\sigma_2$) cannot be earlier than those of the jobs (in σ_1 of $\sigma_1 j^*\sigma_2$).

That is, if we assume that job i($i \neq j^*$) is scheduled in the first position of σ_1 and we can show $C_k(i) = C_k(j^*i)$, k=1, 2, the proof of this proposition is completed. Here, we have $C_1(i) = r_i + p_{i1}$ and $C_2(i) = r_i + p_{i1} + p_{i2}$. On the other hand, we have $C_1(j^*i) = \max\{C_1(j^*), r_i\} + p_{i1}$, $C_2(j^*i) = \max\{C_1(j^*i), C_2(j^*)\} + p_{i2}$, $C_1(j^*) = r_{j*} + p_{j*1}$ and $C_2(j^*) = r_{j*} + p_{j*1} + p_{j*2}$. In addition, from the condition $r_{j*} + p_{j*1} + p_{j*2} \leq \min_{i \neq j*}\{r_i\}$ of this proposition, $C_1(j^*i)$ and $C_2(j^*i)$ can be reduced to $C_1(j^*i) = \max\{C_1(j^*), r_i\} + p_{i1} = \max\{r_{j*} + p_{j*1}, r_i\} + p_{i1} = r_i + p_{i1}$ and $C_2(j^*i) = \max\{C_1(j^*i), C_2(j^*)\} + p_{i2} = \max\{r_i + p_{i1}, r_{j*} + p_{j*1} + p_{j*2}\} + p_{i2} = r_i + p_{i1} + p_{i2}$.

In conclusion, we have $C_k(i) = C_k(j^*i)$ for k = 1, 2. This completes the proof. ■

From proposition 6, the following corollary holds:

Corollary 1. There is a given partial schedule σ, if we have two jobs i ($i \notin \sigma$) and j ($j \notin \sigma$) satisfying the following condition: $C_2(\sigma j^*) \leq \min_{i \neq j*}\{r_i\}$, then schedules that start with σj^* dominate the others.

Lower Bounding schemes

To calculate a lower bound on the total flowtime of a complete schedule constructed from a given partial schedule **σ**, we add lower bounds on flowtimes of the unscheduled jobs (that are not included to the partial schedule **σ)** to the total flowtime of the scheduled jobs in the partial schedule **σ**. First, we show a modified version (proposition 7) based on the propostion of Kim [17], and it will be used to obtain a lower bound on the total flowtime. Here, Let C_i, i = 1, 2,. . . , n be nonnegative real numbers. Also, let $r_{(i)}(S)$ denote the ready time of the job i to be completed i-th in schedule S. This proposition can be proved by the proposition of Kim [17].

Proposition 7. If $C_1 \leq C_2 \leq \ldots \leq C_n$, then for any schedule S' in a typical two-machine permutation flowshop,

$$\sum_i \max\{0, C_i - r_{(i)}(S')\} \geq \sum_i \max\{0, C_i - r_{(i)}(S_{ERT})\},$$

where S_{ERT} is the earliest ready time sequence, in which jobs with earlier ready times are placed earlier.

In this proposition, if we let C_i be the completion time of the job that is placed ith among jobs in an arbitrary sequence S for the two-machine permutation flowshop, $C_1 \leq C_2 \leq \ldots \leq C_n$ is satisfied (for any sequence). In addition, if we let $r_{(i)}$ (S) denote the ready time of the job that is placed ith among jobs in S, from the proposition 7, the total flowtime of sequence S is no less than $\sum_i \max\{0, C_i - r_{(i)}(S_{ERT})\}$.

Here, we cannot know the exact completion times of the (unscheduled) jobs that are not included in a partial schedule σ. Therefore, we calculate the lower bounds on the completion times of the unscheduled jobs in a partial schedule σ by using the following propositions. In addition, the notations below are used in the following propostions.

σ	partial schedule corresponding to a node being considered in the branch and bound algorithm
U	set of jobs not scheduled yet, i.e., those that are not included in σ
w*	the largest limited waiting time of jobs in U
r*	the shortest processing time of jobs in U
$p_{\{j\}k}$	the j-th shortest processing time on machine k among jobs in U
P_{jk}	sum of processing times of j operations in the order of the shortest processing times (on machine k) among jobs in U, i.e., $P_{jk} = \sum_{q=1}^{j} p_{\{q\}k}$
lb_j^m	lower bounds (m = 1, 2, 3, there are three versions of lower bounds) on the completion time on machine 2 of the job that is completed j-th among jobs in U (in any complete schedule that starts with a partial schedule σ)

In the proposition 8, we can obtain three lower bounds (lb_j^1, lb_j^2 and lb_j^3) on the completion times of the unscheduled jobs, associated with a partial schedule σ through the relaxation of ready times and limited waiting times of jobs in U.

Proposition 8. In the two-machine permutation flowshop, the completion time of the job that is completed j-th among jobs in U (in any complete schedule that starts with σ) is no less than $lb_j^1 = \max[\max(C_1(\sigma), r^*) + p_{\{1\}1}, C_2(\sigma)] + P_{j2}$ or $lb_j^2 = \max[\max(C_1(\sigma), r^*), C_2(\sigma) - w^* - p_{\{1\}1}] + P_{j1} + p_{\{1\}2}$ or $lb_j^3 = \max[lb_j^1, lb_j^2]$.

Proof. P_{jk} is a lower bound on the time needed to operate j jobs among jobs included in U on machine k. In the case of lb_j^1, since the jobs included in U cannot start its second operation (on machine 2) earlier than $\max(C_1(\sigma), r^*) + p_{\{1\}1}$ or $C_2(\sigma)$, the completion time of the job that is completed j-th among jobs included in U is no less than $\max[\max(C_1(\sigma), r^*) + p_{\{1\}1}, C_2(\sigma)] + P_{j2}$.

Also, in the case of lb_j^2, since the jobs included in U cannot start its first operation (on machine 1) earlier than than $\max(C_1(\sigma), r^*)$ or $C_2(\sigma) - w^* - p_{\{1\}1}$, the completion time of the job that is completed j-th among jobs included in U is no less than $\max[\max(C_1(\sigma), r^*), C_2(\sigma) - w^* - p_{\{1\}1}] + P_{j1} + p_{\{1\}2}$. Finally, since the proofs of lb_j^1 and lb_j^2 are completed, lb_j^3 (max [lb_j^1, lb_j^2]) is obvious. This completes the proof. ■

Using the two propositions above (proposition 7 and 8), we can obtain a lower bound on the total flowtime of any complete schedule that can be constructed from a partial schedule σ as

$$\sum_{i \in \sigma} \max\{0, c_{i2}(\sigma\sigma - r_i\} + \sum_{j=1}^{|U|} \max\{0, lb_j^m - r_{[j]}(S_{ERT})\}$$
for m=1, 2, 3,

where $r_{[j]}(S_{ERT})$ denotes the ready time of the job completed j-th in an earliest ready time sequence of jobs in U, and therefore, we can obtain the three versions of lower bounds (LB1, LB2 and LB3) by using lb_j^1, lb_j^2 and lb_j^3.

Branch and Bound Algorithms

In this research, we suggest various versions of branch and bound algorithms for the considered scheduling problem. The suggested branch and bound algorithms are based on the typical branch and bound algorithm of Baker [18]. In the 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 [3, 14].

In the branch and bound algorithms, to obtain initial upper bounds, AGLS (Applied Greedy Local Search) algorithm of Choi [4] is used. For the same scheduling problem of this research, Choi [4] suggested four heuristic algorithms, H1, H2, AN (Applied NEH) and AGLS. Among the four heuristic algorithms above, AGLS showed the best performance. AGLS algorithm was developed based on the extended neighborhood search algorithm of Kim [19]. In AGLS algorithm, the initial solution (sequence) obtained from AN algorithm is then improved by interchanging a pair of jobs in the sequence, and all pairs of jobs are considered for interchanges at each attempt for improvement, and then this algorithm is terminated when the current solution cannot be improved any more [2 and 4]. The AN algorithm applies MN (modified NEH) algorithm of Choi and Kim [14] to the scheduling problem of this research.

Also, in the suggested branch and bound algorithm, child nodes are generated from a selected parent node, and we prune some child nodes (dominated by others) by using the developed dominance properties in this research. In addition, for the nodes that are not pruned by the dominance properties, we calculate the developed three lower bounds (LB1, LB2 and LB3), and the we eliminate the nodes with larger lower bounds 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 [3].

Simulation Tests

To confirm the effectiveness of the suggested lower bounding schemes, the developed dominance properties and the existing initial upper bound, and to show the performnaces of the branch and bound algorithms, we executed some simulation tests on randomly generated problems. The simulation test problems and the branch and bound algorithms were coded in C++ language, and computational experiments were done on a desktop computer with a i5 processor of 2.3 GHz clock speed. We generated the processing times on the two machines (the first and second machines), the ready times and the limited waiting times of jobs in the same way with those of Choi [4] as follows.

That is, the processing times (on the machine 1 and the machine 2) were randomly generated from DU(1, 50), where DU(a, b) means the discrete uniform distribution with a range of [a, b], and limited waiting times of jobs were generated from DU(1, 100), and the ready times of jobs were randomly generated from DU(0, LB3), in which LB3 is the lower bound suggested in this research [4].

To evaluate the effectiveness of the suggested lower bounding schemes, the developed dominance properties and the existing initial upper bound, we randomly generated 60 test problems, 10 problems for each of six levels (10, 11, 12, 13, 14 and 15) for the number of jobs. By conducting the tests above, we suggested the best branch and bound algorithm, and to see the the performnace of the best branch and bound algorithm, we generated 90 additional test problems, 10 problems for each of nine levels (16, 18, 20, 22, 24, 26, 28, 30 and 32) for the number of jobs. In tables 2-5, for a test instance, the CPU time for a branch and bound algorithm means the CPU time (in seconds counter) to find optimal solution, and if a branch and bound algorithm cannot give the optimal solution within 3600 seconds (1 hour), its CPU time becomes 3600 seconds. For a branch and bound algorithm, as average CPU times (ACPUT) and median CPU times (MCPUT) decrease, the performances of the branch and bound algorithms get better.

In addition, in tables 2-4, for a test instance, the ratio of CPU time for a branch and bound algorithm means its relative ratio of CPU time in comparison with that of other branch and bound algorithm (BB3), and the average ratio of CPU time for a B&B algorithm is the mean of the ratio values of CPU times (ARCPUT) for 10 instances for each of six levels for the number of jobs. For example, in a test problem, 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" increases, the performance of BB1 gets worse in comparison with BB1.

Table 2: Results of the tests on the effectiveness of the lower bounds

n	ACPUT† (MCPUT‡)			ARCPUT*	
	BB1	BB2	BB3	BB1/BB3	BB2/BB3
10	0.177	0.006	0.006	133.767	1.810
	(0.130)	(0.001)	(0.001)		
11	5.911	0.041	0.028	5408.652	2.158
	(0.625)	(0.010)	(0.010)		
12	4.147	0.042	0.021	3207.830	1.280
	(2.025)	(0.006)	(0.006)		
13	20.866	0.092	0.028	20027.546	3.300
	(1.700)	(0.010)	(0.006)		
14	158.327	0.149	0.091	138617.170	1.158
	(30.925)	(0.030)	(0.030)		
15	336.424	0.803	0.550	93783.888	1.027
	(47.245)	(0.040)	(0.040)		

†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

‡ median CPU time

* average ratio of CPU times of a branch and bound algorithm in comparison with those of BB3

Firstly, we performed three branch and bound algorithms (BB1, BB2 and BB3) with different lower bounds (LB1, LB2 anc LB3), respectively. In BB1, BB2 and BB3, all the suggested dominance properties were used in the branch and bound algorithms, and the solution of GLS is used as initial upper bound. In the Table 2, BB3 shows the best performances in terms of the average CPU time and the median CPU time. Also, in term of average ratio of CPU time, ARCPTs of BB1/BB3 are larger than those of BB2/BB3, which means that BB2 outperforms BB1 (LB2 is more efficient than LB1).

Secondly, we evaluated the effectiveness of the developed dominance properties. For this evaluation, we compared BB4 (in which none of the dominance properties is used) with BB3 (in which all dominance properties are used). The results of the tests are shown in Table 3. When the suggested dominance properties are not used, the CPU times increase, that is, the suggested dominance properties are effective.

Table 3: Results of the test on the effectiveness of the dominance properties

n	ACPUT† (MCPUT‡)		ARCPUT*
	BB3	BB4	BB4/BB3
10	0.006	0.006	1.753
	(0.001)	(0.001)	
11	0.028	0.034	1.129
	(0.010)	(0.006)	
12	0.021	0.029	1.933
	(0.006)	(0.010)	
13	0.028	0.044	2.182
	(0.006)	(0.010)	
14	0.091	0.126	1.274
	(0.030)	(0.050)	
15	0.550	0.766	1.262
	(0.040)	(0.090)	

†, ‡, * See the footnotes of Table 2.

Table 4: Results of the test on the effectiveness of the initial upper bound

n	ACPUT† (MCPUT‡)		ARCPUT*
	BB3	BB5	BB5/BB3
10	0.006	0.207	186.200
	(0.001)	(0.030)	
11	0.028	0.721	632.276
	(0.010)	(0.050)	
12	0.021	2.170	1461.590
	(0.006)	(0.140)	
13	0.028	39.818	3696.537
	(0.006)	(0.285)	
14	0.091	944.656	20399.859
	(0.030)	(10.400)	
15	0.550	807.385	6803.230
	(0.040)	(16.070)	

†, ‡, * See the footnotes of Table 2.

Thirdly, we compared BB5 (in which initial upper bounds were not used) with BB3 (in which the initial upper bounds were used). The results of the tests are shown in Table 4. When the initial upper bounds are not used, the CPU times increase significantly, that is, the brand and bound algorithm with the initial upper bounds becomes effective.

Finally, to show the performances of the best branch and bound algorithm (BB3), we conducted the additional simulation tests. Therefore, we randomly generated 90 additional test problems, 10 problems for each of nine levels (16, 18, 20, 22, 24, 26, 28, 30 and 32) for the number of jobs, and in the Table 5, we show the ACPUT, MCPUT and NINS (the number of instances that have not been solved to optimality by BB3 within 3600s among 10 instances). As you see in Table 5, BB3 could find optimal solutions for problems with up to 28 jobs in a reasonable amount of CPU time.

Table 5: Results of the test on the BB3 algorithm

n	ACPUT†	MCPUT‡	NINS[#]
16	0.215	0.130	0
18	1.136	0.130	0
20	32.553	1.920	0
22	304.025	8.820	0
24	685.476	61.630	1
26	1138.969	355.640	2
28	1803.423	1803.423	4
30	2132.195	2187.120	5
32	2882.403	3600.010	6

†, ‡ See the footnotes of Table 2.

[#] number of instances among 10 instances that have not been solved to optimality by the branch and bound algorithm within 3600s

CONCLUDING REMARKS

In this research, we suggested the branch and bound algorithms for the scheduling problem with the objective function of minimizing the total flowtime in a two-machine permutation flowshop with ready times and limited waiting times of jobs in semiconductor fabs. To boost the efficiency of the suggested branch and bound algorithms, we developed several dominance properties and lower bounding schemes, and the upper bounds from the existing heuristic algorithms were used in those. From the results of the simulation tests, we could confirm that the branch and bound algorithms became more efficient by using the developed dominance properties, lower bounding schemes and initial upper bounds, and the best branch and b ound algorithm (BB3) could find optimal solutions for problems with up to 28 jobs in a reasonable amount of CPU time.

We can (need to) extend the results of this research in various directions. For example, it may be necessary to develop dominance properties and lower bounds for the m-machine permutation flowshop scheduling flowshop problems or flowshops with the other objective functions (minimizing makespan or total tardiness). Also, more general cases can be considered, in which setup times are required between the operations of jobs with different recipes. We can find such scheduling problems easily in a semiconductor manufacturing line.

ACKNOWLEDGMENTS

This work was supported by Kyonggi University Research Grant 2015.

REFERENCES

[1] Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G., 1979, “Optimization and approximation in deterministic sequencing and scheduling,” Annals of Discrete Mathematics, 5, pp. 287-326.

[2] Choi, S. W., 2016, “Heuristics for a two-machine permutation flowshop with limited queue time constraint and arrival times in a semiconductor manufacturing line,” International Journal of Applied Engineering Research, 11(8), pp. 5852-5855.

[3] Choi, S. W., 2015, “Optimization Algorithms for a Two-Machine Permutation Flowshop with Limited Waiting Times Constraint and Ready Times of Jobs,” Journal of Information Technology Applications & Management, 22, pp. 1-17.

[4] Choi, S. W., 2017, “Minimizing total flowtime in a two-machine flowshop with limited waiting time constraint and ready time in 12 a semiconductor fabrication plant,” Submitted to International Journal of Operations and Quantitative Management.

[5] Joo, B. J., and Kim, Y. D., 2009, “A branch-and-bound algorithm for a two-machine flowshop scheduling problem with limited waiting time constraints,” Journal of the Operational Research Society, 60, pp. 572-582.

[6] Choi, S. W., Lim, T. K., and Kim, Y. D., 2010, “Heuristics for Scheduling Wafer Lots at the Deposition Workstation in a Semiconductor Wafer Fab,” Journal of the Korean Institute of Industrial Engineers, 36, pp. 125-137.

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

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

[9] Gupta, J. N. D., and Stafford, J. E., 2006, “Flowshop scheduling research after five decades”, European Journal of Operational Research,” 169, pp. 699-711.

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

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

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

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

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

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

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

[17] Kim, Y. D., 1974, 1993, "A new branch and bound algorithm for minimizing mean tardiness in two-machine flowshops," Computers and Operations Research, 20, pp. 391–401.

[18] Baker, K. R., 1974, Introduction to Sequencing and Scheduling, John Wiley & Sons, New York.

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