APEM journal
Advances in Production Engineering & Management
Volume 18 | Number 2 | June 2023 | pp 152–162
https://doi.org/10.14743/apem2023.2.463
ISSN 1854-6250
Journal home: apem-journal.org
Original scientific paper

Genetic algorithm-based approach for makespan minimization in a flow shop with queue time limits and skipping jobs

Han, J.H.[a], **Lee, J.Y.**[b,*]

[a]Department of Industrial Engineering, Pusan National University, Republic of Korea
[b]Division of Business Administration & Accounting, Kangwon National University, Republic of Korea

ABSTRACT

This study investigates a flow shop scheduling problem with queue time limits and skipping jobs, which are common scheduling requirements for semiconductor and printed circuit board manufacturing systems. These manufacturing systems involve the most complex processes, which are strictly controlled and constrained to manufacture high-quality products and satisfy dynamic customer orders. Further, queue times between consecutive stages are limited. Given that the queue times are limited, jobs must begin the next step within the maximum queue time after the jobs in the previous step are completed. In the considered flow shop, several jobs can skip the first step, referred to as skipping jobs. Skipping jobs exist because of multiple types of products processed in the same flow shop. For the considered flow shop, this paper proposes a mathematical programming formulation and a genetic algorithm to minimize the makespan. The GA demonstrated its strengths through comprehensive computational experiments, demonstrating its effectiveness and efficiency. As the problem size increased, the GA's performance improved noticeably, while maintaining acceptable computation times for real-world fab facilities. We also validated its performance in various scenarios involving queue time limits and skipping jobs, to further emphasize its capabilities.

ARTICLE INFO

Keywords:
Scheduling;
Flow shop;
Makespan;
Queue time limits;
Skipping jobs;
Optimization;
Modeling;
Genetic algorithm

**Corresponding author:*
Jy.lee@kangwon.ac.kr
(Lee, J.Y.)

Article history:
Received 9 October 2022
Revised 4 July 2023
Accepted 6 July 2023

1. Introduction

This study focused on the scheduling problem in a three-machine flow shop with queue time limits and skipping jobs. Such scheduling problems are found in semiconductor and printed circuit board (PCB) manufacturing systems, which play key roles in the electronics industry as the major components of most electronic products. Such manufacturing systems generally fabricate multiple types of products. Therefore, meeting customer demand in terms of quality and improving the throughput rate are the most important objectives. Queue time limits and skipping jobs are related to product quality and multiple types of products, respectively. In this paper, we investigate the scheduling problem in the considered flow shop with the objective of minimizing makespan subject to queue time limits and skipping jobs.

Semiconductor wafer fabrication is generally a time-critical production environment owing to many highly reactive chemical processes. Thus, the queue times of work-in-process wafers in a fab

are strictly managed and constrained to satisfy wafer quality requirements [1]. These constraints are common in wafer lot scheduling, especially at Korean semiconductor manufacturing companies that enforce queue time limits across multiple process steps to enhance overall quality.

Generally, semiconductor manufacturing involves limited queue times for two main reasons. Firstly, it is crucial to maintain the cleanliness of the wafer surfaces during the waiting period before proceeding to the next stage. Prolonged exposure of wafer surfaces to air heightens the risk of contamination by impurities, leading to significant quality issues. Secondly, it is essential to preserve the chemical efficacy treated in the previous step while awaiting the subsequent stage. Therefore, chemically treated wafers must proceed to the next step within a specified timeframe known as the limited queue time. Failure to initiate the subsequent processing step within this queue time limit may require reworking or discarding the wafers, especially in cases of severe contamination [2]. Consequently, this causes lower productivity and business losses. As semiconductor processes become more complex, the number of processes with queue time limits increases, increasing their importance to semiconductor manufacturing. In addition to semiconductor manufacturing systems, queue time limits can also be found in many manufacturing systems for batteries, food, steel, and crude oil [33].

In a semiconductor manufacturing fab, several types of wafer products are produced simultaneously. Their main process flows are similar, but some specific steps are different depending on the product types. Thus, flow shops modeled in a semiconductor fab are designed to process multiple product types, some skipping unnecessary steps. For example, if the first step is cleaning or metrology, some lots that do not need this step skip it and go straight to the second step. For such jobs, the queue time limits are set between the second and third steps. With an increase in the number of semiconductor products, this feature has become more common. Flow shops with skipping jobs are also found in PCB manufacturing systems for multi-variety low-volume production. Additionally, skipping jobs are found in various manufacturing systems, including pharmaceuticals, molten iron, stainless steel, and flour [1, 27, 30].

A typical flow shop consists of a series of machines or workstations arranged in sequential order and processes every job through the machines in that order. If the orders of all jobs are the same across all machines, this is referred to as a permutation schedule. In general, when job i completes on machine k but job j is still unfinished on machine $(k + 1)$, job i can wait until job j completes. However, in a flow shop with queue time limits, job i cannot wait for a long time. In other words, after job i completes on machine k, its subsequent operation on machine $(k + 1)$ must begin within a specified limited queue time. On the other hand, Most flow shop scheduling studies have assumed that every job processes at all stages, which is currently not valid because of the increased customization and diversification of products [3].

Using three-field notation in [4] to define theoretically the considered scheduling problem, it is $F_3|max\text{-}wait, skip|C_{max}$. The first field means a flow shop with three machines. The second field represents problem characteristics, i.e., queue time limits and skipping jobs, respectively. The last field shows the makespan as the objective function to be minimized. As mentioned in [5], a two-machine flow shop with queue time limits is NP-hard, so this scheduling problem is also NP-hard.

We provide a mixed-integer programming (MIP) formulation to describe the scheduling problem clearly. The MIP can be solved using a commercial optimization solver. However, obtaining optimal solutions using the solver may require a significant computation time or it cannot be achieved within an acceptable time, given that the problem is NP-hard. Therefore, we propose a genetic algorithm to rapidly obtain adequate and effective solutions. The genetic algorithm was evaluated under different cases of computational experiments, and it showed highly effective and efficient performance.

The structure of this paper is as follows. Section 2 outlines the assumptions and notation used to describe the proposed algorithms and the mixed-integer programming formulation for the scheduling problem. Section 3 introduces heuristic algorithms, while their evaluation is discussed in Section 4. Lastly, Section 5 provides the study's conclusions and a summary of the findings.

2. Literature survey

With reference to the available literature, most flow shop scheduling studies with queue time limits (QTL) have focused on two-machine problems. Yang *et al*. [5] proved that minimizing the makespan in a two-machine flow shop with QTL is NP-hard and proposed a branch-and-bound (B&B) algorithm. Additionally, advanced lower bounds and dominance properties were developed in [6, 7], while a constructive heuristic algorithm was proposed in [8]. Furthermore, An *et al*. [9] included not only QTL but also sequence-dependent setup times and developed heuristic algorithms and a B&B algorithm, while Lee [10] suggested a genetic algorithm to minimize the total tardiness for the same flow shop. In addition, Dhouib *et al*. [11] proposed a MIP and a simulated annealing algorithm to hierarchically minimize the number of tardy jobs and makespan for multi-machine flow shops with QTL, whereas Kim and Lee [12] introduced a three-machine flow shop with overlapping QTL. To minimize the total tardiness, Hamdi and Loukil [13] suggested a lower bound scheme based on Lagrangian relaxation and a heuristic algorithm, and Joo *et al*. [14] performed simulation experiments with a list scheduling rule. Furthermore, flow shop problems with QTL have been considered in several recent studies [15-23].

In the relevant literature, scheduling problems with skipping jobs have not been extensively investigated in comparison with other types of flow shops. However, a feature of skipping jobs has recently received significant research attention, given that multiple product types with different specifications are produced in the same manufacturing fabrication. Rajendran and Ziegler [24] examined the performance of dispatching rules and a heuristic algorithm to minimize total flow time. Saravanan *et al*. [25] and Dios *et al*. [26] proposed a simulated annealing algorithm and various heuristic algorithms, respectively, to minimize the makespan, whereas, Saravanan *et al*. [27] suggested a genetic algorithm to minimize the mean tardiness. On the other hand, Tseng *et al*. [28] proposed a heuristic to change a given permutation schedule to an improved non-permutation schedule for minimizing the makespan, while Li *et al*. [29] proposed a multi-objective artificial bee colony algorithm with respect to flowtime, earliness, and tardiness in molten iron processing. Recently, in the context of Industry 4.0 production, Rossit *et al*. [30] introduced a flow shop with skipping jobs.

In the literature, QTL and skipping jobs in flow shop scheduling have been separately studied, and their incorporation of them has received limited attention, with only a few studies exploring them together. Notably, for no-wait flow shop problems with skipping jobs, Glass *et al*. [31] proposed and analyzed several heuristic algorithms for minimizing the makespan in a two-machine, while Smutnicki *et al*. [32] devised a method to determine the cyclic schedule with minimal cycle time. For the general queue time limits, Ruiz *et al*. [33] investigated a comprehensive flow shop problem encompassing various features, such as queue time limits, skipping jobs, and the inclusion of machine eligibility, machine release dates, precedence constraints, and sequence-dependent setup times, and they evaluated simple dispatching rules and heuristic algorithms. Additionally, Yu *et al*. [1] focused on a two-machine flow shop with QTL and skipping jobs and they analyzed mathematical properties, explored the reduction of the search space, and developed efficient approximation algorithms for minimizing queue time variations. Furthermore, Han and Lee [34] dealt with minimizing total tardiness in a flow shop scheduling problem that involved queue time limits and skipping jobs.

In the existing literature, there is a notable gap in flow shop research regarding $F_3|max\text{-}wait, skip|C_{max}$. It is important to highlight that the study conducted by Han and Lee [34] focused on minimizing total tardiness in a flow shop scheduling problem specifically within the semiconductor foundry business. This industry places significant emphasis on satisfying customer delivery dates. However, in our present study, we shift our attention to the memory business, where factors such as equipment utilization and throughput take precedence. Thus, the objective of this study is to minimize the makespan. It is worth mentioning that this study represents the first attempt to tackle this particular problem, and the outcomes obtained can serve as a foundational basis for future research and advancements in this field.

3. Problem description

In this section, we describe the considered scheduling problem with assumptions, notation, and MIP formulation. The considered flow shop has three stages (k = 1, 2, 3), and each stage has a machine m_k. All jobs (i = 1, 2,..., n) are given and are available to start at time zero, and all information for making schedules are provided in advance. That is, for job i, processing time PT_{ik} on machine k and queue time limit QT_{ik} between m_k and $m_{(k+1)}$ are already known. Among the jobs, there are jobs starting at m_1, called *normal jobs*, and skipping jobs starting at m_2. For this scheduling problem, the objective is to minimize the makespan which is equal to the completion time of the last processed job on m_3.

To address this problem, we restrict a solution space only to permutation schedules. In particular, given a specific job sequence, all jobs follow that sequence and are processed on machines in the same order. Although an optimal schedule may not be included in the permutation schedules, many studies with QTL have assumed permutation schedules. This is because lots in semiconductor manufacturing fabs are typically processed in permutation schedules for traceability, manageability, and flexibility in material handling [7]. The followings are additional assumptions made in this study.

Pre-emption is not permitted, meaning a job cannot be interrupted once it starts processing. Each job can be processed on only one machine at a time, and each machine can process only one job at a time. The queue time limits must be satisfied without failure. However, the number of jobs queueing between machines is not limited.

Additionally, the following symbols in Table 1 are used to describe the MIP and the proposed algorithms.

Table 1 The symbols used in the algorithms

Symbol	Meaning
h	index of the position in a sequence, h = 1,..., n
$[h]$	index of a job placed at the h-th position in a sequence
x_{ih}	= 1 if job i is placed at the h-th position in a sequence; otherwise, 0
$ST_{[h]k}$	the time at which the h-th job starts on m_k
$CT_{[h]k}$	the time at which the h-th job completes on m_k

We provide equations for the completion times of all jobs in a given sequence. For the first scheduled job (h = 1), Eqs. 1 to 3 calculate the completion times on the three machines, respectively. Obviously, the first job is processed immediately without waiting.

$$CT_{[1]1} = PT_{[1]1} \tag{1}$$

$$CT_{[1]2} = CT_{[1]1} + PT_{[1]2} \tag{2}$$

$$CT_{[1]3} = CT_{[1]2} + PT_{[1]3} \tag{3}$$

From the second position onwards (h = 2,..., n), the calculations for the completion times on the first two machines (m_1 and m_2) differ between the normal jobs starting at m_1 and skipping jobs starting at m_2. Eqs. 4 and 5 computes the completion times of the normal jobs on m_1 and m_2, while Eqs. 6 and 7 computes those of the skipping jobs. The completion times of the final machine are computed using Eq. 8.

$$CT_{[h]1} = max\{CT_{[h-1]1} + PT_{[h]1}, CT_{[h-1]2} - QT_{[h]1}, CT_{[h-1]3} - QT_{[h]2} - PT_{[h]2} - QT_{[h]1}\} \tag{4}$$

$$CT_{[h]2} = max\{max\{CT_{[h]1}, CT_{[h-1]2}\} + PT_{[h]2}, CT_{[h-1]3} - QT_{[h]2}\} \tag{5}$$

$$CT_{[h]1} = CT_{[h-1]1} \tag{6}$$

$$CT_{[h]2} = max\{CT_{[h-1]2} + PT_{[h]2}, CT_{[h-1]3} - QT_{[h]2}\} \tag{7}$$

$$CT_{[h]3} = max\{CT_{[h]2}, CT_{[h-1]3}\} + PT_{[h]3} \tag{8}$$

The next is the MIP formulation to clearly define the considered scheduling problem and to serve a benchmark schedule using an optimization solver. Herein, normal jobs and skipping jobs are not distinguished. In particular, it is assumed that skipping jobs have zero processing time at m1 and sufficiently long queue times between m_1 and m_2, to render the queue time limits non-significant, i.e., $PT_{i1} = 0$ and $QT_{i1} = \infty$ for skipping jobs.

$$\text{[P] Minimize } CT_{[n]3} \tag{9}$$

$$\text{subject to } \sum_i x_{ih} = 1, \quad 1 \le h \le n \tag{10}$$

$$\sum_h x_{ih} = 1, \quad \forall i \tag{11}$$

$$ST_{[h]k} + \sum_i PT_{ik} x_{ih} \le ST_{[h+1]k}, \quad 1 \le h \le n-1, 1 \le k \le 3 \tag{12}$$

$$ST_{[h]k} + \sum_i PT_{ik} x_{ih} \le ST_{[h]k+1}, \quad 1 \le h \le n, 1 \le k \le 2 \tag{13}$$

$$ST_{[h]k} + \sum_i (PT_{ik} + QT_{ik}) x_{ih} \ge ST_{[h]k+1}, \quad 1 \le h \le n, 1 \le k \le 2 \tag{14}$$

$$ST_{[h]3} + \sum_i PT_{i3} x_{ih} \le CT_{[h]3}, \quad 1 \le h \le n \tag{15}$$

$$ST_{[h]k} \ge 0, \quad 1 \le h \le n, 1 \le k \le 3 \tag{16}$$

$$x_{ih} \in \{0, 1\}, \quad \forall i, 1 \le h \le n \tag{17}$$

Eq. 9 represents the objective function's makespan, which is the maximum completion time of the jobs. To ensure permutation schedules without preemption, Eqs. 10 and 11 stipulate that each position can accommodate only one job and each job can and must only occupy one position in the sequence, respectively. Eq. 12 defines the relationship between the start times of two consecutive jobs, while Eq. 13 defines the relationship between the start times on two consecutive machines. To satisfy queue time limits, Eq. 14 guarantees that each job's subsequent operation starts within the specified queue time. Eq. 15 calculates the completion times of the jobs, and Eqs. 16 and 17 establish the decision variables' domains.

4. Used Methods: Genetic algorithms

Because the problem considered is NP-hard, an effective and efficient heuristic algorithm is required to quickly obtain adequate solutions. Thus, we propose a genetic algorithm (GA), which is a nature-inspired evolutionary optimization method. The GA is one of the most commonly used methods because it has low mathematical requirements and is highly flexible in application [35-38]. The following subsections describe the proposed GA design.

4.1 Solution representation and fitness

The solutions to the problem considered are expressed in permutation schedules, which are sequences of job indices that can be directly represented in the chromosome structure of the GA. Each solution has an objective function value, which is the makespan of the schedule and should be minimized. Thus, the makespan to be minimized was used for fitness evaluation.

4.2 Initial population

To generate an initial population, a constructive heuristic algorithm referred to as the NEH (Nawaz, Enscore, and Ham) algorithm was used. Since the NEH algorithm has shown good performance in many flow shop scheduling problems, but it does not guarantee an optimal solution. Thus, it has been commonly used to find an initial solution for further improvement using other optimization techniques.

The NEH is based on the concept of inserting jobs into a sequence iteratively to minimize the makespan. Initially, it calculates the total processing time for each job by summing up the processing times across all machines. Next, the jobs are sorted in decreasing order based on their

total processing times. For each job in the sorted order, it is inserted at all possible positions in the sequence, and the resulting makespan is evaluated. Then, the position that yields the lowest makespan is chosen, and the job is inserted at that position. Herein, for population diversity, we used shortest processing time (SPT)-based list scheduling rules to obtain different solutions. The remaining initial population was randomly generated.

- SPT_1 with p_{i1}
- SPT_2 with p_{i2}
- SPT_3 with p_{i3}
- SPT_4 with $p_{i2} + p_{i3}$
- SPT_5 with $p_{i1} + p_{i2} + p_{i3}$

4.3 Selection

The selection operation randomly selects chromosomes for the next generation. The chromosomes are selected with a selection probability proportional to their fitness values. This operation is analogous to the survival of the fittest in the theory of evolution and is referred to as the *roulette* method in the GA. Note that the performance of the roulette method has been demonstrated in recent related studies [14, 21, 22]. Because the objective function should be minimized, the inverse f_i of the objective function value for each chromosome i in the population is computed, and the selection probability $prob_i$ of each chromosome is obtained by $prob_i = f_i / \sum_j f_j$. The next population is composed based on the selection probabilities of the chromosomes.

4.4 Crossover

The crossover operation generates offspring by exchanging information about the selected parents. Various methods can be used for crossover, such as the one-point crossover and two-point crossover methods [14]. Hosseinabadi *et al*. [39] and Hasançebi and Erbatur [40] evaluated various crossover methods for genetic algorithms and found that a one-point crossover demonstrated high performance in scheduling problems. Lee [14] reported that a one-point crossover demonstrated relatively high performance in a flow shop with queue time limits. Based on the existing research results, a one-point crossover was applied in this study.

For each chromosome in the current population, if a randomly generated number (between 0 and 1) is less than the probability of crossover p_c, the chromosome is copied into a pool. They are then randomly paired and referred to as *parents*. For the parents ($Parent_1$ and $Parent_2$), a one-point crossover is applied to generate two offsprings ($Offspring_1$ and $Offspring_2$) using the following procedure:

Step 0: Randomly select a crossover point from among the parent genes.
Step 1: $Offspring_1$ inherits genes up to the crossover point within $Parent_1$.
Step 2: Except for the genes that $Offspring_1$ already contains, $Offspring_1$ inherits genes from $Parent_2$ in the order in which they appear in $Parent_2$.
Step 3: $Offspring_2$ is generated in the same manner in Steps 1 and 2.

4.5 Mutation

The mutation is an operation that prevents premature convergence and maintains the diversity within a population. This operation partially mutates several genes in a chromosome with low probability.

For each chromosome in the population, if a random number (between 0 and 1) is less than the mutation probability, the chromosome is mutated once. The mutated ones are also included in the population. In this study, two common methods were used, namely, *insertion* and *exchange*, with the same probability. Insertion selects a job at random and inserts it into a randomly selected position, whereas exchange interchanges two randomly selected jobs

4.6 Local search

We used a local search technique to improve the performance of the GA. To prevent a rapid increase in the calculation time owing to the local search, it was only applicable to 10 % of the so-

lutions in a population at each generation. In this study, *insertion* and *exchange* methods were used. In particular, *insertion* or *exchange* with the same probability was applied 3*n* times to the solution, where *n* is the number of jobs. If the new solution was superior to the current solution, the current solution was replaced.

4.7 Entire procedure

The proposed GA terminates when the maximum number of generations is reached; the overall flowchart is shown in Fig. 1.

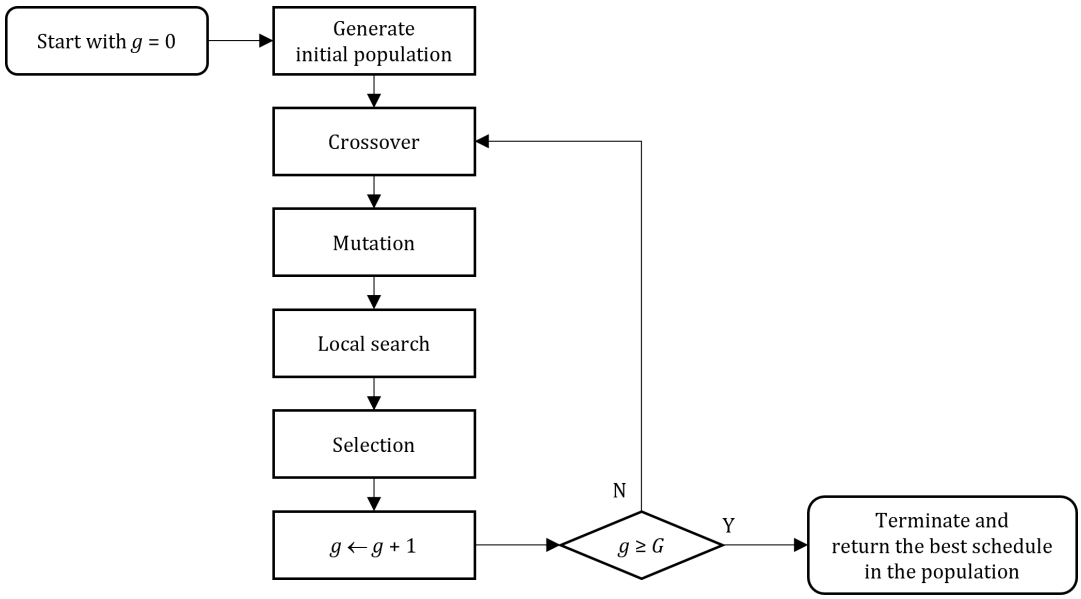

Fig. 1 Flowchart of the proposed GA

5. Computational experiments

To assess the performance of the proposed algorithms, we performed computational experiments using instances generated as follows. The experiments encompassed various levels of the number of jobs, n = (10, 20, 30, 40, 50, 100, 150, and 200). The processing times p were generated from discrete uniform distributions within the range of [1, 50]. Additionally, the queue times were generated from three different distributions with a range of [1, w], where w = (30, 50, and 70). Regarding the proportion of skipping jobs, we considered three levels of the proportion λ of skipping jobs, where λ = (0.3, 0.5, and 0.7), i.e., (0.3n, 0.5n, and 0.7n). For each combination of (n, w, and λ), ten instances were generated, resulting in a comprehensive set of experimental data.

We coded the proposed algorithms using the Java programming language and conducted experiments on a personal computer equipped with an Intel Core i7-8700 CPU running at 3.2GHz. To solve the MIP formulation, we used CPLEX 12.10, a commercial solver, with a maximum CPU time of 1,000 seconds to prevent excessive computation time. As for the proposed GA, it terminated once the maximum number of generations was reached and returned the best solution found within the population. For each instance, the GA was run independently 30 times, and a mode value from the 30 runs was used for the evaluation.

Before evaluating the GA, we performed experimental calibration to improve its performance. In the calibration, we assumed n = 100, λ = 0.5, and w from [1, 50], which were the medium levels of the considered problem instances.

The GA contained four parameter types: population size S, number of generations G, probability of crossover p_c, and probability of mutation p_m. If the population size is excessively small, diverse solutions are not available. On the other hand, if the population size is excessively large, the time required for crossover and local search increases. Thus, excessive time is required to obtain a solution. Therefore, it is necessary to select an appropriate population size related to the number of jobs to be considered. In this study, the size of the population was set to

($\rho \cdot n$) and a preliminary experiment was conducted for ρ = 1, 2,..., 5. Considering the trade-off between the algorithm performance and calculation time, ρ was set to 4.

The proposed GA used the maximum number of generations G as the termination condition. Therefore, an appropriate number of generations was required. If G is excessively large, a significant amount of time is required to obtain a solution, whereas, if G is excessively small, it is highly probable that a solution close to the optimal solution will not be obtained. We tested {500, 1000, 1500, 2000, 2500, and 3000} for G. Thereafter, 1000 was set as the appropriate number of generations for the preliminary experiments. In particular, preliminary experiments were conducted on {0.3, 0.5, 0.7, and 0.9} for p_c and {0.05, 0.1, 0.2, 0.3, and 0.4} for p_m, which were set to 0.7 and 0.2, respectively. Note that because of the extensive range of potential combinations among the four types of parameters examined and the resulting substantial amount of data, this paper does not include the results of the preliminary tests.

First, we tested the MIP performance using CPLEX. Table 2 lists the average CPU times of CPLEX with a computation time limit (1,000 s). As seen from the table, CPLEX shows a tendency to work well when the number of jobs skipping the first stage increased and the queue time limits were loose, whereas CPU times increased in the opposite case. The level of difficulty can be estimated based on the problem situation. The CPU time increased significantly as the number of jobs increased. When n = 50, 100, 150, and 200, CLPEX did not obtain an optimal solution for 10, 40, 79, and 89 instances (out of 90) within the time limit.

Next, we evaluated the effectiveness of the simple heuristics used to generate the initial population. Table 3 presents the average percentage error (PE), which is defined as $100 \times (obj_A - obj_{CPLEX}) / obj_{CPLEX}$ for Algorithm A, where N in the names of the last six algorithms represents the NEH algorithm. As seen from the table, the six list scheduling rules provided solutions with large errors compared with those from CPLEX. However, the proposed NEH algorithm demonstrated high performance. Every NEH algorithm obtained schedules with errors of approximately 2 % or less. Moreover, for instances with n = 150 and 200, schedules superior to those obtained from CPLEX were obtained. In addition, all NEH algorithms required less than 1 s. Thus, it was confirmed that an effective and efficient heuristic algorithm should be developed.

In addition, we evaluated the performance of the proposed GA. In addition, to verify the effectiveness of the local search, a GA without a local search (GA_{l-}) was tested. Table 4 presents the performance in comparison with CPLEX, where NI and CPUT indicate the number of instances in which the GA and GA_{l-} obtained a superior schedule to that obtained from CPLEX and the average computation time of the GAs, respectively. The relative deviation index (RDI) is defined as $(obj_A - obj_{min}) / (obj_{max} - obj_{min})$, where obj_A is the objective value for Algorithm A, and obj_{min} and obj_{max} are the minimum and maximum objective values, respectively. As shown in the table, the effectiveness of the GA was demonstrated more clearly as the number of jobs increased. For all instances with n = 150 and 200, the GA obtained a solution that was superior to that obtained by CPLEX. The performance of the GA was improved using local search. Although the calculation time increased owing to the inclusion of the local search, the average CPUT was 87 s for n = 200, which is practicable.

Table 2 CPU times of CPLEX

λ	w	n=10	20	30	40	50	100	150	200
0.3	30	0.08	231.62	378.25	277.15	688.18	984.75	1000	1000
	50	0.06	0.53	3.95	144.09	313.47	547.89	1000	1000
	70	0.05	0.14	0.31	3.31	39.78	571	940.44	1000
0.5	30	0.06	64.17	4.9	169.36	140.77	818.78	1000	1000
	50	0.04	0.25	3.32	21.56	59.21	432.6	933.92	1000
	70	0.03	0.13	0.89	12.95	12.35	529.04	953.12	1000
0.7	30	0.06	0.46	2.83	111.82	44.83	411.79	852.79	972.94
	50	0.03	0.13	0.89	12.95	12.35	529.04	953.12	1000
	70	0.04	0.09	0.19	10.08	2.62	254.99	804.98	1000

Finally, we evaluated the performance of the GA concerning different problem parameters (λ and w) for instances with n = 100. The results are summarized in Table 5, which presents the

average PE values. When both parameters decreased, the PE values also decreased. This indicates that the GA was effective when the limited queue times were strict and the number of jobs that skipped the first stage decreased. Problems with these features are more difficult than others; thus, CPLEX failed to find an optimal or adequate solution to these problems. It should be noted that if λ and w are zero, the problem transforms into a three-machine no-wait flow shop and into a two-machine flow shop problem if both parameters increase.

6. Conclusion

This study investigated a flow shop scheduling problem specifically focusing on the incorporation of queue time limits and skipping jobs, which are critical requirements in semiconductor manufacturing. To address this problem, this study proposed a MIP formulation and GA. Computational experiments were carried out to assess the performance of both the MIP and GA, with a particular emphasis on evaluating the effectiveness and efficiency of the GA. The results demonstrated that as the problem size increased, the GA exhibited improved performance and maintained an acceptable computation time within a real fab facility. Additionally, the GA's performance was verified under various conditions of queue time limits and skipping jobs.

However, it should be noted that in certain conditions during the computational experiments, the GA failed to achieve an optimal solution. Thus, future endeavors should be directed toward enhancing the performance of the GA. Potential approaches for improvement include exploring alternative heuristic algorithms such as state-of-the-art metaheuristics, genetic programming-based algorithms, or machine learning-based algorithms. Furthermore, it is worth considering the integration of bi-objective optimization approaches, incorporating measures based on both makespan and due dates, as the semiconductor business often follows an order-based manufacturing paradigm. Additionally, future research can explore different types of queue time limits,

Table 3 Percent errors of heuristic algorithm

	n=10	20	30	40	50	100	150	200	average
SPT_1	19.23	22.32	21.97	23.99	24.72	25.83	24.58	22.73	23.17
SPT_2	22.41	23.11	22.41	22.99	23.76	24.73	23.64	21.55	23.08
SPT_3	30.00	29.28	27.43	27.21	27.68	27.53	25.59	23.19	27.24
SPT_4	21.91	20.44	18.86	17.84	18.54	17.62	15.92	14.06	18.15
SPT_5	17.05	17.02	14.94	15.54	15.97	16.36	15.12	13.30	15.66
LPT	25.60	22.53	19.93	19.10	18.80	17.72	16.03	14.10	19.23
SPT_1N	1.65	1.23	1.13	1.18	1.19	0.89	-0.54	-2.14	0.57
SPT_2N	2.12	2.00	1.49	1.69	1.48	0.72	-0.76	-2.36	0.80
SPT_3N	2.33	2.03	1.55	1.49	1.37	0.72	-0.86	-2.41	0.78
SPT_4N	1.91	1.33	0.99	0.75	0.64	-0.21	-1.81	-3.35	0.03
SPT_5N	1.50	1.06	0.72	0.81	0.60	-0.08	-1.60	-3.31	-0.04
LPTN	1.31	0.74	0.43	0.35	0.15	-0.43	-1.98	-3.57	-0.37

Table 4 Performance of the proposed GA

n	PE (%)		NI		CPUT (s)		RDI		
	GA	GA_{I-}	*GA*	GA_{I-}	*GA*	GA_{I-}	*GA*	GA_{I-}	*CPLEX*
10	0.144	0.281	75	54	0.0	0.0	0.134	0.393	0.000
20	0.106	0.243	81	47	0.2	0.1	0.076	0.476	0.000
30	0.039	0.144	82	54	0.5	0.1	0.052	0.400	0.000
40	0.065	0.129	75	48	1.0	0.3	0.121	0.472	0.022
50	-0.050	0.010	82	57	1.8	0.4	0.068	0.393	0.108
100	-0.554	-0.532	89	76	12.1	1.8	0.011	0.181	0.444
150	-2.071	-2.054	90	90	38.0	4.6	0.000	0.010	0.878
200	-3.612	-3.608	90	90	87.3	9.3	0.000	0.001	0.989

Table 5 Percent errors of the proposed GA on different parameters (n = 100)

w	λ			average
	30	50	70	
0.3	-2.377	-0.499	-0.360	-1.079
0.5	-1.185	-0.261	-0.043	-0.496
0.7	-0.097	-0.157	-0.004	-0.086
average	-1.220	-0.306	-0.136	-0.554

including overlapping queue time limits as presented in [11] and generalized skipping jobs that possess the ability to skip any stage. Lastly, extending the investigation of this problem to hybrid flow shops with multiple machines at each stage could be a valuable direction for future research.

Acknowledgment

This work was supported by the Technology Development Program(RS-2022-00140527) funded by the Ministry of SMEs and Startups(MSS, Korea)

References

[1] Yu, T.-S., Kim, H.-J., Lee, T.-E. (2017). Minimization of waiting time variation in a generalized two-machine flowshop with waiting time constraints and skipping jobs, *IEEE Transactions on Semiconductor Manufacturing*, Vol. 30, No. 2, 155-165, doi: 10.1109/Tsm.2017.2662231.

[2] Lee, J.-H., Zhao, C., Li, J.S., Papadopoulos, C.T. (2018). Analysis, design, and control of Bernoulli production lines with waiting time constraints, *Journal of Manufacturing Systems*, Vol. 46, 208-220, doi: 10.1016/j.jmsy.2018.01.001.

[3] Henneberg, M., Neufeld, J.S. (2016). A constructive algorithm and a simulated annealing approach for solving flowshop problems with missing operations, *International Journal of Production Research*, Vol. 54, No. 12, 3534-3550, doi: 10.1080/00207543.2015.1082670.

[4] Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy-Kan, A.H.G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey, *Annals of Discrete Mathematics*, Vol. 5, 287-326, doi: 10.1016/S0167-5060(08)70356-X.

[5] Yang, D.-L., Chern, M.-S. (1995). A two-machine flowshop sequencing problem with limited waiting time constraints, *Computers & Industrial Engineering*, Vol. 28, No. 1, 63-70, doi: 10.1016/0360-8352(94)00026-J.

[6] Bouquard, J.-L., Lenté, C. (2006). Two-machine flow shop scheduling problems with minimal and maximal delays, *4OR*, Vol. 4, 15-28, doi: 10.1007/s10288-005-0069-7.

[7] Joo, B.-J., 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*, Vol. 60, No. 4, 572-582, doi: 10.1057/palgrave.jors.2602598.

[8] Wang, B.L., Huang, K., Li, T. (2018). Permutation flowshop scheduling with time lag constraints and makespan criterion, *Computers & Industrial Engineering*, Vol. 120, 1-14, doi: 10.1016/j.cie.2018.04.021.

[9] An, Y.-J., Kim, Y.-D., Choi, S.-W. (2016). Minimizing makespan in a two-machine flowshop with a limited waiting time constraint and sequence-dependent setup times, *Computers & Operations Research*, Vol. 71, 127-136, doi: 10.1016/j.cor.2016.01.017.

[10] Dhouib, E., Teghem, J., Loukil, T. (2013). Lexicographic optimization of a permutation flow shop scheduling problem with time lag constraints, *International Transactions in Operational Research*, Vol. 20, No. 2, 213-232, doi: 10.1111/j.1475-3995.2012.00876.x.

[11] Lee, J.-Y. (2020). A genetic algorithm for a two-machine flowshop with a limited waiting time constraint and sequence-dependent setup times, *Mathematical Problems in Engineering*, Vol. 2020, Article ID 8833645, doi: 10.1155/2020/8833645.

[12] Kim, H.-J., Lee, J.-H. (2019). Three-machine flow shop scheduling with overlapping waiting time constraints, *Computers & Operations Research*, Vol. 101, 93-102, doi: 10.1016/j.cor.2018.06.009.

[13] Hamdi, I., Loukil, T. (2015). Minimizing total tardiness in the permutation flowshop scheduling problem with minimal and maximal time lags, *Operational Research*, Vol. 15, No. 1, 95-114, doi: 10.1007/s12351-014-0166-5.

[14] Joo, B.J., Kim, Y.D., Bang, J.Y. (2009). A scheduling algorithm for workstations with limited waiting time constraints in a semiconductor wafer fabrication facility, *Journal of the Korean Institute of Industrial Engineers*, Vol. 35, No. 4, 266-279.

[15] Zhou, N., Wu, M., Zhou, J. (2018). Research on power battery formation production scheduling problem with limited waiting time constraints, In: *Proceedings of 2018 10th International Conference on Communication Software and Networks,* Chengdu, China, 497-501, doi: 10.1109/ICCSN.2018.8488247.

[16] Hamdi, I., Toumi, S. (2019). MILP models and valid inequalities for the two-machine permutation flowshop scheduling problem with minimal time lags, *Journal of Industrial Engineering International*, Vol. 15, 223-229, doi: 10.1007/s40092-019-00331-1.

[17] Samarghandi, H. (2019). Minimizing the makespan in a flow shop environment under minimum and maximum time-lag constraints, *Computers & Industrial Engineering*, Vol. 136, 614-634, doi: 10.1016/j.cie.2019.07.048.

[18] Li, Y., Dai, Z. (2020). A two-stage flow-shop scheduling problem with incompatible job families and limited waiting time, *Engineering Optimization*, Vol. 52, No. 3, 484-506, doi: 10.1080/0305215X.2019.1593974.

[19] Maassen, K., Perez-Gonzalez, P., Gunther, L.C. (2020). Relationship between common objective functions, idle time and waiting time in permutation flow shop scheduling, *Computers & Operations Research*, Vol. 121, Article No. 104965, doi: 10.1016/j.cor.2020.104965.

[20] Jeong, B.J., Han, J.-H., Lee, J.-Y. (2021). Metaheuristics for a flow shop scheduling problem with urgent jobs and limited waiting times, *Algorithms*, Vol. 14, No. 11, Article No. 323, doi: 10.3390/a14110323.

[21] Lee, J.H., Kim, H.J. (2021). 3-machine flow shop scheduling with overlapping waiting time constrains to minimize total completion time, *Journal of the Korean Society of Supply Chain Management*, Vol. 21, No. 3, 1-12, doi: 10.25052/KSCM.2021.12.21.3.1.
[22] Lee, J.H., Yu, T.S., Park, K.S. (2021). Scheduling of flow shop with overlapping waiting time constraints using genetic algorithm, *Journal of the Korean Institute of Industrial Engineers*, Vol. 47, No. 1, 34-44, doi: 10.7232/JKIIE.2021.47.1.034.
[23] Ištoković, D., Perinić, M., Borić, A. (2021). Determining the minimum waiting times in a hybrid flow shop using simulation-optimization approach, *Tehnički Vjesnik – Technical Gazette*, Vol. 28, No. 2, 568-575, doi: 10.17559/TV-20210216132702.
[24] Rajendran, C., Ziegler, H. (2001). A performance analysis of dispatching rules and a heuristic in static flowshops with missing operations of jobs, *European Journal of Operational Research*, Vol. 131, No. 3, 622-634, doi: 10.1016/S0377-2217(00)00105-3.
[25] Saravanan, M., Sridhar, S., Harikannan, N. (2014). Optimization of realistic multi-stage hybrid flow shop scheduling problems with missing operations using meta-heuristics, *International Journal of Engineering Technology*, Vol. 6, No. 1, 484-496.
[26] Dios, M., Fernandez-Viagas, V., Framinan, J.M. (2018). Efficient heuristics for the hybrid flow shop scheduling problem with missing operations, *Computers & Industrial Engineering*, Vol. 115, 88-99, doi: 10.1016/j.cie.2017.10.034.
[27] Saravanan, M., Sridhar, S., Harikannan, N. (2016). Minimization of mean tardiness in hybrid flow shop with missing operations using genetic algorithm, *Journal of Advanced Manufacturing Systems*, Vol. 15, No. 2, 43-55, doi: 10.1142/S0219686716500050.
[28] Tseng, C.-T., Liao, C.-J., Liao, T.-X. (2008). A note on two-stage hybrid flowshop scheduling with missing operations, *Computers & Industrial Engineering*, Vol. 54, No. 3, 695-704, doi: 10.1016/j.cie.2007.09.005.
[29] Li, J.-Q., Pan, Q.-K., Duan, P.-Y. (2016). An improved artificial bee colony algorithm for solving hybrid flexible flowshop with dynamic operation skipping, *IEEE Transactions on Cybernetics*, Vol. 46, No. 6, 1311-1324, doi: 10.1109/Tcyb.2015.2444383.
[30] Rossit, D.A., Toncovich, A., Rossit, D.G., Nesmachnow, S. (2021). Solving a flow shop scheduling problem with missing operations in an Industry 4.0 production environment, *Journal of Project Management*, Vol. 6, No. 1, 33-44, doi: 10.5267/j.jpm.2020.10.001.
[31] Glass, C.A., Gupta, J.N.D., Potts, C.N. (1999). Two-machine no-wait flow shop scheduling with missing operations, *Mathematics of Operations Research*, Vol. 24, No. 4, 911-924, doi: 10.1287/moor.24.4.911.
[32] Smutnicki, C., Pempera, J., Bocewicz, G., Banaszak, Z. (2022). Cyclic flow-shop scheduling with no-wait constraints and missing operations, *European Journal of Operational Research*, Vol. 302, No. 1, 39-49, doi: 10.1016/j.ejor.2021.12.049.
[33] Ruiz, R., Şerifoğlu, F.S., Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems, *Computers & Operations Research*, Vol. 35, No. 4, 1151-1175, doi: 10.1016/j.cor.2006.07.014.
[34] Han, J.-H., Lee, J.-Y. (2022). Scheduling for a flow shop with waiting time constraints and missing operations in semiconductor manufacturing, *Engineering Optimization*, doi: 10.1080/0305215X.2022.2124406.
[35] Xu, W., Sun, H.Y., Awaga, A.L., Yan, Y., Cui, Y.J. (2022). Optimization approaches for solving production scheduling problem: A brief overview and a case study for hybrid flow shop using genetic algorithms, *Advances in Production Engineering & Management*, Vol. 17, No. 1, 45-56, doi: 10.14743/apem2022.1.420.
[36] Shi, D.L., Zhang, B.B., Li, Y. (2020). A multi-objective flexible job-shop scheduling model based on fuzzy theory and immune genetic algorithm, *International Journal of Simulation Modelling*, Vol. 19, No. 1, 123-133, doi: 10.2507/IJSIMM19-1-CO1.
[37] Boudjemline, A., Chaudhry, I.A., Rafique, A.F., Elbadawi, I.A., Aichouni, M., Boujelbene, M. (2022). Multi-objective flexible job shop scheduling using genetic algorithms, *Tehnički Vjesnik – Technical Gazette*, Vol. 29, No. 5, 1706-1713, doi. 10.17559/TV-20211022164333.
[38] Chen, D., Zhao, X.R. (2021). Production management of hybrid flow shop based on genetic algorithm, *International Journal of Simulation Modelling*, Vol. 20, No. 3, 571-582, doi: 10.2507/IJSIMM20-3-CO12.
[39] Hosseinabadi, A.A.R., Vahidi, J., Saemi, B., Sangaiah, A.K., Elhoseny, M. (2019). Extended genetic algorithm for solving open-shop scheduling problem, *Soft Computing*, Vol. 23, No. 13, 5099-5116, doi: 10.1007/s00500-018-3177-y.
[40] Hasançebi, O., Erbatur, F. (2000). Evaluation of crossover techniques in genetic algorithm based optimum structural design, *Computers & Structures*, Vol. 78, No. 1-3, 435-448, doi: 10.1016/S0045-7949(00)00089-4.