International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V73I12P124

Volume 73 Issue 12, 297-305, December 2025

Original Article

B&B Algorithms for Minimization of Total Weighted Flowtime in Two-Machine Flowshop with Re-Entrant Flow

Choi Seong-Woo

Department of Business Administration, Kyonggi University, Suwon-si, South Korea

Corresponding Author : swchoi@kyonggi.ac.kr

Received: 20 August 2025 Revised: 02 December 2025 Accepted: 09 December 2025 Published: 19 December 2025

Abstract - *This research investigates the two-machine re-entrant permutation flowshop scheduling problem in which every job undergoes two consecutive processing cycles through the same pair of machines following the sequence $M_1 \rightarrow M_2 \rightarrow M_1 \rightarrow M_2$. The aim of this problem is the minimization of the total weighted flowtime, and branch and bound algorithms are developed that integrate newly formulated dominance rules, a lower-bounding procedure, and an effective heuristic to generate high-quality initial schedules. The algorithm's efficiency is evaluated using extensive computational experiments on randomly generated instances. The experiments confirm that the suggested branch and bound algorithm can identify optimal schedules while significantly reducing computation time by integrating some dominance rules, a lower-bounding procedure, and an effective heuristic.*

Keywords - *Branch-and-Bound, Dominance Rule, Lower Bound, Heuristic, Re-entrant Flowshop, Weighted Flowtime.*

1. Introduction

Re-entrant flowshops are commonly encountered in modern production systems such as semiconductor manufacturing, textile dyeing, mirror processing, and Printed Circuit Board (PCB) assembly lines, where products revisit the same machines multiple times. Proper scheduling of these systems plays a crucial role in improving throughput and minimizing overall production time.

Most prior studies on re-entrant flowshop scheduling have focused on objectives such as makespan or maximum tardiness minimization [1, 2]. However, in many real-world production environments, jobs often vary in importance or urgency, making the minimization of total weighted flowtime a more realistic and meaningful criterion. Despite this, studies on minimizing total weighted flowtime in re-entrant scheduling environments remain scarce, particularly for systems involving two machines with repeated job visits.

Recent literature has begun addressing flowshop models that incorporate weighting factors into scheduling objectives. For example, von Aspern et al. [3] examined flowshops with re-entry under total weighted completion time, proposing approximation and priority-based rules for general *m*-machine settings. Yuan et al. [4] introduced a reinforcement learning framework for scheduling re-entrant flowshops with weighted completion measures. In related research, Tang et al. [5] studied hybrid flowshop environments that combine re-entrant and missing operations, while jointly optimizing energy consumption and weighted completion time. Nevertheless, the scheduling problem of this research has not been systematically analyzed.

The present research addresses this gap by developing a formal model and a solution approach for the two-machine re-entrant permutation flowshop scheduling problem that minimizes total weighted flowtime. Although Pan and Chen [2] proved that the unweighted version of this problem is *NP*-hard and proposed a mixed-integer programming formulation, their and other subsequent studies mainly focused on makespan- or tardiness-oriented objectives. According to the authors' examination of existing studies, no published work appears to deal explicitly with the weighted-flowtime version of this problem.

This paper offers several notable contributions, which are summarized below:

- A rigorous formulation of the two-machine re-entrant permutation flowshop is developed, capturing the objective of minimizing total weighted flowtime.
- This research proposes a refined branch-and-bound method that integrates some dominance properties and a lower-bounding strategy specifically constructed for optimizing total weighted flowtime.

- A heuristic initialization method is designed to improve computational performance for large-scale instances.
- Comprehensive numerical analyses are provided to demonstrate the superiority of the proposed approach over existing makespan- or tardiness-oriented methods.

According to Graves et al. [6], "a re-entrant flowshop is characterized by jobs revisiting the same machines multiple times, thereby generating re-entrant flows."

Two-step re-entrant production lines that can be represented as permutation flowshops are frequently found in various industries. In textile dyeing factories, fabrics undergo two consecutive processes-a dyeing process and a drying process-and these steps are usually repeated twice for deep dyeing. In mirror manufacturing, the back of the mirror is treated to produce a clear image of the reflected object, and the painting and drying processes are generally repeated twice. In PCB production lines, component insertion is typically carried out by either automated assembly machines or manual assembly stations, and the selection between these methods varies with the type of electronic component involved. Because the process must be carried out on both sides of the board, a two-machine re-entrant permutation flowshop arises that requires visiting the automatic assembly equipment and manual assembly lathe twice [10]. The scheduling objective is the minimization of total weighted flowtime, which simultaneously accounts for production lead time and job importance. Minimizing total flowtime becomes crucial when in-process buffers are limited and when product priorities differ due to due dates or production plans [7-9].

Representative studies addressing scheduling problems related to flow processes with re-entrant flows include the work of Graves et al. [6], who defined semiconductor fabrication as a re-entrant flowshop or job shop and developed an efficient heuristic methodology to improve productivity. Demirkol and Uzsoy [10] proposed a method to minimize maximum tardiness for re-entrant flow processes with setup times. Pan and Chen [2] demonstrated that minimizing total flowtime in a two-machine permutation re-entrant flow shop is NP-hard and proposed a mixed-integer programming formulation and a heuristic approach. Building on these foundations, several researchers developed several B&B and heuristic algorithms for makespan and tardiness minimization. The present study extends this line of research by addressing the weighted-flowtime objective.

Within the two-machine re-entrant permutation flowshop framework, as shown in Figure 1, each job undergoes two consecutive processing cycles across machine 1 and machine 2, following the sequence $M_1 \rightarrow M_2 \rightarrow M_1 \rightarrow M_2$. That is, two cycles of a job through the two machines are modeled as partial jobs, thereby transforming the scheduling problem into an equivalent flowshop involving $2n$ partial jobs.

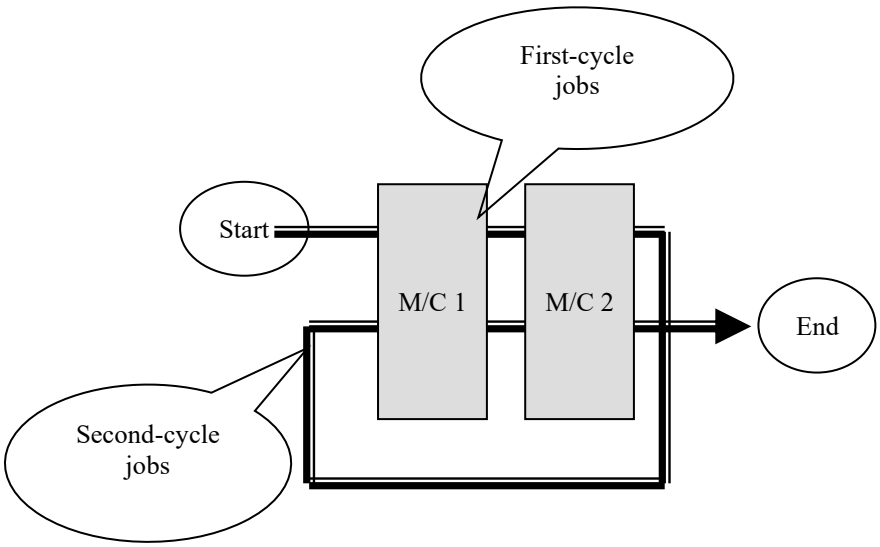

Fig. 1 View of a two-machine re-entrant flowshop

Here, a precedence dependency exists between the two cycles of the same job, requiring that the first operation of the second-cycle job on machine 1 commence only after completion of the corresponding second operation of the first-cycle job on machine 2 (refer to Figure 2). The following assumptions are adopted for the problem formulation and experimental analysis. All jobs are independent and re-entrant, each requiring two processing cycles through the two machines in the same order ($M_1 \rightarrow M_2 \rightarrow M_1 \rightarrow M_2$). Processing times are deterministic and generated from a uniform distribution $U[1,10R]$, where R controls the variability of processing times and takes values of 1 or 2 in this study. Job weights are independently drawn from a uniform distribution $U[1,10]$.

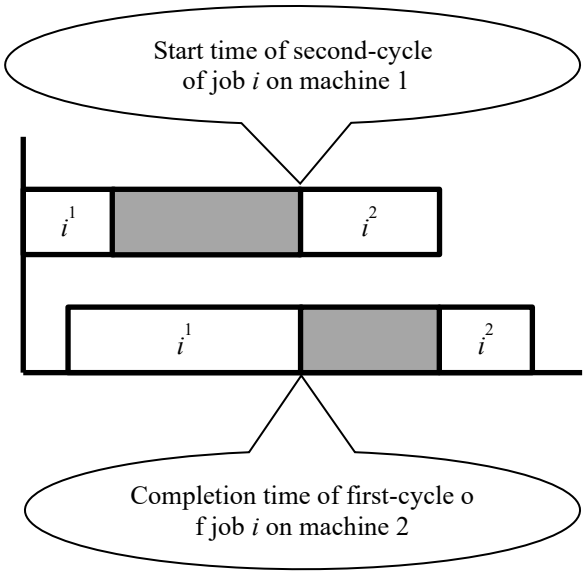

Fig. 2 Precedence between the operation of the first cycle on M/C 2 and the second cycle on M/C 1 of job *i*

In this shop, sequence-dependent setup times, machine breakdowns, or transfer delays are not considered. The processing times of the first and second cycles are assumed to be statistically independent, and all machines are continuously available during the scheduling horizon. These assumptions are consistent with previous re-entrant flowshop scheduling studies [10] and provide a clear basis for evaluating the relative performance and robustness of the proposed algorithms.

The *NP*-hardness of this problem can be inferred from the reasoning presented below. Garey and Johnson [11] demonstrated that "the scheduling problem of minimizing the total flowtime in a two-machine permutation flowshop is NP-hard." Their result implies that the current problem constitutes a special case of that general class if the operation times of all second-cycle partial jobs are assumed to be zero on both machines and all job weights are identical. Consequently, the scheduling problem addressed in this study is also classified as *NP*-hard.

The remainder of this paper is organized as follows. Section 2 describes the dominance properties. Section 3 introduces the B&B algorithm and its lower-bounding scheme. Section 4 presents the heuristic for generating initial solutions. Sections 5 and 6 report the computational results and robustness/sensitivity analysis of B&Bs. Finally, section 6 concludes the study.

2. Dominance Properties

This section presents the dominance principles employed to identify and eliminate sub-optimal partial schedules during the branch-and-bound search. A dominance relation is used to determine whether one partial sequence is guaranteed to yield an equal or smaller total weighted flowtime than another. If a partial sequence is dominated, all extensions derived from it can be safely pruned.

Following the formulation, each job i is composed of two partial jobs (first-cycle and second-cycle). The first-cycle partial job must finish its operation on machine 2 before the corresponding second-cycle partial job begins on machine 1. When the job order is identical across both machines, the system is defined as a two-machine permutation re-entrant flowshop. Once the processing sequence of all partial jobs is determined, the total weighted flowtime can be directly computed. Therefore, in this study, the terms "sequence" and "schedule" are used interchangeably.

A permutation schedule for the two-machine re-entrant flowshop is defined as one in which the job order on both machines is identical when the first- and second-cycle partial jobs are treated as independent jobs. In a standard two-machine flowshop, a permutation schedule likewise implies identical job sequences on both machines.

Therefore, for the scheduling problem of the two-machine permutation flowshop aiming to minimize the total weighted flowtime, once the processing order of the partial jobs is fixed, the corresponding schedule and the flowtime of each job can be readily computed. Throughout this study, the terms "sequence" and "schedule" are used interchangeably.

The notations, the parameters, and symbols used in this study are defined as follows.

i^k:	kth-cycle partial-job of job i (i=1, 2,…, n, and k=1, 2)
$i^\bullet$:	partial-job of job i ($i^\bullet$ can be either a first-cycle partial-job or a second-cycle partial-job)
w_i:	weight of job i
$p_{i^k j}$:	operation time of partial-job i^k on machine j (j=1, 2)
σ:	A partial schedule representing the preceding portion of the complete job sequence
$c_{i^k j}(S)$:	time at which partial-job i^k finishes on machine j in schedule S
$C_j(S)$:	makespan of schedule S, defined as the time when the operations of all partial jobs are finished
π:	arbitrary (partial) sequences of partial-jobs that are not included in σ ($\|\sigma \cup \pi\|=n$)
$\sigma_{i_1^\bullet, i_2^\bullet, \ldots, i_l^\bullet}$:	partial sequence (schedule) obtained with σ followed by partial-jobs $i_1^\bullet, i_2^\bullet, \ldots, i_l^\bullet$.
$\sigma\pi$:	(partial) sequence obtained with σ followed by π
$TWF(S)$:	total weighted flowtime of jobs in, i.e., $\sum_{i_1^\bullet \in S}[w_i \times c_{i^2 2}(S)]$

The completion time of job i is defined as the completion time of its second-cycle partial job on machine 2. Therefore,

the completion time of job i in the schedule ($i^2 \in S$) is represented as $c_{i^2 2}(S)$.

For example of dominance property 1, consider two partial schedules ($\sigma i^1 k^1 l^\bullet$ and $\sigma\ k^1 i^1 l^\bullet$). If the completion times of the partial schedule ($\sigma i^1 k^1 l^\bullet$) on both machines are no greater than those of the partial schedule ($k^1 i^1 l^\bullet$), then the partial schedule ($\sigma k^1 i^1 l^\bullet \pi$) is said to be dominated and can be pruned from the B&B tree. This example illustrates the intuitive basis of the dominance concept - eliminating schedules that are guaranteed to be inferior in terms of total weighted flowtime. Propositions 1–3 formalize these dominance rules mathematically, establishing sufficient conditions for pruning based on completion time inequalities and flowtime comparisons. By applying these dominance properties, the B&B algorithm substantially reduces the number of explored nodes and enhances computational efficiency without compromising optimality.

The proof ideas follow the logic of Schaller [12] for tardiness-minimization flowshops. If a partial schedule A always yields completion times no greater than those of another schedule B for all remaining extensions, then B is said to be dominated and can be pruned. Through these propositions, branches in the B&B tree that cannot lead to improved total weighted flowtime are efficiently eliminated.

Proposition 1. Let σ be a partial schedule and let three partial-jobs i^1, k^1, $l^\bullet$(i^1,k^1,$i^\bullet \notin \sigma$, $l^\bullet \neq k^2$). If $C_2(\sigma i^1 k^1) \leq C_1(\sigma i^1 k^1 l^\bullet)$, then any complete schedules ($\sigma k^1 i^1 l^\bullet \pi$) are dominated by others ($\sigma i^1 k^1 l^\bullet \pi$). Hence, the node ($\sigma k^1 i^1 l^\bullet \pi$) can be pruned.

Proof of Proposition 1. Proposition 1 can be proved by showing that the following inequalities (A), (B), and (C) are satisfied.

(A) $C_j(\sigma i^1 k^1 l^\bullet) \leq C_j(\sigma k^1 i^1 l^\bullet)$ for j=1, 2.

(B) $TWF(\sigma i^1 k^1 l^\bullet) \leq TWF(\sigma k^1 i^1 l^\bullet)$

(C) $\boldsymbol{c_{k^2 2}}(\sigma i^1 k^1 l^\bullet \pi) \leq \boldsymbol{c_{k^2 2}}(\sigma\ k^1 i^1 l^\bullet \pi)$

To show (A) $C_j(\sigma i^1 k^1 l^\bullet) \leq C_j(\sigma k^1 i^1 l^\bullet)$, we can obtain $C_1(\sigma i^1 k^1) = C_1(\sigma k^1 i^1) = C_1(\sigma) + p_{i^1 1} + p_{k^1 1}$. Also, due to $l^\bullet \neq k^2$, $C_1(\sigma i^1 k^1 l^\bullet) \leq C_1(\sigma k^1 i^1 l^\bullet)$ - (A.1).

In addition, from (A.1) and the condition $C_2(\sigma i^1 k^1) \leq C_1(\sigma i^1 k^1 l^\bullet)$ of this proposition, we can prove that

$C_2(\sigma i^1 k^1 l^\bullet) = C_1(\sigma i^1 k^1 l^\bullet) + p_{l^\bullet 2} \leq$

$\max\{C_1(\sigma k^1 i^1 l^\bullet), C_2(\sigma k^1 i^1)\} + p_{l^\bullet 2} = C_2(\sigma k^1 i^1 l^\bullet)$ - (A.2).

From (A.2) and the condition $l^\bullet \neq k^2$ of this proposition, we can confirm that the completion times of all partial-jobs and the weighted completion times of all second-cycle partial-jobs in $\sigma i^1 k^1 l^\bullet$ are not greater than those in $\sigma k^1 i^1 l^\bullet$(except for k^1). Therefore, we can prove that $WF(\sigma i^1 k^1 l^\bullet) \leq WF(\sigma k^1 i^1 l^\bullet)$ - (B).

Finally, in a full schedule $\sigma i^1 k^1 l^\bullet \pi$, the completion times and the weighted completion times of all second-cycle partial-jobs are not greater than those in $\sigma k^1 i^1 l^\bullet \pi$ (except for k^1). However, from (A) and the condition $l^\bullet \neq k^2$ of this proposition, $\boldsymbol{c_{k^2 2}}(\sigma i^1 k^1 l^\bullet \pi) \leq c_{k^2 2}(\sigma\ k^1 i^1 l^\bullet \pi)$ - (C).

Therefore, from (A), (B), and (C), we can prove this proposition 1 ($TWF(\sigma i^1 k^1 l^\bullet \pi) \leq TWF(\sigma k^1 i^1 l^\bullet \pi)$ i.e.,

$\sum_{v^2 \in \sigma i^1 k^1 l^\bullet \pi} [w_v \times c_{v^2 2}(\sigma i^1 k^1 l^\bullet \pi)] \leq$

$\sum_{v^2 \in \sigma k^1 i^1 l^\bullet \pi} [w_v \times c_{v^2 2}(\sigma k^1 i^1 l^\bullet \pi)])$.

Proposition 2. Let σ be a partial schedule and let three partial-jobs i^2, k^1, $l^\bullet$ ($i^1 \in \sigma$, i^2,$k^1 \notin \sigma$, $l^\bullet \neq k^2$). If $C_2(\sigma i^2 k^1) \leq C_1(\sigma i^2 k^1 l^\bullet)$ and $C_{i^1 2}(\sigma) \leq C_1(\sigma)$. Then any complete schedules ($\sigma k^1 i^2 l^\bullet \pi$) are dominated by others ($\sigma k^1 i^2 l^\bullet$). Hence, the node ($\sigma k^1 i^2 l^\bullet \pi$) can be pruned.

Proof of Proposition 2. Proposition 2 can be proved by showing that the following inequalities (D), (E), and (F) are satisfied.

(D) $C_j(\sigma i^2 k^1 l^\bullet) \leq C_j(\sigma k^1 i^2 l^\bullet)$ for j=1, 2

(E) $TWF(\sigma i^2 k^1 l^\bullet) \leq TWF(\sigma k^1 i^2 l^\bullet)$

(F) $\boldsymbol{c_{k^2 2}}(\sigma i^1 k^2 l^\bullet \pi) \leq \boldsymbol{c_{k^2 2}}(\sigma\ k^2 i^1 l^\bullet \pi)$

To show (D) $C_j(\sigma i^2 k^1 l^\bullet) \leq C_j(\sigma k^2 i^1 l^\bullet)$, we can obtain $C_1(\sigma i^2 k^1) = C_1(\sigma k^1 i^2) = C_1(\sigma^1) + p_{i^2 1} + p_{k^1 1}$. Also, due to $l^\bullet \neq k^2$, $C_1(\sigma i^2 k^1 l^\bullet) \leq C_1(\sigma k^1 i^2 l^\bullet)$ - (D.1).

In addition, from (D.1) and the condition $C_2(\sigma i^2 k^1) \leq C_1(\sigma i^2 k^1 l^\bullet)$ of this proposition, we can prove that

$C_2(\sigma i^2 k^1 l^\bullet) = C_1(\sigma i^2 k^1 l^\bullet) + p_{l^\bullet 2} \leq$

$\max\{C_1(\sigma k^1 i^2 l^\bullet), C_2(\sigma k^1 i^2)\} + p_{l^\bullet 2} = C_2(\sigma k^1 i^2 l^\bullet)$ - (D.2).

From (D.2) and the condition $l^\bullet \neq k^2$ of this proposition, we can confirm that the completion times of all partial-jobs and the weighted completion times of all second-cycle partial-jobs in $\sigma i^2 k^1 l^\bullet$ cannot be larger than those in $\sigma k^2 i^1 l^\bullet$(except for k^1). Therefore, we can prove that $WF(\sigma i^2 k^1 l^\bullet) \leq WF(\sigma k^1 i^2 l^\bullet)$ - (E).

Finally, in a full schedule $\sigma i^2 k^1 l^\bullet \pi$, the completion times and the weighted completion times of all second-cycle partial-

jobs are not greater than those in $\sigma k^1 i^2 l^\bullet \pi$ (except for k^1). However, from (D) and the condition $l^\bullet \neq k^2$ of this proposition, $c_{k^2 2}(\sigma i^2 k^1 l^\bullet \pi) \leq c_{k^2 2}(\sigma\, k^1 l^\bullet \pi)$ - (F).

Therefore, from (D), (E), and (F), we can prove this proposition 1 ($TWF(\sigma i^1 k^1 l^\bullet \pi) \leq TWF(\sigma k^1 i^1 l^\bullet \pi)$, i.e.,

$$\sum_{v^2 \in \sigma i^2 k^1 l^\bullet \pi} [w_v \times c_{v^2 2}(\sigma i^2 k^1 l^\bullet \pi)] \leq$$

$$\sum_{v^2 \in \sigma k^1 i^2 l^\bullet \pi} [w_v \times c_{v^2 2}(\sigma k^1 i^2 l^\bullet \pi)]).$$

Proposition 3. Let σ be a partial schedule and let three partial-jobs i^2, k^1, k^2 ($i^1, k^1 \in \sigma$, $i^2, k^2 \notin \sigma$). If $C_j(\sigma i^2 k^2) \leq C_j(\sigma k^2 i^2)$ for j=1, 2 and $WF(\sigma i^2 k^2) \leq WF(\sigma k^2 i^2)$, then any complete schedules ($\sigma k^2 i^2 \pi$) are dominated by others ($\sigma i^2 k^2 l^\bullet \pi$). Hence, the node ($\sigma k^2 i^2 \pi$) can be pruned.

Let σ be a partial schedule and let three partial-jobs i^2, k^1, $l^\bullet$ ($i^1 \in \sigma$, $i^2, k^1 \notin \sigma$, $l^\bullet \neq k^2$). If $C_2(\sigma i^2 k^1) \leq C_1(\sigma i^2 k^1 l^\bullet)$ and $C_{i^1 2}(\sigma) \leq C_1(\sigma)$. Then any complete schedules ($\sigma k^1 i^2 l^\bullet \pi$) are dominated by others ($\sigma k^1 i^2 l^\bullet$). Hence, the node ($\sigma i^2 k^1 l^\bullet \pi$) can be pruned.

Proof of Proposition 3. From the conditions $C_j(\sigma i^2 k^2) \leq C_j(\sigma k^2 i^2)$ for j=1, 2 and $WF(\sigma i^2 k^2) \leq WF(\sigma k^2 i^2)$ of this proposition, we can know

$$\sum_{v^2 \in c\sigma i^2 k^2 l^\bullet \pi} [w_v \times c_{v^2 2}(\sigma i^2 k^2 l^\bullet \pi)] \leq$$

$$\sum_{v^2 \in \sigma k^2 i^2 l^\bullet \pi} [w_v \times c_{v^2 2}(\sigma k^2 i^2 l^\bullet \pi)], \text{ easily.}$$

3. B&B Algorithm

The proposed B&B framework extends Baker's [13] classical procedure to the two-machine re-entrant flowshop. Each node represents a partial sequence of processed partial jobs. At tree level k, nodes correspond to sequences of k partial jobs. Because there are 2n partial jobs, the tree depth is 2n. Precedence constraints require that the first cycle of each job precedes its second cycle.

A heuristic solution (Section 4) supplies the initial upper bound. Depth-first search is adopted, selecting nodes with the smallest lower bound. Dominance rules remove inferior branches. If a node's lower bound ≥ incumbent upper bound, the node is discarded. Terminal nodes yield complete schedules; if their total weighted flowtime is smaller, the incumbent is updated.

3.1. Lower Bounding Scheme

As stated earlier, in this scheduling problem, the flowtime of job i within a schedule S ($i^2 \notin S$) corresponds to the completion time of its second-cycle partial job on Machine 2. Accordingly, to estimate a lower bound for the total weighted flowtime of a complete schedule derived from an arbitrary partial schedule (σ), the following procedure is applied. First, the lower bound of the total weighted flowtime is computed for the second-cycle partial jobs that have not yet been included in the established partial schedule (σ). The resulting value is then added to the total weighted flowtime of the second-cycle partial jobs already contained in (σ). This combined value provides the lower-bound estimate of the objective function for the corresponding node in the B&B tree.

The notation system adopted in this study follows the formulation, and the symbols used in the explanation of the lower-bounding scheme are defined as follows.

U :set of partial jobs that have not yet been scheduled, that is, those not included in the current partial sequence σ.
$p_{\{k\}j}$: kth smallest processing time on machine j among partial-jobs in U.
$w_{\{k\}}$: kth lowest weight among the parent jobs corresponding to the second-cycle partial-jobs in U.
P_{kj}: total processing time on machine j for the k operations with the shortest processing times among all partial-jobs in U, i.e., $\sum_{q=1}^{k} p_{qj}$.
U^2: subset of U consisting of all second-cycle partial jobs.
A^2: subset of U_2 whose corresponding first-cycle partial jobs already appear in σ.
$lb_k^\bullet$: lower bound on the completion time of the k-th finished partial job among the elements of U, for any complete schedule that extends the partial sequence σ.
lb_m^2: lower bound on the completion time of the m-th finished second-cycle partial job among the elements of U_2, for any complete schedule that extends the partial sequence σ.

The flowtime of job i in schedule S equals the completion time of its second-cycle partial job on Machine 2. The lower bound for a node σ is obtained by adding the total weighted flowtime of scheduled partial jobs to an estimate for the unscheduled set U. Lower bounds for completion times are computed using Propositions 4 and 5, and the Rearrangement Inequality (Mitrinović [14]) is applied to produce a tight weighted sum estimate.

Proposition 4. In a two-machine permutation re-entrant flowshop, the operation completion time (on machine 2) of the k-th completed partial-job among the partial-jobs belonging to U in any entire schedule starting with σ is not smaller than the value of $\max\{C_1(\sigma)+P_{k1}+p_{\{1\}2}, C_2(\sigma)+P_{k2}, C_1(\sigma)+p_{\{1\}1}+P_{k2}\} \equiv lb_k^\bullet$, for k=1, 2,.., |U|.

From proposition 4, in any entire schedule starting with σ, we can obtain the lower bound on the operation completion time of the k-th completed partial-job among the partial-jobs belonging to U. However, to find the lower bound for the objective function of this scheduling problem, firstly, we have to calculate the lower bound for the operation completion time of the second-cycle partial-jobs belonging to U, which can be obtained by applying the following Proposition 5.

Proposition 5. In a two-machine permutation re-entrant flowshop, the operation completion time (on machine 2) of the m-th completed second-cycle partial-job among the partial-jobs belonging to U in any entire schedule starting with σ is not smaller than the value of $lb_m^2=lb_{k'}^{\bullet}$. Here, if m=1, 2,.., $|A^2|$, then, $k'=2m-|A^2|$ and $m=|A^2|+1, |A^2|+2,.., |U^2|$.

The proofs of Propositions 4 and 5 follow the reasoning presented in the study of previous research.

Proposition 6 corresponds to the well-known Rearrangement Inequality introduced by Mitrinović [14], which establishes how the ordering of elements in two sequences influences the value of their combined sum or product. Specifically, the inequality demonstrates that these values reach their extreme (maximum or minimum) forms depending on whether the sequences are similarly or oppositely ordered.

Proposition 6. Given two real sequences ($a_1 \le a_2 \le \ldots \le a_{n-1} \le a_n$) and ($b_1 \le b_2 \le \ldots \le b_{n-1} \le b_n$), the sum of the products of corresponding terms is minimized when one sequence is arranged in increasing order and the other in decreasing order. In other words, when arranged in the opposite order, $a_1 \times b_n + a_2 \times b_{n-1}, \ldots, a_{n-1} \times b_2 + a_n \times b_1$ attains its minimum value. Here, a_i and b_i are positive values, for i=1, 2,...,n.

By applying the results of Propositions 4, 5, and 6, a lower bound for the total weighted flowtime (corresponding to any complete schedule that extends the partial sequence σ) can be formulated as follows.

$$LB(\sigma)= \sum_{v^2 \in \sigma} [w_v \times c_{v^2 2}(\sigma)] + \sum_{m=1,\, m^2 \in U^2} [w_{(|U^2|-m+1)} \times lb_m^2]$$

4. Heuristics

To generate a strong initial upper bound, the NEH heuristic [15] is adapted to the re-entrant context. Jobs are first sorted by total processing time. Each partial job is then inserted into the position that minimizes the incremental total weighted flowtime while satisfying precedence constraints. The procedure iterates forward and backward until no further improvement occurs. This method provides a high-quality initial solution that greatly reduces the B&B search space.

The following is a modified version of the algorithm to suit the scheduling problem of this study. In order to explain the algorithm's procedure in more detail and with greater accuracy, it is organized as follows.

4.1. Procedure of Applied Heuristic

Step 1. Constructs a sorted sequence in ascending order according to the total process time of partial-jobs i^k ($\sum_{j=1}^{2} p_{i^k j}$, k=1, 2).

Step 2. In the sorted set, move second-cycle partial-jobs that do not satisfy the precedence constraints of the first-cycle partial-job and the second-cycle partial-job immediately after each corresponding first-cycle partial-job. We define that set as S^0.

Step 3. The set obtained by repeating the following (3.1-3.3) from u=1 to u=2n is defined as S^N.

3.1) Select the uth partial job from the initial sequence S^0.

3.2)For each possible insertion position k=1, 2, 3…, |σ|+1:

- Insert the selected partial job between the (k-1)th partial job and the kth partial job in the current partial sequence r.
- If the resulting sequence satisfies the precedence and feasibility conditions, evaluate the total weighted flowtime of the resulting schedule S, which consists of the current partial sequence of |σ|+1 partial-jobs followed by the remaining unscheduled partial-jobs originally located from position (u+1) to 2n in S^0.

3.3) Update σ with the partial sequence obtained by placing the selected partial job in the position that produces the minimum total weighted flowtime among all feasible alternatives.

Step 4. If the objective value of S^N obtained in Step 3 is not less than that of S^0, terminate the execution of this algorithm. Otherwise, reset S^0 to S^N ($S^0 = S^N$) and go to Step 5.

Step 5. The set obtained by repeating the following (3.1-3.3) from u=2n to u=1 is defined as S^N.

Step 6. If the objective value of S^N obtained in Step 5 is not less than that of S^0, terminate the execution of this algorithm. Otherwise, reset S^0 to S^N ($S^0 = S^N$) and go to Step 3.

5. Computational Experiments

All algorithms were implemented in C++ and executed on a PC (Intel Core i7-12700, 32 GB RAM, Windows 11). Every algorithm produced optimal solutions, so comparisons focus on CPU time and the number of generated nodes. Four versions were tested: BB1 (complete version), BB2 (without lower bound), BB3 (without dominance rules), and BB4 (without initial upper bound).

In Table 1, problem sizes involved 10-16 partial-jobs, and in Tables 2-3, problem sizes involved 14-20, with processing-time ranges R = 1 and 10. Five random instances per configuration produced 40 test cases in total. Job weights and processing times followed uniform distributions U[1, 10] and U[1, 10R], respectively. If an algorithm failed to reach optimality within 1,800 s, execution stopped. Since no standard benchmark datasets exist for this specific two-machine re-entrant flowshop problem,

the test instances were generated according to the data construction procedures used in Demirkol and Uzsoy [10]. The performance of each algorithm was evaluated using the CPU time (in seconds) and the number of generated nodes. The comparative outcomes are summarized in Table 1, which presents the average execution times and node-generation ratios of the two algorithms.

In this table, BB1 and BB2 refer respectively to the versions with and without the proposed lower-bounding scheme. Both versions employed the three dominance rules and the initial upper bound obtained through the NEH-based heuristic. The results clearly indicate that BB1 outperforms BB2 in terms of computational time and the number of nodes explored, demonstrating that the incorporation of the lower-bounding scheme substantially enhances the efficiency of the search process.

Table 1. Effectiveness of the lower bounding scheme

No. of Partial-jobs	ACPUT†		ARCPUT‡	ARSC#
	BB1	BB2	(BB1/BB2)	(BB1/BB2)
10	0.003	0.128	2.2×10^{-1}	5.6×10^{-3}
12	0.017	9.458	2.3×10^{-3}	9.0×10^{-4}
14	0.314	699.3	4.4×10^{-4}	1.81×10^{-4}
16	4.936	1800	-	-

†Average execution time or lower limit for execution time (if the optimal solution cannot be found within 1800 seconds, the execution time is assumed to be 1800 seconds)
‡Average execution time ratio
 # Average of the ratio of the number of nodes generated

Tables 2-3 show that dominance rules significantly reduced node counts, and the heuristic contributed modest additional time savings by lowering the initial bound.

Table 2. Efficiency of dominance properties

No. of Partial-jobs	ACPUT†		ARCPUT‡	ARSC#
	BB1	BB3	(BB1/BB3)	(BB1/BB3)
14	0.314	0.416	0.793	0.586
16	4.936	7.653	0.664	0.768
18	31.152	50.094	0.399	0.546
20	253.844	544.715	0.622	0.318

†‡# refer to Table 1

Table 3. Efficiency of the heuristic algorithm (initial upper bound)

No. of Partial-jobs	ACPUT†		ARCPUT‡	ARSC#
	BB1	BB4	(BB1/BB4)	(BB1/BB4)
14	0.314	0.314	0.998	0.883
16	4.936	5.301	0.931	0.906
18	31.152	32.382	0.962	0.940
20	253.844	287.153	0.884	0.899

†‡# refer to Table 1

The final test (Table 4) evaluated BB1 under the full configuration-lower bound, dominance rules, and heuristic upper bound. Partial-job counts ranged from 14 to 24 (6 levels), R = 1 and 10, yielding 60 random instances. The results indicate that BB1 can solve most problems with up to 22 partial jobs within reasonable CPU times, demonstrating both robustness and scalability.

Table 4. Main experimental results of BB1 algorithm

No. of Partial-jobs	*R*	ACPUT†	MCPUT‡	NPNS#
14	1	0.14	0.13	0
	10	0.14	0.11	0
16	1	0.88	0.58	0
	10	2.01	1.31	0
18	1	7.36	6.36	0
	10	9.28	7.64	0
20	1	70.61	30.59	0
	10	208.60	123.34	0
22	1	764.02	407.19	0
	10	972.39	472.14	1
24	1	1357.99	1734.16	2
	10	1848.13	1981.95	4

†refer to Table 1
‡Median value of execution time
#Number of problems for which the optimal solution was not found within 1800 seconds out of 5 problems

6. Robustness and Sensitivity Analysis of B&Bs

In this study, robustness and sensitivity analyses were exclusively based on CPU time, reflecting the algorithms' computational scalability and stability across different problem configurations. In addition, BB2, without the lower-bounding scheme, showed an exponential escalation in CPU time with increasing problem size. Therefore, it was omitted from the robustness and sensitivity analyses, as its results would not provide meaningful statistical interpretation.

While BB2 was omitted because its CPU time increased exponentially without the lower-bounding mechanism, further robustness and sensitivity analyses were performed on BB1, BB3, and BB4 to validate their performance differences statistically. Figure 3 summarizes the overall CPU time distributions for all tested configurations (four partial-job sizes and two R-values combined). It provides a general comparison of the computational efficiency and robustness of the algorithms, whereas Figures 4 and 5 further detail the effects of job size and processing-time range (R) individually. The CPU time distributions clearly show that BB1 consistently achieves the lowest computational cost, followed by BB4, while BB3 requires significantly more CPU time. This pattern holds across all problem sizes and processing-time ranges, indicating that the inclusion of the lower-bounding and dominance properties in BB1 and BB4 effectively improves search efficiency.

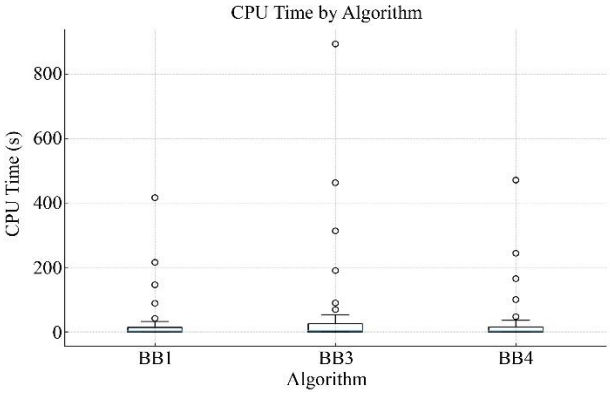

Fig. 3 CPU Times by B&Bs

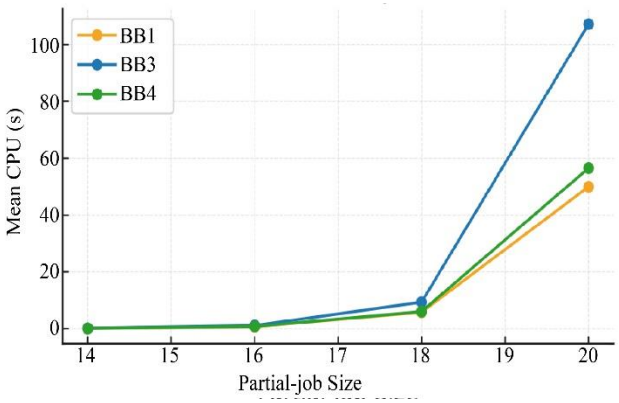

Fig. 4 Mean CPU vs Partial-Job Size (R=1)

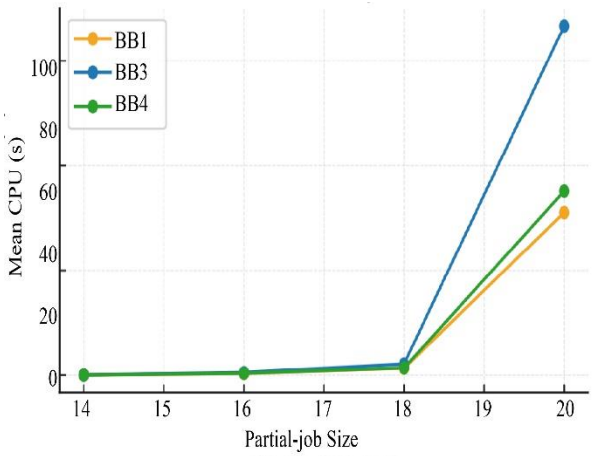

Fig. 5 Mean CPU vs Partial-Job Size (R=2)

Table 5 depicts the slope of CPU time growth with respect to partial-job size. The slope analysis indicates that BB1 remains the most stable and robust, while BB3 exhibits the steepest escalation rate, implying lower scalability for larger problems.

A three-way ANOVA was conducted with Algorithm, Partial-job size, and R as factors, using normalized CPU ratios (BB1=1). The results indicated significant main effects of Algorithm ($F=1.49\times10^{38}$, $p<0.001$), Problem size ($F=2.32\times10^{37}$, $p<0.001$), and R ($F=6.72$, $p=0.011$), as well as a strong interaction between Algorithm and Problem size.

Table 5. Summary of ANOVA Results (α = 0.05)

Factor	F-value	p-value	Interpretation
Algorithm	1.49×10^{38}	0.0000	Clear differences in CPU ratios among algorithms.
Partial-job size	2.32×10^{37}	0.0000	CPU ratio increases significantly with problem size.
R (processing-time range)	6.72	0.0110	Larger R values slightly increase the overall mean CPU ratio.
Algorithm × Partial-job size	2.32×10^{37}	0.0000	Growth patterns differ across algorithms depending on problem size.

This implies that computational growth patterns differ substantially among algorithms. BB3 shows exponential increases in CPU ratio with problem size, while BB4 maintains moderate scalability and robustness, making it the most practical option for large-scale re-entrant flowshop scheduling problems. Overall, BB1 demonstrates the most consistent and efficient performance, while BB4 provides a balanced trade-off between robustness and computational cost.

7. Conclusion

This study focuses on a re-entrant permutation flowshop operating with two machines, where the performance criterion is the minimization of total weighted flowtime. To address this optimization problem, the authors developed a family of branch-and-bound-type solution procedures. A set of branch-and-bound algorithms was developed, incorporating dominance rules, a lower-bounding scheme, and a heuristic for generating initial solutions. Experimental results demonstrated that optimal solutions can be obtained for instances up to 22 partial jobs within practical CPU times, confirming the effectiveness of the proposed approach. While this study assumes a single facility for each process without setup or buffer restrictions, real manufacturing systems often include parallel machines, setup times, and limited buffers. Based on the results of this study, a semiconductor FAB of Company S in Korea is currently applying, testing, and observing the proposed approach through a pilot implementation in a production segment that involves re-entrant processing. Future research will extend the model to account for such practical constraints and validate the algorithm using actual industrial data.

Acknowledgements

This work was supported by Kyonggi University Research Grant 2025.

References

[1] BongJoo Jeong, and Yeong-Dae Kim, "Minimizing Total Tardiness in a Two-Machine Re-Entrant Flowshop with Sequence-Dependent Setup Times," *Computers & Operations Research*, vol. 47, pp. 72-80, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[2] JC-H Pan, and J-S.Chen, "Minimizing Makespan in Re-Entrant Permutation Flow-Shops," *Journal of the Operational Research Society*, vol. 54, no. 6, pp. 642-653, 2003. [CrossRef] [Google Scholar] [Publisher Link]

[3] Maximilian von Aspern et al., "Flow Shops with Reentry: The Total Weighted Completion Time Objective," *arXiv Preprint*, pp. 1-19, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[4] Jingwen Yuan et al., "Scheduling Reentrant Flow Shops: Reinforcement Learning Guided Meta-Heuristics," *IET Collaborative Intelligent Manufacturing*, vol. 7, no. 1, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[5] Hongtao Tang et al., "Hybrid Flowshop Scheduling Problems with Missing and Reentrant Operations Considering Process Scheduling and Energy Consumption," *Sustainability*, vol. 15, no. 10, pp. 1-19, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[6] Stephen C. Graves et al., "Scheduling of Re-Entrant Flow Shops," *Journal of Operations Management*, vol. 3, no. 4, pp. 197-207, 1983. [CrossRef] [Google Scholar] [Publisher Link]

[7] F. Della Croce, V. Narayan, and R. Tadei, "The Two-Machine Total Completion Time Flow Shop Problem," *European Journal of Operational Research*, vol. 90, no. 2, pp. 227-237, 1996. [CrossRef] [Google Scholar] [Publisher Link]

[8] J.A. Hoogeveen, and S.L. van de Velde, "Stronger Lagrangean Bounds by Use of Slack Variables: Application to Machine Scheduling Problems," *Mathematical Programming*, vol. 70, no. 1-3, pp. 173-190, 1995. [CrossRef] [Google Scholar] [Publisher Link]

[9] Edward Ignall, and Linus Schrage, "Application of the Branch-and-Bound Technique to Some Flow Shop Problems," *Operations Research*, vol. 13, no. 3, pp. 400-412, 1965. [CrossRef] [Google Scholar] [Publisher Link]

[10] Ebru Demirkol, and Reha Uzsoy, "Decomposition Methods for Reentrant Flow Shops with Sequence Dependent Setup Times," *Journal of Scheduling*, vol. 3, no. 3, pp. 115-177, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[11] Michael R. Garey, and David S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completenessv San Francisco*, W.H. Freeman and Company, 1979. [Google Scholar]

[12] Jeffrey Schaller, "Note on Minimizing Total Tardiness in a Two-Machine Flowshop," *Computers & Operations Research*, vol. 32, no. 12, pp. 3273-3281, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[13] Kenneth R. Baker, *Introduction to Sequencing and Scheduling*, New York: John Wiley & Sons, 1974. [Google Scholar]

[14] Dragoslav S. Mitrinović, *Analytic Inequalities*, Berlin: Springer, 1970. [Google Scholar] [Publisher Link]

[15] Muhammad Nawaz, E Emory Enscore Jr, and Inyong Ham, "A Heuristic Algorithm for the M-Machine, N-Job Flow-Shop Scheduling Problem," *Omega*, vol. 11, no. 1, pp. 91-95, 1983. [CrossRef] [Google Scholar] [Publisher Link]