processes

MDPI

Article

Scheduling Jobs with a Limited Waiting Time Constraint on a Hybrid Flowshop

Sang-Oh Shim [1], BongJoo Jeong [2,*], June-Yong Bang [3] and JeongMin Park [1]

[1] Department of Business Administration, Hanbat National University, Daejeon 305-719, Republic of Korea; soshim@hanbat.ac.kr (S.-O.S.); jmpark@hanbat.ac.kr (J.P.)
[2] Department of Business Administration, Kongju National University, Gongju-si 314-701, Republic of Korea
[3] Department of Industrial and Management Engineering, Sungkyul University, Anyang-si 430-742, Republic of Korea; jybang@sungkyul.ac.kr
* Correspondence: jbj@kongju.ac.kr; Tel.: +82-41-850-8432

Abstract: In this paper, we address a two-stage hybrid flowshop scheduling problem with identical parallel machines in each stage. The problem assumes that the queue (Q)-time for each job, which represents the waiting time to be processed in the current stage, must be limited to a predetermined threshold due to quality concerns for the final product. This problem is motivated by one that occurs in the real field, especially in the diffusion workstation of a semiconductor fabrication. Our objective is to minimize the makespan of the jobs while considering product quality. To achieve this goal, we formulated mathematical programming, developed two dominance properties for this problem, and proposed three heuristics with the suggested dominance properties to solve the considered problem. We conducted simulation experiments to evaluate the performance of the proposed approaches using randomly generated problem instances that are created to closely resemble real production scenarios, and the results demonstrate their superiority over existing methods. Furthermore, we applied the proposed methods in a real-world setting within the semiconductor fabrication industry, where they have exhibited better performance compared to the dispatching rules commonly used in practical applications. These findings validate the effectiveness and applicability of our proposed methodologies in real-world scenarios.

Keywords: scheduling; hybrid flowshop; limited waiting time; semiconductor fabrication; diffusion workstation

Citation: Shim, S.-O.; Jeong, B.; Bang, J.-Y.; Park, J. Scheduling Jobs with a Limited Waiting Time Constraint on a Hybrid Flowshop. *Processes* **2023**, *11*, 1846. https://doi.org/10.3390/pr11061846

Academic Editors: Dapeng Zhang, Qiangda Yang, Yuwen You and Xu Ji

Received: 30 April 2023
Revised: 19 May 2023
Accepted: 16 June 2023
Published: 19 June 2023

1. Introduction

As surveyed by Linn and Zhang [1], Ribas et al. [2], and Tosun et al. [3], scheduling problems in a hybrid flowshop (also known as a flexible flowshop) environment are commonly encountered in various manufacturing settings. Typically, in the hybrid flowshop, there are multiple stages of machines, and each stage consists of one or more identical parallel machines. The jobs need to be processed sequentially through the stages; once a job completes processing in one stage it moves to the next stage without any idle time. The hybrid flowshop scheduling problem is known for its complexity and has been widely studied in the context of production planning and scheduling. In the hybrid flowshop, each job follows a predetermined sequence of stages, but the number of machines available for processing can vary at each stage. On the other hand, in a flexible flowshop the number of machines remains constant across the stages.

This research focuses on two-stage hybrid flowshop scheduling problems with queue (Q)-time-sensitive constraints. These constraints require the operations of each job in the second stage to start within a predetermined time after completing the previous (first stage) operation of the same job.

There are numerous instances of such problems in various manufacturing systems, particularly in semiconductor manufacturing fabrications. One prominent example is the

diffusion workstation within a 300 mm (12″)-sized wafer fabrication process. In these workstations, there are typically two or three stages (such as diffusion chamber, cleaning, and etching) with parallel machines that may be identical or nonidentical, and the wafers undergo processing on a machine at each stage. With advancements in semiconductor technology, the distance between circuits on the surface of semiconductor wafers has significantly reduced. Therefore, the control of particles becomes crucial as they can have a detrimental impact on wafer quality, given that their sizes are larger than the distances between the circuits. Despite strict control measures for particles within the air of semiconductor manufacturing fabrications, there are instances where particles can adhere to the wafers.

One of the methods employed to prevent particles from adhering to the surfaces of wafers is to initiate the next process within a specific time limit after the completion of the current process. The longer the products wait for the subsequent operation, the higher the likelihood of particle adhesion. Consequently, it is crucial to process the product for the next operation promptly after it completes the current process. In the case of the diffusion process, particles tend to adhere more easily compared to other workstations. To address this issue, many manufacturing managers determine a time duration within which they prefer the wafers to be processed for the next operation after the completion of the current process. This time duration, empirically determined by operators or established using statistical methods, is commonly referred to as the limited waiting time or queue (Q)-time.

The main objective of this study is to address the two-stage hybrid flowshop scheduling problem with limited waiting time constraints, aiming to minimize the makespan of jobs. Extensive research has been conducted to tackle the challenges associated with hybrid flowshop scheduling. Several researchers, including Brah and Hunsucker [4], Rajendran and Chaudhuri [5], Gupta et al. [6], Moursli and Pochet [7], Azizoglu et al. [8], and Lee and Kim [9], have proposed a branch-and-bound algorithm to solve the hybrid (flexible) flowshop scheduling problem. Their goal was to obtain an optimal solution considering various performance measures such as makespan, total completion time, and tardiness. These researchers have suggested the use of dominance properties, as well as lower and upper bounds, to improve the efficiency of the branch-and-bound algorithms.

While optimal solution algorithms offer theoretical guarantees, their practicality is limited due to computational complexity. Consequently, research efforts have focused on developing effective heuristics to obtain near-optimal solutions for the hybrid flowshop scheduling problem in real-world applications. Dugardin et al. [10] and Gomez-Gasquet et al. [11] proposed metaheuristics such as genetic algorithms, taboo search, and simulated annealing, while considering mathematical programming techniques. Eskandari and Hosseinzadeh [12] used heuristics based on dispatching rules, while Gupta [13] and Fernandez-Viagas et al. [14] developed constructive heuristics based on a combination of a mathematical model and a local search algorithm.

For the specific case of the two-stage hybrid flowshop scheduling problem with arbitrary waiting times (inserted idle time) and the objective of minimizing the makespan, researchers have proposed branch-and-bound (B&B) algorithms. Yang and Chern [15] and Bonquard and Lente [16] suggested B&B algorithms and developed dominance properties, lower bound schemes, and heuristics to obtain upper bounds. Furthermore, Fondervelle et al. [17] identified several optimal solution properties, and Bouquard et al. [18] utilized Max-Plus algebra to present additional properties for the problem when $k \geq 2$.

In the context of the two-stage hybrid flowshop scheduling problem with limited waiting times and a configuration of one batch machine in stage one and one single machine in stage two, Su [19] devised a heuristic algorithm. Subsequently, other studies further explored flowshop scheduling, including Ruiz and Vázquez-Rodríguez [20], Behnamian and Fatemi Ghomi [21], Ying and Lin [22], Öztop et al. [23], and Lin et al. [24].

Extensive research has been conducted on hybrid flowshop scheduling problems, whereas relatively fewer studies have addressed scheduling problems with waiting time constraints in the context of flowshop scheduling. A well-known special case arises when

assuming infinite waiting times for all jobs at each stage, which corresponds to the classical two-machine flowshop scheduling problem. Johnson's algorithm [25] can be employed optimally to minimize the makespan of jobs in this case. Similarly, if all waiting times are required to be zero, the problem reduces to the no-wait two-machine flowshop scheduling problem, for which an optimal solution can be obtained in polynomial time [26].

In the previous studies, Yang and Chern [15] showed the NP hardness of the problem when considering arbitrary waiting times. More recently, An et al. [27] proposed a branch-and-bound algorithm for the two-machine flowshop problem with limited waiting time and sequence-dependent setup times. Lee [28] and Jeong et al. [29] developed meta-heuristic algorithms to address two-machine flowshop scheduling problems with limited waiting time constraints. It is worth noting that most of the research on limited waiting time constraints focuses on the two-machine flowshop scheduling problem. Chung et al. [30] investigates a two-stage hybrid flowshop problem with a single batch processing machine in the first stage and a single machine in the second stage with the limited waiting time constraint. However, to the best of our knowledge, the specific problem considered in this study has not yet been addressed in the existing literature.

In this paper, we propose two dominance properties and three heuristics to address two-stage hybrid flowshop scheduling problems with limited waiting time constraints, where each stage consists of identical parallel machines. The primary objective is to minimize the makespan of the scheduling problem. The structure of this research is as follows: In the subsequent section, we provide a clear description of the problem under consideration and formulate it as a mathematical programming model. Additionally, we explore and establish several properties related to optimal solutions in this study. To evaluate the performance of the proposed algorithms, we conducted computational experiments using randomly generated problem instances. The results obtained from these experiments are presented and analyzed in the results section. Finally, we conclude the research by summarizing the key findings and contributions. Furthermore, we suggest potential avenues for future research in this field.

2. Problem Description

Typically, in K-stage hybrid flowshop scheduling problems, there exist n independent jobs comprising k operations, and they need to be processed on a machine at each stage. In the k-th stage (where $k = 1, \ldots, K$) there are m_k (≥ 1) parallel machines that can be identical, uniform, or unrelated. The problem at hand involves n independent jobs that need to go through two stages. These stages are equipped with M_1 and M_2 identical parallel machines, respectively. Several assumptions are made for this problem: (i) All jobs are available at time zero; (ii) Each job has given processing times; (iii) No setup time is required; (iv) Each machine can process only one job at a time and no machine can handle multiple jobs simultaneously; (v) Preemption is not allowed.

One key constraint in this problem is the limited waiting time, also referred to as the queue (Q)-time. After completing the first operation of a job, the second operation must commence within a predetermined time. The waiting time, denoted as w_i, may vary for different jobs, indicating the time period a job must wait before the second operation begins.

To describe the problem more clearly, we use the following notation.

p_{ik} processing time of job i in stage k ($i = 1, \ldots, n$ and $k = 1, 2$)

w_i (predetermined) limited waiting time (Q-time) of job i that can be allowed between stage one and two

C_{jtk} completion time of t-th positioned job on a j-th machine in stage k ($t = 1, \ldots, n$)

C_{max} maximum completion time of all jobs; i.e., makespan

S_{jt2} start time of the job at stage two (which is completed in the t-th position on j-th machine at stage one)

$Z_{ijtk} = 1$ if job i is in the t-th position on j-th machine at k-th stage, otherwise 0.

M_k number of identical parallel machines at k-th stage

M big M
Using the notation, a mathematical formulation is given as follows.
Objective function:

$$\text{Min } Z = C_{\max} \tag{1}$$

Constraints:

$$\sum_{t=1}^{n} \sum_{j=1}^{M_k} Z_{ijtk} = 1 \quad \forall i = 1, \ldots, n; \ \forall k = 1, \ 2 \tag{2}$$

$$\sum_{t=1}^{n} \sum_{i=1}^{n} Z_{ijtk} \leq n \quad \forall j = 1, \ldots, M_k; \ \forall k = 1, \ 2 \tag{3}$$

$$\sum_{i=1}^{n} Z_{ijtk} \leq 1 \quad \forall j = 1, \ldots, M_k; \ \forall k = 1, \ 2; \forall t = 1, \ldots, n \tag{4}$$

$$C_{jt-1k} + \sum_{i=1}^{n} p_{ik} Z_{ijtk} \leq C_{jtk} \quad \forall i, \forall t = 1, \ldots, n; \ \forall j = 1, \ldots, M_k; \ \forall k = 1, \ 2 \tag{5}$$

$$S_{i2} \geq C_{jt1} Z_{ijt1} \quad \forall i = 1, \ldots, n, \ \forall j = 1, \ldots, M_1; \ \forall t = 1, \ldots, n \tag{6}$$

$$S_{i2} \leq \left(C_{jt1} Z_{ijt1} + w_i\right) + \left(1 - Z_{ijt1}\right) \mathrm{M} \ \forall i = 1, \ldots, n, \ \forall j = 1, \ldots, M_1; \ \forall t = 1, \ldots, n \tag{7}$$

$$C_{jt2} \geq (S_{i2} + p_{i2}) Z_{ijt1} \forall i = 1, \ldots, n, \ \forall j = 1, \ldots, M_1; \ \forall t = 1, \ldots, n \tag{8}$$

$$C_{\max} \geq C_{jt2} \quad \forall j = 1, \ldots, M_2; \ \forall t = 1, \ldots, n \tag{9}$$

$$C_{j0k} = 0 \quad \forall j = 1, \ldots, M_k; \ \forall k = 1, \ 2 \tag{10}$$

The objective function (1) expresses the goal of the problem, which is to minimize the makespan, representing the total time required to complete all jobs. Constraint (2) ensures that each job is scheduled exactly once in the problem. Constraint (3) guarantees that the number of jobs assigned to a machine does not exceed the total number of jobs. Constraint (4) states that, at any given time in the schedule horizon, only one job can be processed on a machine. The completion time of a job is defined in constraint (5), indicating that it is the sum of the processing times on all machines. Constraint (6) ensures that the start time of a job in stage two is greater than or equal to the completion time of the same job in stage one. Constraint (7) represent the limited waiting time constraint, specifying that the waiting time of a job between stages should be less than or equal to a predetermined time. Additionally, the completion time of job i on the second stage is represented as constraint (8). The makespan is defined in constraint (9), representing the maximum completion time among all jobs. Finally, constraint (10) sets the initial completion time to zero. This mixed-integer programming formulation is used to solve a small-sized problem and serves as a benchmark to compare the performance of the suggested methods proposed in this paper.

3. Solution Approach

3.1. Dominance Properties

Although a mathematical programming formulation is provided, obtaining optimal solutions becomes increasingly difficult and time-consuming as the number of jobs and machines increases. Therefore, heuristic algorithms are considered to quickly respond to dynamic scheduling requirements by utilizing dominance properties.

Permutation schedules are commonly used to solve flowshop scheduling problems, where the same order of jobs is considered for each stage. However, in hybrid flowshop

scheduling problems, permutation schedules do not guarantee optimal solutions. Therefore, this study focuses on nonpermutation schedules.

In this study, intentional idle time, which refers to deliberately inserting idle time, is not considered since it would increase the makespan. However, there may be unavoidable idle time in two cases: (1) When an operation in the second stage cannot immediately start after the completion of an operation in the first stage due to pending operations in the second stage; (2) When the associated first operation needs to be shifted because it is impossible for an operation in the second stage to start within the limited waiting time.

This paper proposes several properties, starting with property one, which is based on the results of Azizoglu and Kirca [31]. Their work demonstrates the property of the maximum completion time on each machine in parallel machine scheduling problems with the objective of minimizing total regular costs.

Property 1. If the number of machines in stage one is greater than or equal to that of machines in stage two and $\max_i p_{i1} \leq \min_i p_{i2}$, then there exists an optimal schedule in which the waiting times of all jobs in stage two are zero and the completion time of each machine in stage two cannot exceed $\max_i p_{i1} + \{\sum p_{i2} + (M_2 - 1) \max_i p_{i2}\}/M_2$.

Proof. The second stage is bottleneck since $M_2 \leq M_1$ and $\max_i p_{i1} \leq \min_i p_{i2}$. That is, whenever the first operation of any job is completed on a machine, this job can be processed on a machine in stage two without any intentional idle time (the idle time between jobs in stage one can occur since $M_2 \leq M_1$ and $\max_i p_{i1} \leq \min_i p_{i2}$). Consequently, there is no idle time between two successive jobs in stage two in any optimal schedules and there are only M_1 idle times to wait before starting the second operations for which the first operations are completed. According to the results of Azizoglu and Kirca [31], $\{\sum p_{i2} + (M_2 - 1) \max_i p_{i2}\}/M_2$ can be the upper bound of total processing times of jobs in stage two. Additionally, there can be idle time in stage two as much as $\max_i p_{i1}$.This completes the proof. □

Similar to property one, a property is developed specifically for the case when stage one becomes the bottleneck. These two properties allow us to obtain an initial upper bound for the makespan under each condition.

Property 2. If the number of machines in stage two is greater than or equal to that of machines in stage one and $\max_i p_{i2} \leq \min_i p_{i1}$, then there exists an optimal schedule in which the waiting times of all jobs in stage two are zero and the completion time of each machine in stage two cannot exceed $\max_i p_{i2} + \{\sum p_{i1} + (M_1 - 1) \max_i p_{i1}\}/M_1$.

Proof. In this case, the second stage is bottleneck since $M_1 \leq M_2$ and $\max_i p_{i2} \leq \min_i p_{i1}$. That is, whenever the first operation of any job is completed on a machine, this job can be processed on a machine in stage two without any intentional idle time (the idle time between jobs in stage two can occur since $M_1 \leq M_2$ and $\max_i p_{i2} \leq \min_i p_{i1}$). Consequently, there is no idle time between two successive jobs in stage one in any optimal schedule. Similarly, $\{\sum p_{i1} + (M_1 - 1) \max_i p_{i1}\}/M_1$ can be the upper bound of the total processing times of jobs in stage one. The completion time of the last scheduled job is $\max_i p_{i2}$ in stage two. □

By using the suggested properties, heuristic algorithms are developed as follows. Let $C_{[i]k}$ be the completion time of the i-th completed job in stage k (if more than two jobs are completed at the same time on different machines, then the job on the machine with the lowest index has the smallest $[i]$). Then, each $C_{[i]k}$ can be defined using the following equations:

$$\begin{aligned} &C_{[i]1} = \max\{C_{[i-1]1} + p_{[i]1}, C_{[i-1]2} - w_{[i]}\} && \text{for all } i = 1, \ldots, n; \\ &C_{[i]2} = \max\{C_{[i]1}, C_{[i-1]2}\} + p_{[i]2} && \text{for all } i = 1, \ldots, n; \\ &C_{[0]1} = C_{[0]2} = 0. \end{aligned} \tag{11}$$

3.2. Heuristic Algorithms

In this paper, three heuristic algorithms, H1, H2, and H3, are suggested. The first heuristic algorithm, H1, uses Johnson's rule [25]. It is assumed that this problem is a two-machine flowshop scheduling problem, and the sequence of jobs is obtained. Then, by using the list scheduling method and considering the limited waiting time, the schedule is constructed one by one. The second algorithm, H2, is based on the algorithm devised by Gilmore and Gomory [26]. Suppose that the second operations should start as soon as the first operation is completed, and this problem is considered to be a conventional two-machine flowshop scheduling problem. Then, this problem, F_2/no-wait/C_{max}, can be solved optimally. After obtaining a sequence, construct a list schedule using the concept of the backward scheduling method suggested by Kim [32]. Notice that a schedule is obtained by scheduling the second operation first in order of the reversed sequence for a reversed time frame. That is, the start time and completion time of jobs in the original problem are the completion and start time in the reversed one, respectively. After obtaining a reversed schedule, this time frame is adjusted to the original one (it is easy to understand by imagining a mirror). The last method, H3, is similar to H1 except it utilizes a sequence obtained by using the modified largest processing time rule for list scheduling. The following are detailed procedures of each heuristic.

H1

Step 1. Obtain a sequence by using Johnson's rule assuming that there is only one machine in each stage; that is, consider this problem as a conventional two-machine flowshop scheduling problem. Let U be the set of the obtained sequence.
Step 2. Select the first (leftmost) job in the set U. Say job x.
Step 3. Construct a schedule in stage one using the list scheduling method, in which a job is assigned to the machine with the earliest start time (ties are broken by selecting a machine with the lowest index). Update the start time $S_{[x]1}$ and completion time $C_{[x]1}$ of the job x.
Step 4. Construct a schedule in stage two by selecting a machine with the minimum completion time, of which the start time is greater than $C_{[x]1}$. Update the start time $S_{[x]2}$ and completion time $C_{[x]2}$. If the second operation of a job cannot start in the limited waiting time, i.e., $S_{[x]2} - C_{[x]1} \geq w_x$, the completion time of the first operation associated with the job is adjusted to start in the limited waiting time, new $C_{[x]1} = S_{[x]2} - w_x$. Update the other variable $S_{[x]1}$.
Step 5. If all jobs are scheduled then stop; otherwise, eliminate job x in the set U and go to step 2.

H2

Step 1. Obtain a sequence by using the algorithm suggested by Gilmore and Gomory [26] assuming that there is only one machine in each stage and waiting time in the second stage is not allowed; that is, consider this problem as a conventional two-machine flowshop scheduling problem with no-wait constraints (all wi's are zero). Let U be the set of the obtained sequence.
Step 2. In the reversed time frame problem, let $S'_{[i]k}$ and $C'_{[i]k}$ be a start time and a completion time of a job in stage k, respectively. Notice that if $k = 1$ in this notation, it implies that $k = 2$ in the original problem.
Step 3. Select the last (rightmost) job in the set U'. Say job x.
Step 4. Construct a schedule in stage one (stage two in the original problem) by using the list scheduling method, in which a job is assigned to the machine with the earliest time (ties are broken by selecting a machine with the lowest index). Update the start time $S'_{[x]1}$ and completion time $C'_{[x]1}$ of the job x.
Step 5. Construct a schedule in stage two (stage one in the original problem) by selecting a machine with the smallest completion time, of which the start time is greater than $C'_{[x]1}$. Update the start time $S'_{[x]2}$ and completion time $C'_{[x]2}$. If the second operation of a job cannot start immediately, i.e., $S'_{[x]2} - C'_{[x]1} \geq 0$, the completion

time of the first operation associated with the job is adjusted to the start time of the second operation, i.e., $C'_{[x]1} = S'_{[x]2}$. Update another variable $S'_{[x]1}$.

Step 6. If all jobs are scheduled then stop and compute C'max; otherwise, eliminate job x in the set U and go to step 3.

Step 7. Restore the reversed problem to the original one. That is, without any shift in jobs, the start time and completion time of each job is recalculated reversely. In this manner, the last scheduled job in the reversed problem is the one scheduled first in the original problem, in which the start time and completion time in stage one are 0 and p[1]2, respectively. In a similar way, reschedule other jobs and update original variables.

Step 8. In the original problem, sort the completion times of the jobs in stage one. Let set U be the set of the obtained order of jobs.

Step 9. Select a leftmost job, i.e., a job with the smallest $Ci1$, in the set U.

Step 10. If the completion time can be improved by shifting this job on the same machine, reschedule this job as well as succeeding jobs in each stage and update the variables. Otherwise, eliminate this job in the set U and repeat step 9 and 10 until set U is none.

H3

Step 1. Sort jobs in nonincreasing order of the processing times of the jobs in stage one. Let U be the set of the obtained order.

Step 2. Select the first (leftmost) job in the set U. Say job x.

Step 3. Construct a schedule in stage one by using the list scheduling method, in which a job is assigned to the machine with the earliest start time (ties are broken by selecting a machine with the lowest index). Update the start time $S_{[x]1}$ and completion time $C_{[x]1}$ of the job x.

Step 4. Construct a schedule in stage two by selecting a machine with the minimum completion time, of which the start time is greater than $C_{[x]1}$. Update the start time $S_{[x]2}$ and completion time $C_{[x]2}$. If the second operation of a job cannot start in the limited waiting time, i.e., $S_{[x]2} - C_{[x]1} \geq w_x$, the completion time of the first operation associated with the job is adjusted to start in the limited waiting time, new $C_{[x]1} = S_{[x]2} - w_x$. Update the other variables $S_{[x]1}$.

Step 5. If all jobs are scheduled then stop; otherwise, eliminate job x in the set U and go to step 2.

4. Computational Experiments

To compare the performance of the suggested heuristics, computational experiments were conducted using randomly generated problems. The heuristics were implemented using the commercial software ezDFS, which is based on the C++ programming language and widely used in manufacturing execution systems (MES) in real manufacturing environments. The software provides a graphical user interface (GUI) tool that allows users to develop methodologies and generate test data sets easily.

The computational tests were performed on a personal computer equipped with an i7-6700 processor operating at a clock speed of 3.4 GHz. Since the scheduling problem studied in this research is motivated by real situations in the diffusion process of semiconductor fabrication, randomly generated problem instances were created to closely resemble real production scenarios. The number of jobs were set to 20, 40, 60, 80, and 100, and the number of machines in each stage were set to 5, 10, and 15, respectively.

In the context of semiconductor fabrication, a lot refers to a unit that encompasses 25 wafers and represents a job within the scope of this study. The problem under consideration arises in a diffusion workstation, which consists of two stages. The first stage comprises two to four diffusion chamber machines capable of simultaneously processing two to six jobs in batches. The second stage consists of five to ten clean or etching machines that operate in a sequential manner. Typically, in real-world scenarios, there are 10 to 30 jobs waiting to be processed at each stage. The diffusion workstation scheduler,

integrated within the MES, performs scheduling operations to assign jobs to machines at regular intervals, usually every 5 to 10 min. Consequently, scheduling outcomes, including job–machine assignments and corresponding start/finish times, are obtained periodically.

The processing times of jobs were generated using a discrete uniform distribution ranging from one to fifty, while the waiting times were generated from one to one hundred. For the experimental setup, a total of 450 problem instances were generated by performing 10 runs for each combination mentioned above.

We have employed the relative deviation index (RDI) and the number of best solutions (NBS) as performance measures in this study. The RDI of each heuristic algorithm, denoted as H, is computed using the formula (X–B)/(W–B), where X represents the makespan obtained using algorithm H, B is the best makespan achieved among all heuristic algorithms, and W is the worst makespan among all heuristic algorithms for a given problem. The RDI ranges between 0 and 1 and provides an indication of how well each heuristic algorithm performs in comparison to others. A smaller RDI for algorithm H indicates superior performance compared to other algorithms for the problem at hand. Additionally, the NBS reflects the number of cases in which each algorithm achieves the minimum makespan, serving as a measure of algorithm superiority.

Table 1 presents the overall results, including the average and standard deviation of the RDIs for each heuristic, the number of best solutions, and the average and deviation of CPU times. Overall, the results demonstrate that H3 exhibits superior performance in terms of both RDI and NBS when compared to other methods. However, there is no significant difference observed in CPU times to obtain solutions since most solutions are achieved in real time. Thus, all heuristic algorithms can be effectively applied to real-world production scenarios while considering the Q-time constraint.

Table 1. The overall results of the test.

Heuristics	RDI [1]		NBS [2]	CPU Time	
H1	0.58	(0.42)	57	0.12	(0.36)
H2	0.53	(0.67)	105	0.08	(0.51)
H3	0.33	(0.59)	363	0.14	(0.75)

[1] Average and standard deviation (in parenthesis) of relative deviation index; [2] Number of simulation runs (out of 450) for which each heuristic found the best solutions.

To examine the impact of various factors, such as the number of jobs and machines in each stage, on the relative performance of the algorithms, an analysis of variance (ANOVA) was conducted on the solution values, specifically the makespan of jobs, for each heuristic method. The results of the ANOVA are presented in Table 2. The ANOVA table reveals that the performance of the heuristics was significantly influenced by the choice of heuristic, with a significance level of 0.01. This confirms our earlier findings that H3 outperforms the other algorithms. Interestingly, the analysis indicates that the relative performance of the algorithms was not affected by other factors, such as the number of jobs and the number of machines in each stage. This suggests that certain algorithms, such as H3, consistently demonstrate superior performance regardless of the specific characteristics of the problem.

In addition to the previous tests, a separate set of experiments were conducted to assess the absolute performance of the superior algorithm, H3, by comparing its solutions with optimal solutions and lower bounds. As there is no efficient optimal solution algorithm available for the problem, optimal solutions were obtained using mixed integer programming (MIP) for small-sized problems, as suggested in Section 2. The MIP was implemented in C++ with a MIP solver function from the commercial MES software ezDFS. For this test, the number of machines in each stage was set to two and three and the number of jobs was set to five and six. The remaining data were generated following the same procedure as described earlier. The results of the test are summarized in Table 3, which presents the percentage gaps between the solutions obtained using the heuristic algorithm and the optimal solutions (lower bounds), as well as the computation times required to

obtain the optimal solution (lower bound). Since the computation time to obtain a solution was very short (less than 0.05 s), it is not shown in the table. From the table, it can be observed that the suggested algorithm, H3, consistently produced very good solutions and, in some cases, even found optimal solutions. The small percentage gaps indicate the effectiveness of H3 in finding high-quality solutions, considering the complexity of the problem.

Table 2. Analysis of variance for the performance of each heuristic.

Source of Variation	Degree of Freedom	Sum of Squared Errors	Mean Squared Errors	F
Heuristics	2	209	104.5	13.8 †
Number of jobs	4	65	16.3	2.1
NM1 [1]	2	58	29.0	3.8
NM2 [2]	2	63	31.5	4.2
NM1 × NM2	4	93	23.3	3.1
Error	1335	10,096	7.6	
Total	1349	10,584		

[1] Number of machines in stage 1; [2] Number of machines in stage 2; † Statistically different at the significant level of 0.01.

Table 3. Percentage gap between solutions of the algorithm and optimal solutions.

NM1	NM2	Number of Jobs	Percentage Gaps [1]			CPU Time [2]	
2	2	5	0.70 [2]	(1.55)	8	121.2	(549.1)
	3	6	2.26	(4.18)	7	369.8	(1032.6)
3	2	5	3.31	(4.81)	6	898.7	(804.5)
	3	6	1.33	(2.98)	8	1542.1	(2408.1)

[1] Average and standard deviation (in parenthesis) of the percentage gaps of heuristic solutions from the optimal solutions and the number of problems for which the algorithm found optimal solutions out of 10 test problems; [2] Average and standard deviation (in parenthesis) of the CPU time required to find an optimal solution.

For larger-sized problems, the solutions obtained using the heuristic algorithm were compared with a lower bound based on the branch-and-bound algorithm suggested by Brah and Hunsucker [4]. This lower bound assumes that the waiting time of each job is infinite, i.e., $w_i = \infty$. Thus, it does not require jobs to be processed within their waiting time after completion in stage one. The branch-and-bound algorithm by Brah and Hunsucker [4] addresses the *k*-stage hybrid flowshop scheduling problem with identical parallel machines in each stage, aiming to minimize the makespan of jobs.

To obtain the lower bounds, the number of machines in each stage were set to four and five, and the number of jobs were set to eight and ten. The remaining data followed the same generation process as described earlier. In Table 4, the results indicate that there is no significant difference between the lower bounds and the solutions obtained using the heuristic algorithm. This suggests that the solutions from the suggested heuristic algorithm can be considered to be relatively good, as they closely approach the lower bounds obtained through the branch-and-bound algorithm by Brah and Hunsucker [4].

Table 4. Percentage gap between solutions of the algorithm and lower bounds.

NM1	NM2	Number of Jobs	Percentage Gaps [1]			CPU Time [2]	
4	4	8	7.26	(5.00)	2	10.3	(5.4)
	5	10	7.67	(7.08)	3	36.1	(40.7)
5	4	8	9.51	(10.46)	3	49.6	(69.3)
	5	10	10.19	(8.05)	2	91.5	(95.4)

[1] Average and standard deviation (in parenthesis) of the percentage gaps of heuristic solutions from the lower bounds and the number of problems for which solutions of the algorithm are equal to the lower bounds of 10 test problems; [2] Average and standard deviation (in parenthesis) of the CPU time required to find lower bounds.

5. Conclusions

This paper focused on the two-stage hybrid flowshop scheduling problem with identical parallel machines in each stage, aiming to minimize the makespan of jobs. The problem considered waiting times for jobs to be processed in the second stage, with a constraint that the waiting time cannot exceed a predetermined limit due to job deterioration. Three heuristic algorithms were tested, based on Johnson's rule [25], Gilmore and Gomory's algorithm [26], and a modified largest processing time rule.

The proposed algorithms demonstrated good performance and, among them, H3 outperformed the others by providing very good solutions that were nearly optimal, all within a very short computation time. This paper is particularly significant as it considers both the quality constraint and the makespan of jobs in the scheduling algorithms. The suggested algorithm holds practical value for real manufacturing systems as it is motivated by real-world scheduling problems and has been extended in various ways. Therefore, although our research is not purely theoretical it is argued that it has practical applications in real-world scenarios.

Furthermore, this paper opens avenues for further research. One possible direction is the development of more sophisticated algorithms based on the suggested algorithm, specifically for cases with more than two stages in the hybrid flowshop problem or for objectives other than makespan. Additionally, it is important to explore optimal solution properties and devise more efficient algorithms that leverage these properties.

Author Contributions: Conceptualization, S.-O.S. and J.-Y.B.; methodology, S.-O.S.; software, B.J.; validation, S.-O.S., J.P. and J.-Y.B.; formal analysis, J.-Y.B.; investigation, J.P.; resources, B.J.; data curation, J.P.; writing—original draft preparation, S.-O.S.; writing—review and editing, B.J.; visualization, J.-Y.B.; supervision, B.J.; project administration, S.-O.S.; funding acquisition, S.-O.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the research fund of Hanbat National University in 2018.

Data Availability Statement: If the data is required, please contact the author for availability.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Linn, R.; Zhang, W. Hybrid flow shop scheduling: A survey. *Comput. Ind. Eng.* **1999**, *37*, 57–61. [CrossRef]
2. Ribas, I.; Leisten, R.; Framinan, J.M. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Comput. Oper. Res.* **2010**, *37*, 1439–1454.
3. Tosun, Ö.; Marichelvam, M.K.; Tosun, N. A literature review on hybrid flow shop scheduling. *Int. J. Adv. Oper. Manag.* **2020**, *12*, 156–194. [CrossRef]
4. Brah, S.A.; Hunsucker, J.L. Branch and bound algorithm for the flow shop with multiple processors. *Eur. J. Oper. Res.* **1991**, *51*, 88–99. [CrossRef]
5. Rajendran, C.; Chaudhuri, D. A multi-stage parallel-processor flowshop problem with minimum flowtime. *Eur. J. Oper. Res.* **1992**, *57*, 111–122. [CrossRef]
6. Gupta, J.N.D.; Hariri, A.M.A.; Potts, C.N. Scheduling a two-stage hybrid flow shop with parallel machines at the first stage. *Ann. Oper. Res.* **1997**, *69*, 171–191. [CrossRef]
7. Moursli, O.; Pochet, Y. A branch-and-bound algorithm for the hybrid flowshop. *Int. J. Prod. Econ.* **2000**, *64*, 113–125. [CrossRef]
8. Azizoglu, M.; Cakmak, E.; Kondakci, S. A flexible flowshop problem with total flow time minimization. *Eur. J. Oper. Res.* **2001**, *132*, 528–538.
9. Lee, G.-C.; Kim, Y.-D. A branch-and-bound algorithm for a two-stage hybrid flowshop scheduling problem minimizing total tardiness. *Int. J. Prod. Res.* **2004**, *42*, 4731–4743. [CrossRef]
10. Dugardin, F.; Yalaoui, F.; Amodeo, L. New multi-objective method to solve reentrant hybrid flow shop scheduling problem. *Eur. J. Oper. Res.* **2010**, *203*, 22–31. [CrossRef]
11. Gomez-Gasquet, P.; Andres, C.; Lario, F.C. An agent-based genetic algorithm for hybrid flowshop with sequence dependent setup times to minimize makespan. *Expert Syst. Appl.* **2012**, *39*, 8095–8107. [CrossRef]
12. Eskandari, H.; Hosseinzadeh, A. A variable neighbourhood search for hybrid flow-shop scheduling problem with rework and set-up times. *J. Oper. Res. Soc.* **2014**, *65*, 1221–1231. [CrossRef]

13. Gupta, J.N.D. Two-Stage hybrid flowshop scheduling problem. *J. Oper. Res. Soc.* **1988**, *39*, 359–364. [CrossRef]
14. Fernandez-Viagas, V.; Molina-Pariente, J.M.; Framinan, J.M. New efficient constructive heuristics for the hybrid flowshop to minimise makespan: A computational evaluation of heuristics. *Expert Syst. Appl.* **2018**, *114*, 345–356.
15. Yang, D.-L.; Chern, M.-S. A two-machine flowshop scheduling problem with limited waiting time constraints. *Comput. Ind. Eng.* **1995**, *28*, 63–70. [CrossRef]
16. Bouquard, J.-L.; Lente, C. Two-machine flow shop scheduling problems with minimal and maximal delays. 4OR-A Q. *J. Oper. Res.* **2006**, *4*, 15–28. [CrossRef]
17. Fondrevelle, J.; Oulamara, A.; Portmann, M. -C. Permutation flowshop scheduling problems with maximal and minimal time lags. *Comput. Oper. Res.* **2006**, *33*, 1540–1556. [CrossRef]
18. Bouquard, J.-L.; Lente, C.; Billaut, J. -C. Application of an optimization problem in Max-Plus algebra to scheduling problems. Discret. *Appl. Math.* **2006**, *154*, 2064–2079.
19. Su, L.-H. A hybrid two-stage flowshop with limited waiting time constraints. *Comput. Ind. Eng.* **2003**, *44*, 409–424.
20. Ruiz, R.; Vázquez-Rodríguez, J.A. The hybrid flow shop scheduling problem. *Eur. J. Oper. Res.* **2010**, *205*, 1–18.
21. Behnamian, J.; Fatemi Ghomi, S.M.T. Hybrid flowshop scheduling with machine and resource-dependent processing times. *Appl. Math. Model.* **2011**, *35*, 1107–1123. [CrossRef]
22. Ying, K.-C.; Lin, S.-W. Minimizing makespan for the distributed hybrid flowshop scheduling problem with multiprocessor tasks. *Expert Syst. Appl.* **2018**, *92*, 132–141. [CrossRef]
23. Öztop, H.; Tasgetiren, M.F.; Eliiyi, D.T.; Pan, Q.-K. Metaheuristic algorithms for the hybrid flowshop scheduling problem. *Comput. Oper. Res.* **2019**, *111*, 177–196. [CrossRef]
24. Lin, S.-W.; Cheng, C.-Y.; Pourhejazy, P.; Ying, K.-C.; Lee, C.-H. New benchmark algorithm for hybrid flowshop scheduling with identical machines. *Expert Syst. Appl.* **2021**, *183*, 115422. [CrossRef]
25. Johnson, S.-M. Optimal two- and three-stage production schedules with setup times included. *Naval Res. Logist. Q.* **1954**, *1*, 61–68. [CrossRef]
26. Gilmore, P.-C.; Gomory, R.-E. Sequencing a one state-variable machine: A solvable case of the traveling salesman problem. *Oper. Res.* **1964**, *12*, 655–679. [CrossRef]
27. An, Y.-J.; Kim, Y.-D.; Choi, S. -W. Minimizing makespan in a two-machine flowshop with a limited waiting time constraint and sequence-dependent setup times. *Comput. Oper. Res.* **2016**, *71*, 127–136. [CrossRef]
28. Lee, J.-Y. A genetic algorithm for a two-machine flowshop with a limited waiting time constraint and sequence-dependent setup times. *Math. Probl. Eng.* **2020**, *13*, 8833645. [CrossRef]
29. Jeong, B.J.; Han, J.-H.; Lee, J.-Y. Metaheuristics for a flow shop scheduling problem with urgent jobs and limited waiting times. *Algorithms* **2021**, *14*, 323. [CrossRef]
30. Chung, T.P.; Sun, H.; Liao, C.J. Two new approaches for a two-stage hybrid flowshop problem with a single batch processing machine under waiting time constraint. *Comput. Ind. Eng.* **2017**, *113*, 859–870. [CrossRef]
31. Azizoglu, M.; Kirca, O. Scheduling jobs on unrelated parallel machines to minimize regular total cost functions. *IIE Trans.* **1999**, *31*, 153–159. [CrossRef]
32. Kim, Y.D. A backward approach in list scheduling algorithms for multi-machine tardiness problems. *Comput. Oper. Res.* **1995**, *22*, 307–319.