machines

MDPI

Article

A Two-Step Approach to Scheduling a Class of Two-Stage Flow Shops in Automotive Glass Manufacturing

Yan Qiao [1,*], Naiqi Wu [1,2], Zhiwu Li [1], Abdulrahman M. Al-Ahmari [3], Abdul-Aziz El-Tamimi [3] and Husam Kaid [3]

[1] Institute of Systems Engineering and Collaborative Laboratory for Intelligent Science and Systems, Macau University of Science and Technology, Macao 999078, China
[2] State Key Laboratory of Precision Electronic Manufacturing Technology and Equipment, Guangdong University of Technology, Guangzhou 510006, China
[3] Industrial Engineering Department, College of Engineering, King Saud University, P.O. Box 800, Riyadh 11421, Saudi Arabia
* Correspondence: yqiao@must.edu.mo

Abstract: Driven from real-life applications, this work aims to cope with the scheduling problem of automotive glass manufacturing systems, that is characterized as a two-stage flow-shop with small batches, inevitable setup time for different product changeover at the first stage, and un-interruption requirement at the second stage. To the best knowledge of the authors, there is no report on this topic from other research groups. Our previous study presents a method to assign all batches to each machine at the first stage only without sequencing the assigned batches, resulting in an incomplete schedule. To cope with this problem, if a mathematical programming method is directly applied to minimize the makespan of the production process, binary variables should be introduced to describe the processing sequence of all the products, not only the batches, resulting in huge number of binary variables for the model. Thus, it is necessary and challenging to search for a method to solve the problem efficiently. Due to the mandatory requirement that the second stage should keep working continuously without interruption, solution feasibility is essential. Therefore, the key to solve the addressed problem is how to guarantee the solution feasibility. To do so, we present a method to determine the minimal size of each batch such that the second stage can continuously work without interruption if the sizes of all batches are same. Then, the conditions under which a feasible schedule exists are derived. Based on the conditions, we are able to develop a two-step solution method. At the first step, an *integer linear program* (ILP) is formulated for handling the batch allocation problem at the first stage. By the ILP, we need then to distinguish the batches only, greatly reducing the number of variables and constraints. Then, the batches assigned to each machine at the first stage are optimally sequenced at the second step by an algorithm with polynomial complexity. In this way, by the proposed method, the computational complexity is greatly reduced in comparison with the problem formulation without the established feasibility conditions. To validate the proposed approach, we carry out extensive experiments on a real case from an automotive glass manufacturer. We run ILP on CPLEX for testing. For large-size problems, we set 3600 s as the longest time for getting a solution and a gap of 1% for the lower bound of solutions. The results show that CPLEX can solve 96.83% cases. Moreover, we can obtain good solutions with the maximum gap of 4.9416% for the unsolved cases.

Keywords: CPLEX; glass manufacturing; integer linear program; scheduling; two-stage flow-shop

Citation: Qiao, Y.; Wu, N.; Li, Z.; Al-Ahmari, A.M.; El-Tamimi, A.-A.; Kaid, H. A Two-Step Approach to Scheduling a Class of Two-Stage Flow Shops in Automotive Glass Manufacturing. *Machines* **2023**, *11*, 292. https://doi.org/10.3390/machines11020292

Academic Editor: Mosè Gallo

Received: 1 December 2022
Revised: 3 February 2023
Accepted: 5 February 2023
Published: 15 February 2023

1. Introduction

Due to globalization, products are diversified and should be produced in a customized way with small batches and low cost. To do so, a manufacturing system should be flexible. One important flexibility type of a production system is characterized by the capability

to produce multiple product types. To produce different products in a single system, changeovers are performed frequently due to the small batch size and setups often required [1]. In many cases, the setup activities take significant time, which cannot be ignored in operating a system. Flow-shop systems are widely adopted production systems. In a classical flow-shop system, a set of jobs are sequentially processed at multiple stages, where each stage has one machine only. If there is at least one stage consisting of more than one machine, the system is addressed as a flexible flow-shop [2]. A flexible flow-shop is also called a hybrid flow shop. Scheduling a flexible flow-shop production system that processes a variety of product types with small batches is an important issue in industrial applications.

In this paper, we investigate the scheduling problem of *automotive glass manufacturing systems* (AGMSs) that is described as follows. In an automotive context, there are many glass products and all of them should be produced in a single AGMS. For a glass product to be produced in an AGMS, there are five processing steps: pre-processing, bending, *Polyvinyl Butyral* (PVB) stretching and lamination, vacuuming, and final inspection. The shapes of different products produced in an AGMS are different, changeovers for producing different products are inevitable. For the addressed AGMS, setups with significant time for a changeover are required at Step 1. At Step 2, dies are used to form different shapes for different products. By proper process planning, dies can be changed when a machine is operating, leading to no setup time. Furthermore, in an AGMS, no setup is required for Steps 3–5 either and Steps 2–5 are designed to be paced to maximize the productivity. Therefore, an AGMS is a typical *two-stage flow-shop* (TSFS), where Step 1 forms the first stage, while the other steps form the second stage. Further, there are multiple machines at both Stages 1 and 2. Thus, an AGMS is also a *two-stage flexible flow-shop* (TSFFS).

For the addressed AGMS, the second stage is naturally recognized and designed as the bottleneck in practice since a machine at the second stage cannot be interrupted so as to keep the processing environment (i.e., temperature) due to quality assurance requirements. Thus, this work aims to minimize the makespan of TSFFSs with small batches, sequence-independent setup time at the first stage and no-interruption requirement at the second stage. For this addressed problem, a set of ζ batches (each batch can be treated as a job) are processed. To meet the no-interruption requirement for Stage 2, when a product in a batch is completed by a machine at Stage 1, this product should be scheduled to be processed at Stage 2 immediately without waiting for the completion of the entire batch. Furthermore, after processing a product in a batch that is previously completed by machine i, the next product processed by Stage 2 may be a product from a different batch completed by machine j with $i \neq j$. Thus, if a mathematical programming method is directly applied to minimize the makespan of the presented TSFFS in this work, we need binary variables to identify the processing sequence of all the products but not only the batches. In this way, a huge number of binary variables should be introduced, leading to a model size that is extremely large. Thus, it is an important issue to search for a method such that the problem can be efficiently solved, which it is very challenging. This motivates us to conduct this work.

Due to the mandatory requirement that the second stage should keep working continuously without interruption, solution feasibility is essential. Therefore, the key to solve the addressed problem is how to guarantee solution feasibility. To do so, this work makes the following contributions:

(1) We present a method to determine the minimal size of each batch such that the second stage can continuously keep working without interruption if the sizes of all batches are same;
(2) The conditions under which a feasible schedule exists are derived;
(3) Based on the conditions, we are able to develop a two-step solution method;
    (3.1) At the first step, an *integer linear program* (ILP) is formulated for handling the batch allocation problem at the first stage.
    (3.2) At the second step, the batches assigned to each machine at the first stage are optimally sequenced by an algorithm with polynomial complexity.

For the proposed two-step method, at the first step, we need to distinguish the batches only, greatly reducing the number of variables and constraints in the ILP. In this way, by the proposed method, we greatly reduce the computational complexity in comparison with the problem formulation without the established feasibility conditions.

The rest of the paper is constructed as follows. Section 2 briefly reviews the related work. Section 3 analyzes properties of the system. Then, a two-step solution method is proposed in Section 4. Experimental results are given to test the performance of the proposed method in Section 5. Finally, the last section concludes this work.

2. Literature Review

Many studies have been conducted on flow-shop scheduling problems. For flow-shop scheduling problems, there are various variants, considering different constraints, scheduling scenarios, and optimization objectives [3]. The main constraints include setup time [4], limited buffers [5], uncertain processing time [6], due dates [7], etc. Many researchers attempt to solve such scheduling problems by using mathematical models [4,8], meta-heuristics algorithms [5,9], artificial intelligence [4], and methods based on scheduling policy [10,11]. Interested readers can refer to the surveys in [3,12,13] for more details about the scheduling analysis of flow-shop systems. Since the addressed scheduling problem comes from a real AGMS that is treated as a TSFS, this work mainly reviews the works related to the scheduling problems of TSFS.

Some studies have been carried out for TSFS system scheduling problems. With one machine at Stage 1, by considering the machine breakdown, [14] propose efficient approaches to minimize the makespan. With one machine at Stage 2, based on a branch and bound technique, [15] developed an algorithm to find an optimal schedule. With arbitrary arrival time of jobs, [16] present an ant colony algorithm to schedule a TSFS system. A multi-objective hybrid ant colony algorithm is constructed by [17] to trade off the makespan and total energy costs for a two-stage blocking permutation flow-shop. For a no-waiting TSFS, [18] design a linear-time combinatorial algorithm.

By considering independent setup time, [19] construct several heuristic algorithms for scheduling a TSFS. With the fuzzy makespan and total agreement index as objective, [20] adopt a diversified teaching–learning-based algorithm for a TSFS with sequence-dependent setup time. By applying reinforcement learning, [21] propose a novel approach for the TSFS system scheduling problem. In addition, efficient algorithms are presented for two-stage assembly flow-shops [22–25]. In such a system, normally, the first stage is used to fabricate multiple types of components, while the second stage assembles the components to form the final product.

For a TSFS system, if each stage has one machine only, the system becomes a *two-machine flow shop* (TMFS) and great efforts have been made for scheduling such systems. With sequence-independent setup time, the branch and bound techniques are employed to optimize the makespan in [26]. In [27,28], methods are proposed to minimize makespan by considering the release dates, while in [29], several metaheuristics are examined by considering both release dates and blocking constraint. To minimize the total tardiness, lower bounds and dominance conditions are established in [30]. In addition, there are studies [31,32] that focus on TMFSs with deteriorating jobs. Note that there are extensive studies on scheduling TMFSs. Due to the fact that the addressed AGMS in this work is much more complex, we do not make an extensive review on this issue, in order to save space.

To the best knowledge of the authors, there is no report on the scheduling problem of TSFFSs with small batches, sequence-independent setup time at the first stage and no-interruption requirement at the second stage from other research groups. For the addressed problem in this work, our previous study presents a method to assign all batches to each machine at Stage 1 for being processed [33]. However, it gives no way to sequence the assigned batches, resulting in an incomplete schedule. Moreover, it does not present how to find the minimal size of each batch such that Stage 2 can continuously work

without interruption if the sizes of all batches are same. Notice that the key to present the existence conditions of a feasible schedule is to determine the minimal sizes of all batches. This motivates us to perform this work such that the addressed problem can be completely solved.

3. Scheduling Analysis

3.1. System Description

Before presenting the proposed method, we first introduce the addressed AGMS as follows.

(1) There are parallel multiple machines with same processing functions at each stage;
(2) Only one product can be processed by a machine at a time;
(3) At Stage 1, a batch should be processed by a single machine without splitting;
(4) Stage 2 is the bottleneck;
(5) Stage 1 has the setup time;
(6) Each batch contains identical products;
(7) For any two batches, their product types are different, with the result that if one batch is just processed at Stage 1, setup time is required before another one is processed next; and
(8) After its completion at Stage 1, a product is ready to be processed immediately by a machine at Stage 2.

Since each stage has multiple machines, $g \geq 1$ and $h \geq 1$ are used to denote the number of machines at Stages 1 and 2, respectively. Further, let $\mathbb{N}_k = \{1, 2, \ldots, k\}$. Then, M_{1i}, $i \in \mathbb{N}_g$, and M_{2j}, $j \in \mathbb{N}_h$, represent the i-th and j-th machines at the two stages, respectively. Figure 1 describes an AGMS with small batches to be processed and multiple machines at each stage. The time for processing a product of any type by M_{1i} and M_{2j} is known and denoted as α and μ, respectively. Further, the setup time for a machine at Stage 1 is known and it is represented by δ. Without setup time at Stage 2, we treat the h machines there as just one machine, leading to that the processing time of a product at Stage 2 is $\beta = \frac{\mu}{h}$ time units.

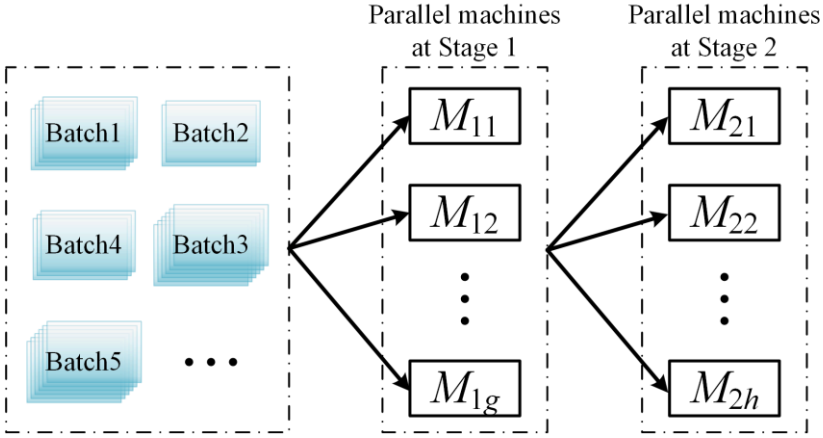

Figure 1. An AGMS.

We ensure that $\frac{\alpha}{g} < \beta$ should hold, otherwise some machines at the second stage have to be interrupted from time to time due to the setups for M_{1i}, $i \in \mathbb{N}_g$. In that case, the problem is similar to schedule a single machine. For such a scheduling problem, there are extensive studies [34–40]. Thus, to ensure the non-interruption requirement of machines at the second stage, it should satisfy $\frac{\alpha}{g} < \beta$ when the system is designed. By the experience of the authors with some companies in China, this is true for AGMSs. With $\frac{\alpha}{g} < \beta$ and the non-interruption constraint, it is a great challenge to maximize the productivity by finding a feasible schedule for processing multiple products, which is the aim of this work.

3.2. Properties of the System

Due to there being many glass products for automotives, in an AGMS, the number of machines at the first stage is much less than the number of product types to be produced. Moreover, the total number of products to be produced for a scheduling horizon is much greater than the number of products in a batch. To maximize the throughput of an AGMS, the key is that one should properly allocate the batches to the machines at the first stage and sequence them for each M_{1i}, $i \in \mathbb{N}_{\mathbf{g}}$. Suppose that $\eta(i)$ batches are allocated to M_{1i}, $i \in \mathbb{N}_{\mathbf{g}}$. Let BH_{ij}, $i \in \mathbb{N}_{\mathbf{g}}$ and $j \in \mathbb{N}_{\boldsymbol{\eta(i)}}$, denote the j-th one in the $\eta(i)$ batches with its batch size being b_{ij}. Without loss of generality, it is assumed that, at time zero, the system starts to process batch BH_{i1}, $i \in \mathbb{N}_{\mathbf{g}}$. We use T_{ij} to denote the completion time of batch BH_{ij}, $i \in \mathbb{N}_{\mathbf{g}}$ and $j \in \mathbb{N}_{\boldsymbol{\eta(i)}}$. Then, we have

$$T_{ij} = \alpha \times \sum_{d=1}^{j} b_{id} + (j-1) \times \mathit{ffi},\ i \in \mathbb{N}_{\mathbf{g}} \text{ and } j \in \mathbb{N}_{\boldsymbol{\eta(i)}} \tag{1}$$

Note that $T_{i\eta(i)}$ gives the time when M_{1i}, $i \in \mathbb{N}_{\mathbf{g}}$, completes its allocated batches. Let ϑ_{it} be the average production rate of M_{1i}, $i \in \mathbb{N}_{\mathbf{g}}$, in $[0, t]$, i.e., the average number of products completed by M_{1i} per unit time in $[0, t]$; and $T_{min} = min(T_{i\eta(i)} \mid i \in \mathbb{N}_{\mathbf{g}})$. The following lemma can be obtained.

Lemma 1. *If* $\sum_{i=1}^{g} \vartheta_{it} \geq \frac{1}{\beta'}$, $\forall t \in (0, T_{min}]$, *then Stage 2 can operate uninterruptedly during* $(0, T_{min}]$.

Proof. With ϑ_{it} being the average production rate of M_{1i}, $i \in \mathbb{N}_{\mathbf{g}}$, in $[0, t]$, the average production rate of Stage 1 in $[0, t]$ is $\sum_{i=1}^{g} \vartheta_{it}$. Notice that the maximal production rate of Stage 2 is $\frac{1}{\beta}$. Thus, if $\sum_{i=1}^{g} \vartheta_{it} \geq \frac{1}{\beta'}\ \forall t \in (0, T_{min}]$, then, at any time point in $(0, T_{min}]$, the production rate of Stage 1 is greater than or equal to that of Stage 2. That is to say that there are always products for Stage 2 to process and no machine needs to stop in $(0, T_{min}]$, i.e., the lemma holds. □

With Lemma 1, we obtain the uninterrupted operation condition of Stage 2. In practice, due to significant setup time at Stage 1, if some machines at Stage 1 are switching from processing one batch to another, it may result in Stage 1 being unable to feed enough products to the machines at Stage 2 (the bottleneck), leading to some machines at Stage 2 being unable to keep continuously working. To solve this problem, similar to the basic idea of the drum-buffer-rope [41,42], buffers (the inventory) can be used to protect the bottleneck such that the uninterrupted operation condition of Stage 2 shown in Lemma 1 is satisfied. Therefore, it is important to schedule the batches to be processed at Stage 1. Moreover, it is necessary to determine the minimal size of each batch such that Stage 2 can continuously work without interruption if the sizes of all batches are the same. Thus, for more general cases, it is possible to find a method to schedule the batches to be processed at Stage 1 so as to protect the bottleneck by the inventory.

Suppose that all the machines at Stage 1 process batches with the same size in a paced way. In other words, the size of all batches to be processed are same and all the machines at Stage 1 start to process a batch at the same time. In this way, all the machines switch from producing a type of product to another type at the same time as well. Then, we can determine the minimal size of each batch denoted by Θ such that Stage 2 can operate continuously with no interruption.

Lemma 2. *Suppose that all the* g *machines at Stage 1 perform the product processing and switching operations in a paced way and the size of the batches to be processed are same. Then, if the size of each batch is no less than* $\Theta = \frac{\delta}{g \times \beta - \alpha}$ *Stage 2 can operate uninterruptedly.*

Proof. When the g machines at the first stage start to process a batch at the same time, then the inventory between Stages 1 and 2 should increase over time due to $\frac{\alpha}{g} < \beta$. Assume

that after t_0 time units elapse, a batch at each machine at Stage 1 is just completed. At this time, every one of the g machines switches from processing one batch to another exactly at the same time. Then, the inventory between the two stages decreases over time since Stage 2 continuously consumes the inventory. After δ time units elapse, the inventory reaches zero. In this way, there are always products for Stage 2 to produce such that it cannot be interrupted since the g machines at Stage 1 start to process a batch at the same time, leading to the inventory between Stages 1 and 2 increasing over time. This means that in this situation, the size of the batches is minimal. The inventory between Stages 1 and 2 changes periodically over time, as shown in Figure 2. Thus, we have that during time t_0, each machine at Stage 1 can produce $\Theta = \frac{t_0}{\alpha}$ products, i.e., a batch. In total, Stage 1 produces $g \times \frac{t_0}{\alpha}$ products. During time $t_0 + \delta$, Stage 2 just consumes $\frac{t_0+\delta}{\beta}$ products. Note that $g \times \frac{t_0}{\alpha} = \frac{t_0+\delta}{\beta}$ and $t_0 = \alpha\Theta$, leading to $\Theta = \frac{\delta}{g\times\beta-\alpha}$ Therefore, the lemma holds. □

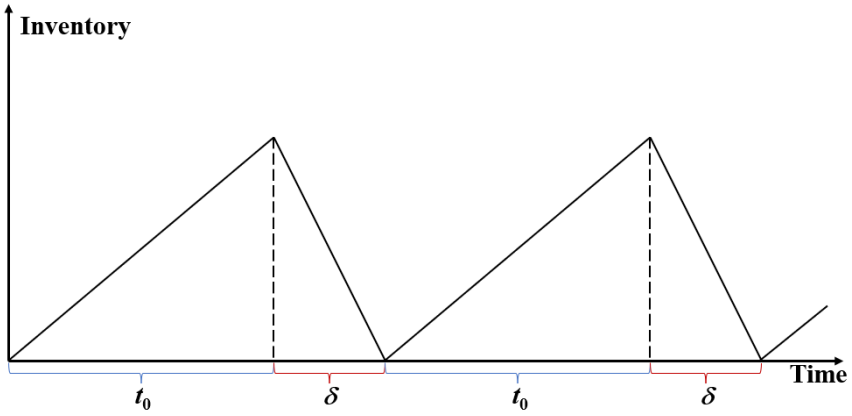

Figure 2. The changes of the inventory between Stages 1 and 2 over time.

Notice that $\Theta > 0$ since $\frac{\alpha}{g} < \beta$ results in $\alpha < g\beta$. With Lemmas 1 and 2, to develop a method such that Stage 2 can operate uninterruptedly, we have the following result for one of the cases.

Theorem 1. *Given that BH_{ij}, $i \in \mathbb{N}_g$ and $j \in \mathbb{N}_{\eta(i)}$, is the j-th batch in the processing sequence for M_{1i}, for $\forall t \in (0, T_{min}]$, $\sum_{i=1}^{g} \vartheta_{it} \geq \frac{1}{\beta}$ holds if $b_{ij} \geq \Theta$, $\forall j \in \mathbb{N}_{\eta(i)}$ and $\forall i \in \mathbb{N}_g$.*

Proof. With $\frac{\alpha}{g} < \beta$, as illustrated in Figure 3, before the completion of the first batch, the production rate of M_{1i} is $\frac{1}{\alpha} > \frac{1}{g\beta}$, at time $t_1 \in [0, T_{i1}]$, $i \in \mathbb{N}_g$.

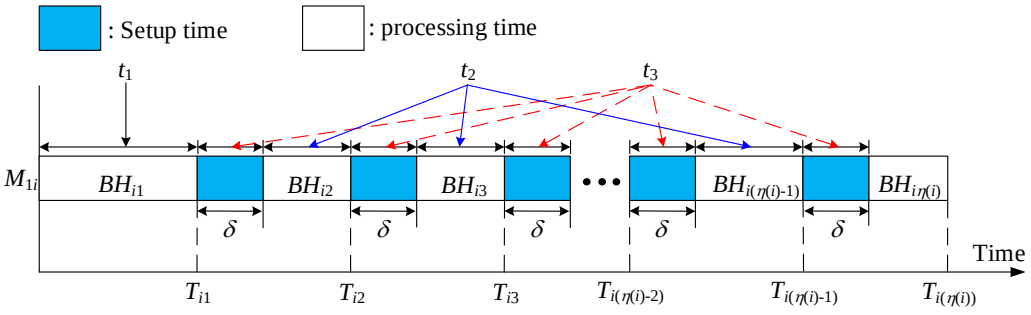

Figure 3. Illustration for Theorem 1.

As illustrated in Figure 3, it follows from (1) that, at time $T_{iu} + \delta$, $1 \leq u \leq \eta(i) - 1$, $\eta(i) \geq 2$, and $i \in \mathbb{N}_{\mathbf{g}}$, the average production rate of M_{1i} can be calculated as

$$\frac{\sum_{j=1}^{u} b_{ij}}{T_{iu} + \delta} = \frac{\sum_{j=1}^{u} b_{ij}}{\alpha \times \sum_{j=1}^{u} b_{ij} + u \times \delta}$$

With $b_{ij} \geq \Theta$, $i \in \mathbb{N}_{\mathbf{g}}$ and $j \in \mathbb{N}_{\eta(i)}$, $\sum_{j=1}^{u} b_{ij} \geq u \times \Theta$ holds. Notice that $a > 0$, $b > 0$, and $c > 0$ are constants, so, we can obtain a monotonically increasing function $f_1(x) = \frac{ax}{bx+c}$. Then, with constants α, u, and δ, another monotonically increasing function $f_2(x) = \frac{x}{\alpha x + u\delta}$ can be obtained. Replace x by $u \times \Theta$ in $f_2(x)$, resulting in $f_2(u \times \Theta) = \frac{1}{g\beta}$. Since $x = \sum_{j=1}^{u} b_{ij} \geq u \times \Theta$, $f_2(\sum_{j=1}^{u} b_{ij}) \geq \frac{1}{g\beta}$ holds, resulting in $\frac{\sum_{j=1}^{u} b_{ij}}{T_{iu}+\delta} \geq \frac{1}{g\beta}$, implying that, at time $T_{iu} + \delta$, $i \in \mathbb{N}_{\mathbf{g}}$, $\eta(i) \geq 2$, and $1 \leq u \leq \eta(i) - 1$, the production rate of M_{1i} is no less than $\frac{1}{g\beta}$.

As illustrated in Figure 3, it follows from (1) that, at any time $t_2 \in (T_{iu} + \delta, T_{i(u+1)}]$, $i \in \mathbb{N}_{\mathbf{g}}$, $\eta(i) \geq 2$, and $1 \leq u \leq \eta(i) - 1$, the average production rate of M_{1i} can be calculated as

$$\frac{\sum_{j=1}^{u} b_{ij} + \left(\frac{t_2 - T_{iu} - \delta}{\alpha}\right)}{t_2} = \frac{\sum_{j=1}^{u} b_{ij} + \left(\frac{t_2 - T_{iu} - \delta}{\alpha}\right)}{\alpha \times \left[\sum_{j=1}^{u} b_{ij} + \left(\frac{t_2 - T_{iu} - \delta}{\alpha}\right)\right] + u \times \delta}$$

Since $\sum_{j=1}^{u} b_{ij} + \left(\frac{t_2 - T_{iu} - \delta}{\alpha}\right) > \sum_{j=1}^{u} b_{ij} \geq u \times \Theta$ holds, we have $f_2(\sum_{j=1}^{u} b_{ij} + \left(\frac{t_2 - T_{iu} - \delta}{\alpha}\right))$ $> \frac{1}{g\beta}$, i.e., $\frac{\sum_{j=1}^{u} b_{ij} + \left(\frac{t_2 - T_{iu} - \delta}{\alpha}\right)}{t_2} > \frac{1}{g\beta}$. Hence, at time t_2, $T_{iu} + \delta < t_2 \leq T_{i(u+1)}$, $i \in \mathbb{N}_{\mathbf{g}}$, $\eta(i) \geq 2$, and $1 \leq u \leq \eta(i) - 1$, the production rate of M_{1i} is greater than $\frac{1}{g\beta}$.

As illustrated in Figure 3, it follows from (1) that, at any time $t_3 \in (T_{iu}, T_{iu} + \delta)$, $i \in \mathbb{N}_{\mathbf{g}}$, $\eta(i) \geq 2$, and $1 \leq u \leq \eta(i) - 1$, the average production rate of M_{1i} can be calculated as

$$\frac{\sum_{j=1}^{u} b_{ij}}{t_3} = \frac{\sum_{j=1}^{u} b_{ij}}{\alpha \times \sum_{j=1}^{u} b_{ij} + (u-1) \times \delta + t_3 - T_{iu}}$$

Due to $0 < t_3 - T_{iu} < \delta$, we have

$$\frac{\sum_{j=1}^{u} b_{ij}}{t_3} = \frac{\sum_{j=1}^{u} b_{ij}}{\alpha \times \sum_{j=1}^{u} b_{ij} + (u-1) \times \delta + t_3 - T_{iu}} > \frac{\sum_{j=1}^{u} b_{ij}}{\alpha \times \sum_{j=1}^{u} b_{ij} + u \times \delta} = f_2\left(\sum_{j=1}^{u} b_{ij}\right) \geq \frac{1}{g\beta}$$

Hence, at time t_3, $T_{iu} < t_3 < T_{iu} + \delta$, $i \in \mathbb{N}_{\mathbf{g}}$, $\eta(i) \geq 2$, and $1 \leq u \leq \eta(i) - 1$, the production rate of M_{1i} is greater than $\frac{1}{g\beta}$.

Therefore, by summarizing the production rates of the g machines, we conclude that, at any time during $(0, T_{min}]$, $\sum_{i=1}^{g} \vartheta_{it} \geq \frac{1}{\beta}$, or the total production rate of Stage 1 is greater than or equal to $\frac{1}{\beta}$ for any $t \in (0, T_{min}]$. □

In this case, to ensure that the machines at Stage 2 can operate uninterruptedly, it requires that the number of products in any batch should be greater or equal to Θ. By the conditions given in Theorem 1, it implies that before any changeover for product switching, there is enough inventory in the buffer between the two stages such that Stage 2 cannot be starved during the setup time δ. Furthermore, by Theorem 1, it can be concluded that, if the conditions given in Theorem 1 are satisfied, different batch sequences do not affect the production rate of Stage 2, since it is always in operation no matter how the batches are sequenced. However, there may be batches with size less than Θ. It raises a question if Stage 2 can operate uninterruptedly if batches with size less than Θ exist. The following theorem presents an answer to it.

Theorem 2. *Given that BH_{ij}, $i \in \mathbb{N}_{\mathbf{g}}$ and $j \in \mathbb{N}_{\eta(i)}$, is the j-th batch in the processing sequence for M_{1i}, then for any $t \in (0, T_{min}]$, $\sum_{i=1}^{g} \vartheta_{it} \geq \frac{1}{\beta}$ holds if $\sum_{j=1}^{s} b_{ij} \geq s \times \Theta$, $\forall i \in \mathbb{N}$ and $\forall s \in \mathbb{N}_{\eta(i)-1}$.*

Proof. Similar to Theorem 1, at time $t_4 \leq T_{i1}$, $i \in \mathbb{N}_g$, the production rate of M_{1i} is $\frac{1}{\alpha} > \frac{1}{g\beta}$. It follows from (1) that, at any time $t_5 \in [T_{iu} + \delta, T_{i(u+1)}]$, $i \in \mathbb{N}_g$, $\eta(i) \geq 2$, and $1 \leq u \leq \eta(i) - 1$, the average production rate of M_{1i} can be calculated as

$$\frac{\sum_{j=1}^{u} b_{ij} + \left(\frac{t_5 - T_{iu} - \delta}{\alpha}\right)}{t_5} = \frac{\sum_{j=1}^{u} b_{ij} + \left(\frac{t_5 - T_{iu} - \delta}{\alpha}\right)}{\alpha \times \left[\sum_{j=1}^{u} b_{ij} + \left(\frac{t_5 - T_{iu} - \delta}{\alpha}\right)\right] + u \times \delta}$$

Due to $\sum_{j=1}^{u} b_{ij} + \left(\frac{t_5 - T_{iu} - \delta}{\alpha}\right) \geq \sum_{j=1}^{u} b_{ij} \geq u \times \Theta$, by Theorem 1, $f_2(\sum_{j=1}^{u} b_{ij} + \left(\frac{t_5 - T_{iu} - \delta}{\alpha}\right)) \geq \frac{1}{g\beta}$, i.e., $\frac{\sum_{j=1}^{u} b_{ij} + \left(\frac{t_5 - T_{iu} - \delta}{\alpha}\right)}{t_5} \geq \frac{1}{g\beta}$ holds. Hence, at time t_5, $T_{iu} + \delta \leq t_5 \leq T_{i(u+1)}$, $i \in \mathbb{N}_g$, $\eta(i) \geq 2$, and $1 \leq u \leq \eta(i) - 1$, the production rate of M_{1i} is no less than $\frac{1}{g\beta}$.

It follows from (1) that, at any time $t_6 \in (T_{iu}, T_{iu} + \delta)$, $i \in \mathbb{N}_g$, $\eta(i) \geq 2$, and $1 \leq u \leq \eta(i) - 1$, the average production rate of M_{1i} can be calculated as

$$\frac{\sum_{j=1}^{u} b_{ij}}{t_6} = \frac{\sum_{j=1}^{u} b_{ij}}{\alpha \times \sum_{j=1}^{u} b_{ij} + (u-1) \times \delta + t_6 - T_{iu}}.$$

Just as Theorem 1, with $0 < t_6 - T_{iu} < \delta$, we can obtain

$$\frac{\sum_{j=1}^{u} b_{ij}}{t_6} > \frac{\sum_{j=1}^{u} b_{ij}}{\alpha \times \sum_{j=1}^{u} b_{ij} + u \times \delta} = f_2\Big(\sum_{j=1}^{u} b_{ij}\Big) \geq \frac{1}{g\beta}$$

Hence, at time t_6 with $T_{iu} < t_6 < T_{iu} + \delta$, $i \in \mathbb{N}_g$, $\eta(i) \geq 2$, and $1 \leq u \leq \eta(i) - 1$, the production rate of M_{1i} is greater than $\frac{1}{g\beta}$, resulting in that, during $(0, T_{min}]$, $\sum_{i=1}^{g} \vartheta_{it} \geq \frac{1}{\beta}$, or the total production rate of Stage 1 is no less than $\frac{1}{\beta}$ for any $t \in (0, T_{min}]$. □

In this case, there exist schedules such that when some machines at Stage 1 execute setup activities, there are enough products that have been processed at Stage 1 in the buffer between the stages such that Stage 2 cannot be starved. Then, we examine the case that the batch size of every batch is less than Θ. In this case, we have the following result.

Theorem 3. *Given that $BH_{ij}, \forall i \in \mathbb{N}_g$ and $\forall j \in \mathbb{N}_{\eta(i)}$ with $\eta(i) \geq 2$, is the j-th batch in the processing sequence for M_{1i}, then interruptions of Stage 2 cannot be avoided in time interval $(0, T_{min}]$ if $b_{ij} < \Theta, \forall i \in \mathbb{N}_g, \forall j \in \mathbb{N}_{\eta(i)}$ with $\eta(i) \geq 2$.*

Proof. It follows from (1) that, at time $T_{iu} + \delta$, $i \in \mathbb{N}_g$, $\eta(i) \geq 2$, and $1 \leq u \leq \eta(i) - 1$, the production rate of M_{1i} can be calculated as

$$\frac{\sum_{j=1}^{u} b_{ij}}{T_{iu} + \delta} = \frac{\sum_{j=1}^{u} b_{ij}}{\alpha \times \sum_{j=1}^{u} b_{ij} + u \times \delta}.$$

With $b_{ij} < \Theta$, $\forall i \in \mathbb{N}_g$, $\forall j \in \mathbb{N}_{\eta(i)}$, and $\eta(i) \geq 2$, $\sum_{j=1}^{u} b_{ij} < u \times \Theta$ holds. It follows from Theorem 1 that $f_2(\sum_{j=1}^{u} b_{ij}) < f_2(u \times \Theta) = \frac{1}{g\beta}$ holds, resulting in that, at time $T_{iu} + \delta$, $i \in \mathbb{N}_g$, $\eta(i) \geq 2$, and $1 \leq u \leq \eta(i) - 1$, the production rate of M_{1i} is less than $\frac{1}{g\beta}$, leading to $\sum_{i=1}^{g} \vartheta_{it} < \frac{1}{\beta}$, $t = T_{iu} + \delta$. This means that Stage 2 should be starved for some time in $(0, T_{min}]$, or the theorem holds. □

In this case, during the time when some machine is performing setups, there are not enough products in the buffer between the two stages for Stage 2 to process such that Stage 2 has to stop. Up until now, we have examined the existence of a feasible schedule and established the corresponding conditions. It should be pointed out that, to satisfy the condition given in Theorem 2, one needs to properly schedule the batches to be

produced. Thus, to solve the scheduling problem of AGMSs, based on Theorems 1 and 2, we present a two-step method to find an optimal schedule by developing a mathematical programming model.

4. Two-Step Solution Method

With the above schedule existence conditions, this section presents a two-step solution method. First, at Step 1, we propose an ILP model to allocate the batches to different machines at Stage 1. Then, the batches to be produced by each machine at Stage 1 are sequenced at Step 2 to obtain the optimal solution.

4.1. Formulating the ILP

To effectively schedule the addressed AGMS, we need to allocate the batches to different machines at Stage 1 to balance the workloads and make a schedule feasible. As discussed above, batch size is crucial to find a feasible schedule. Thus, we partition the batches into two parts. Let $\Theta = \frac{\delta}{g\times\beta-\alpha}$. Then, the set of batches with their sizes being greater or equal to Θ forms the first part called Batch Type 1 and denoted by BT1, and the other batches form the other part called Batch Type 2 and denoted by BT2. Assume that $|\,\mathrm{BT1}\,| = n$ and $|\,\mathrm{BT2}\,| = m$ such that we have in total $n + m$ batches to be produced. We number the batches in BT1 and BT2 by consecutive integer numbers such that their index sets are denote as $\mathbb{N}_{\mathbf{n}}$ and $\mathbb{N}_{\mathbf{m}}$, respectively. Further, let $O_i \geq \Theta$, $i\in\mathbb{N}_{\mathbf{n}}$, be the batch size of the i-th batch in BT1 and $\Phi_j < \Theta$, $j\in\mathbb{N}_{\mathbf{m}}$, the batch size of the j-th batch in BT2.

By Theorem 2, $\sum_{i=1}^{g}\vartheta_{it} \geq \frac{1}{\beta}$ holds for any $t \in (0, T_{min}]$, if $\sum_{j=1}^{s} b_{ij} \geq s \times \Theta$, $\forall i\in\mathbb{N}_{\mathbf{g}}$ and $\forall s\in\mathbb{N}_{\eta(\mathbf{i})-1}$. In this case, by Lemma 1, we are sure that a feasible schedule can be found. However, in some cases, $\sum_{j=1}^{s} b_{ij} \geq s \times \Theta$ may not hold for any $i\in\mathbb{N}_{\mathbf{g}}$ and $s\in\mathbb{N}_{\eta(\mathbf{i})-1}$. In this case, to ensure that a feasible schedule can be found, we need to add some extra products into the assigned batches so as to make the conditions given in Theorem 2 satisfied. Without loss of generality, we assume that extra products are added into BH_{i1} for any machine $i\in\mathbb{N}_{\mathbf{g}}$ at Stage 1 and the number of products added to M_{1i} is ξ_i. Then, based on Theorem 2, we present the ILP as follows. The symbols for this model are given in the Nomenclature.

$$\textbf{ILP}\text{: } \textit{minimize } \Gamma \tag{2}$$

$$\text{S. t. } \sum_{i=1}^{g} x_{ij} = 1,\ j \in \mathbb{N}_{\mathbf{n}}, \tag{3}$$

$$\sum_{i=1}^{g} y_{ij} = 1,\ j \in \mathbb{N}_{\mathbf{m}}, \tag{4}$$

$$1 \ \geq\ \sum_{w=1}^{m} z_{iw},\ i \in \mathbb{N}_{\mathbf{g}}, \tag{5}$$

$$\sum_{w=1}^{m} z_{iw} \ \geq\ y_{\mathrm{ij}},\ j \in \mathbb{N}_{\mathbf{m}} \text{ and } i \in \mathbb{N}_{\mathbf{g}}, \tag{6}$$

$$y_{ij} \geq z_{ij},\ j\in\mathbb{N}_{\mathbf{m}} \text{ and } i\in\mathbb{N}_{\mathbf{g}}, \tag{7}$$

$$\sum_{j=1}^{m}\left[y_{ij} \times \left(\Phi_j \times \alpha + \delta\right)\right] + \xi_i \ \times\ \alpha - \delta + \sum_{j=1}^{n}\left[x_{ij} \times \left(O_j \times \alpha + \delta\right)\right] \ \leq\ \Gamma,\ i \in \mathbb{N}_{\mathbf{g}}, \tag{8}$$

$$\sum_{j=1}^{m}\left[\left(y_{ij} - z_{ij}\right) \times \Phi_j\right] + \zeta_i + \sum_{j=1}^{n}\left(x_{ij} \times O_j\right) \geq \sum_{j=1}^{m}\left[\left(y_{ij} - z_{ij}\right)\times\Theta\right] + \sum_{j=1}^{n}\left(x_{ij} \times \Theta\right) \tag{9}$$

In the above ILP, Γ presents the completion time of all batches at Stage 1. Thus, Objective (2) optimizes the total completion time. Constraints (3) and (4) ensure that a batch in both BT1 and BT2 is allocated to a single machine. Constraints (5) and (6) state that one of batches in BT2 allocated to M_{1i}, $i\in\mathbb{N}_{\mathbf{g}}$, should be put at the last position for producing. Constraint (7) says that if no batch in BT2 is allocated to M_{1i}, $i\in\mathbb{N}_{\mathbf{g}}$, i.e., $y_{ij} = 0$, $i\in\mathbb{N}_{\mathbf{g}}$ and $j\in\mathbb{N}_{\mathbf{m}}$, then we have $z_{ij} = 0$, $i\in\mathbb{N}_{\mathbf{g}}$ and $j\in\mathbb{N}_{\mathbf{m}}$. Constraint (8) enforces that the total completion time of batches that are allocated to M_{1i} should not be longer than Γ. For Constraint (9), two cases exist and they are Case 1: $\sum_{w=1}^{m} z_{iw} = 0$; and Case 2: $\sum_{w=1}^{m} z_{iw} = 1$. For the former, it follows from Constraint (6) that $y_{ij} = 0$, $j\in\mathbb{N}_{\mathbf{m}}$, holds, implying that M_{1i} produces no batches in BT2. Hence, according to Theorem 1, the production rate of M_{1i} is always greater or

equal to $\frac{1}{g\beta}$. For the latter, if $\sum_{j=1}^{m}\left[\left(y_{ij}-z_{ij}\right)\times\Phi_j\right]+\xi_i+\sum_{j=1}^{n}\left(x_{ij}\times O_j\right)\geq\sum_{j=1}^{n}\left(x_{ij}\times\Theta\right)+\sum_{j=1}^{m}\left[\left(y_{ij}-z_{ij}\right)\times\Theta\right]$, $i\in\mathbb{N}_{\mathbf{g}}$, by Theorem 2, the satisfaction of Constraint (9) implies that the production rate of M_{1i} is no less than $\frac{1}{g\beta}$ all the time. Therefore, Constraint (9) ensure that the obtained solution is feasible even batches in BT2 are produced by M_{1i}, $i\in\mathbb{N}_{\mathbf{g}}$.

4.2. Batch Sequencing for Machines at Stage 1

With the above developed ILP, batches are allocated to each machine at Stage 1 without determining their sequence for feasibility. Thus, based on the solution obtained by solving the ILP, we need to sequence the batches allocated to each machine such that the conditions given in Theorem 2 are met to ensure the feasibility of a solution. Assume that, by solving the ILP, the set of batches allocated to M_{1i}, $i\in\mathbb{N}_{\mathbf{g}}$, is Λ_i. We have $|\Lambda_i| = \eta(i)$ if $z_{ij} = 0$, otherwise $|\Lambda_i| = \eta(i)-1$ if $z_{ij} = 1$. Then, Algorithm 1 is used to sequence the batches for M_{1i}, $i\in\mathbb{N}_{\mathbf{g}}$, such that the conditions given in Theorem 2 are ensured.

Algorithm 1:	Batch sequencing for M_{1i}, $i\in\mathbb{N}_{\mathrm{g}}$, based on the solution of ILP.
Input:	$\eta(i)$, Θ, and Λ_i, $i\in\mathbb{N}_{\mathrm{g}}$
Output:	BH_{ij}, $i\in\mathbb{N}_{\mathrm{g}}$ and $j\in\mathbb{N}_{\eta(i)-1}$
1:	$\Lambda_i' = \emptyset$ /*initialize the sequenced batches*/;
2:	Choose a batch in Λ_i and set it to be BH_{i1} such that $\xi_i + b_{i1} \geq \Theta$;
3:	$\Lambda_i' \leftarrow \Lambda_i' \cup \{BH_{i1}\}$;
4:	$\Lambda_i \leftarrow \Lambda_i \backslash \{BH_{i1}\}$;
5:	$s = 2$;
6:	While $s \leq \eta(i) - 1$
7:	Choose a batch in Λ_i and set it to be BH_{is} such that $\xi_i + \sum_{j=1}^{s} b_{ij} \geq s \times \Theta$;
8:	$\Lambda_i' \leftarrow \Lambda_i' \cup \{BH_{is}\}$;
9:	$\Lambda_i \leftarrow \Lambda_i \backslash \{BH_{is}\}$;
10:	$s = s + 1$;
11:	End;

In Algorithm 1, Statement 2 ensures that when the first batch BH_{i1} is determined, we have $\xi_i + b_{i1} \geq \Theta$, i.e., the conditions given in Theorem 2 are met. Then, by Statements 6–11 in Algorithm 1, it ensures that, with every batch being added, the conditions given in Theorem 2 are met. After performing Statement 11, it results in $\Lambda_i = \emptyset$ or $|\Lambda_i| = 1$. If $\Lambda_i = \emptyset$, we can conclude that $z_{ij} = 1$. In this case, all the batches have been sequenced. If $|\Lambda_i| = 1$, we simply put the remaining batch in Λ_i at the last position. In this way, we obtain a feasible schedule.

Let us analyze the worst-case running time of the algorithm. There are two situations as follows.

Situation 1. *If no batch in BT2 is allocated to M_{1i} (i.e., $z_{ij} = 0$), initially, there are $\eta(i)$ batches in Λ_i. At this time, the running time of Statements 2 and 7 can be treated as a constant since $\xi_i + \sum_{j=1}^{s} b_{ij} \geq s \times \Theta$ should be held no matter which batches in Λ_i is selected based on Theorem 1. Thus, the running time of performing Statements 6–11 is $O(\eta(i))$, leading to the worst-case running time of the algorithm is $O(\eta(i))$.*

Situation 2. *If $z_{ij} = 1$ holds, initially, there are $\eta(i) - 1$ batches in Λ_i. Thus, the worst-case running time of Statement 2) is $O(\eta(i) - 1) = O(\eta(i))$. By performing Statement 6–11, it needs to make a selection for $\eta(i) - 2$ times. From the first time to the last time, the worst-case running time of the selections by performing Statement 7) is $O(\eta(i) - 2)$, $O(\eta(i) - 3)$, . . . , $O(1)$, respectively. Thus, the worst-case running time of performing Statements 6–11 is $O(\frac{(\eta(i)-2+1)(\eta(i)-2)}{2}) = O((\eta(i))^2)$.*

From the above analysis, the computational complexity of this algorithm is $O((\eta(i))^2)$ for the worst-case. Note that, to obtain a feasible solution, Statement 2 requires to choose a batch such that $\xi_i + b_{i1} \geq \Theta$ holds, while Statement 7 needs to choose a batch such that

$\xi_i + \sum_{j=1}^{s} b_{ij} \geq s \times \Theta$ holds. It raises a question whether we can find such batches in performing these two statements. The following result answers this question.

Theorem 4. *Given a solution Λ_i from the ILP, Algorithm 1 can find a feasible schedule.*

Proof. Given Λ_i obtained by solving the ILP, it follows from Constraint (9) that no matter how the batches in Λ_i are sequenced, we have $\xi_i + \sum_{j=1}^{\eta(i)-1} b_{ij} \geq (\eta(i) - 1) \times \Theta$, $i \in \mathbb{N}_g$. Hence, executing Statement 2 can find a batch in Λ_i as BH_{i1} to make inequality $\xi_i + b_{i1} \geq \Theta$ hold. Then, after Statements 3–5 are executed, we repeatedly execute Statements 6–10. Assume that when we execute Statement 7 with $s = k$, $2 \leq k \leq \eta(i) - 1$, there is no batch in Λ_i to make $\xi_i + \sum_{j=1}^{k} b_{ij} \geq k \times \Theta$ satisfied. This implies that the size of every batch in the current Λ_i is less than Θ due to that $\xi_i + \sum_{j=1}^{k-1} b_{ij} \geq (k-1) \times \Theta$ holds. Moreover, no matter how we sequence the remaining batches in the current Λ_i, we $\xi_i + \sum_{j=1}^{d} b_{ij} < d \times \Theta$, $k \leq d \leq \eta(i) - 1$ must hold, which contradicts the inequality $\xi_i + \sum_{j=1}^{\eta(i)-1} b_{ij} \geq (\eta(i) - 1) \times \Theta$, $i \in \mathbb{N}_g$. Thus, there is no problem for executing Statement 7) in Algorithm 1 and the theorem holds. □

It follows from Theorem 2 and the above discussion that, by the ILP and Algorithm 1 together, we can obtain a schedule such that the production rate of M_{1i} is no less than $\frac{1}{g\beta}$ at any time and the total production rate of Stage 1 is no less than $\frac{1}{\beta}$ at any time in $(0, T_{min}]$. Hence, according to Lemma 1, Stage 2 can operate uninterruptedly in $(0, T_{min}]$.

Note that the ILP together with Algorithm 1 forms a two-step solution method. Figure 4 demonstrates how the two-step solution method is applied to solve a real problem. Specifically, given the batches to be produced, they are partitioned into two parts (i.e., BT1 and BT2) at first. Then, the developed ILP is used to assign the batches to the machines at Stage 1. At last, Algorithm 1 is responsible for sequencing the batches for each machine at Stage 1. In this way, an optimal and feasible schedule is obtained such that Stage 2 can operate uninterruptedly.

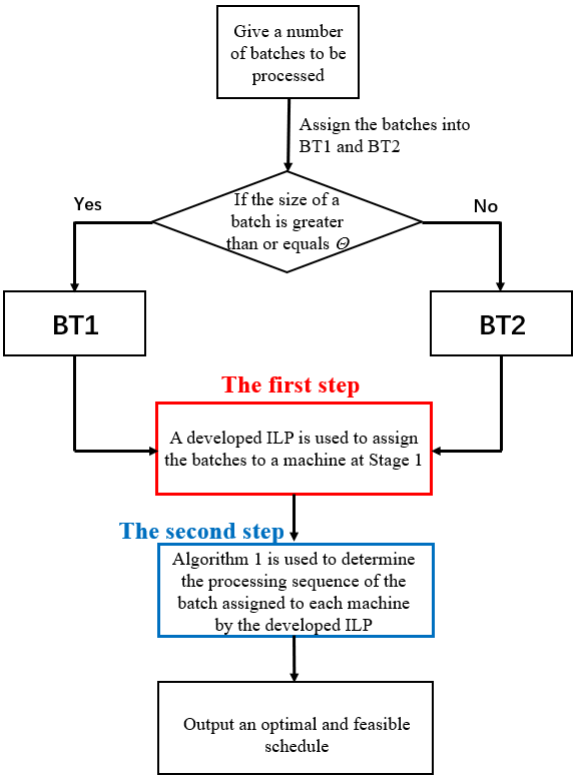

Figure 4. The flowchart of the proposed two-step solution method.

Remark 1. *In this work, for the addressed AGMS, we first analyze the properties of the system and establish the existence condition of feasible solutions by presenting Theorems 1–3. Then, based on the conditions, we develop a two-step solution approach that is implemented by the ILP and Algorithm 1 together. Notice that the problem may be solved in an integrated way by using mathematical programming without establishing the solution existence conditions. However, by doing so, many more binary variables are introduced into the model, resulting in greater difficulty in finding a solution. Thus, by the proposed two-step approach, we greatly reduce the computational burden such that we can find a solution even for large-size application problems, which is demonstrated by the experimental results given in the next section.*

5. Experimental Results

This section uses a real-life AGMS from a company in China as a case problem for validation. As previously described, this AGMS contains five processing steps with the first step and Steps 2–5 forming Stages 1 and 2, respectively; and Stage 2 is the bottleneck and cannot be interrupted. In the experiments, without loss of generality, we just treat Step 2 as the second stage.

For this AGMS, three and five machines are configured for Steps 1 and 2 ($g = 3$ and $h = 5$), respectively. It takes 30 s ($\alpha = 30$ s) for machine M_{1i}, $i \in \mathbb{N}_{\mathbf{3}}$, to complete a product at Step 1, while it takes 55 s ($\mu = 55$ s) for machine M_{2i}, $i \in \mathbb{N}_{\mathbf{5}}$, to complete a product at Step 2. It takes 1800 s ($\delta = 1800$ s) to complete setup for M_{1i}, $i \in \mathbb{N}_{\mathbf{3}}$, while it does not need setup for Step 2. Hence, we obtain $\frac{\alpha}{g} = 10$ s $< \beta = 11$ s and $\Theta = \frac{\delta}{g \times \beta - \alpha} = 600$. Usually, during a scheduling horizon, we have no more than 100 batches for producing and the batch size falls in the interval [50, 1150]. For this case problem, we also have the following property. Let Ω denote the batch size for an arbitrary batch. Then, we have $\Omega \bmod 5 = 0$ or $\Omega \bmod 10 = 0$. With batch size in [50, 1150], batches whose size is less than 600 exist for producing. Then, according to the above parameters, numerical experiments are performed and results are presented to validate the proposed approach.

In the experiments, we use CPLEX to solve the ILP presented in this paper on a laptop with twelve Intel(R) Core(TM) i7-10750H CPU @ 2.60 GHz. The time taken for finding a solution depends on how many batches to be scheduled. When we need to schedule a small number of batches, we can obtain an optimal solution by an exact solution way. However, if there are a large number of batches to schedule, an exact solution method is not efficient and often impossible to find an optimal solution in a given limited time. Let ζ be the number of batches for the AGMS to produce. Given a ζ in [11, 30], we use CPLEX to solve 30 cases and we randomly generate the batch size in [50, 1150] for each case. In total, we generate 600 cases to test the developed ILP.

By experiments, for the cases with $\zeta \in [11, 24]$, optimal solutions can be obtained by the developed ILP for all the cases within 3600 s, while for the cases with $\zeta \in [25, 30]$, within 3600 s, only parts of these cases can be solved such that optimal solutions are obtained. Experimental results for problems with $\zeta \in [11, 24]$ and $\zeta \in [25, 30]$ are summarized in Tables 1 and 2, respectively. Note that Num_{solved} and $Num_{unsolved}$ represent the number of cases that the ILP can and cannot solve within 3600 s, respectively. For the unsolved cases with $\zeta \in [25, 30]$ as shown in Table 2, CPLEX can obtain a solution with a gap between the obtained solution and the lower bound of an optimal one. Such results are summarized in Table 3. From Table 3, we can observe that the gaps are less than 1.5%, implying that CPLEX can obtain very good solutions if optimal solutions cannot be obtained within 3600 s.

Table 1. Experimental results for cases with $\zeta \in [11, 24]$.

ζ	ILP	
	Ave. Running Time (s)	**Num_{solved}**
11	0.15	30
12	0.19	30
13	0.35	30
14	0.17	30
15	0.20	30
16	1.15	30
17	0.58	30
18	0.90	30
19	22.73	30
20	4.64	30
21	7.02	30
22	122.98	30
23	224.28	30
24	118.15	30

Table 2. Experimental results for problems with $\zeta \in [25, 30]$.

ζ	ILP			
	Num_{solved}	**Ave. Running Time (s)**	**$Num_{unsolved}$**	**Running Time (s)**
25	27	366.66	3	3600
26	19	298.31	11	3600
27	23	356.15	7	3600
28	15	124.76	15	3600
29	18	257.70	12	3600
30	18	98.07	12	3600

Table 3. Gaps between current solutions and lower bounds of optimal solutions.

ζ	ILP		
	$Num_{unsolved}$	**Average Gap**	**Maximum Gap**
25	3	1.17%	1.7789%
26	11	1.019%	2.2558%
27	7	0.0094%	0.1115%
28	15	0.8524%	5.2856%
29	12	0.3864%	2.6207%
30	12	0.0083%	0.0106%

For much larger size cases (i.e., $\zeta > 30$), to save time in the experiments, we set a condition to terminate the running of CPLEX as follows by observing the gap between the current solution and a lower bound of an optimal solution. The running of CPLEX terminates if such a gap is less than 1% or this gap cannot be achieved within 3600 s. In this way, we test the proposed approach as follows. Given a $\zeta > 30$, 30 experiments (i.e., 30 cases) are carried out for the ILP. In the experiments, we limit the batches to be produced

being no more than 110 that it is less than 100 in practice. Thus, a total of 2400 cases are generated to test the proposed method.

In the experiments, when a case cannot be solved within 3600 s even for achieving a gap of 1%, a gap between the current solution and the lower bound of an optimal one can be obtained when CPLEX terminates according to the termination condition. Thus, given a ζ, 30 gaps for 30 randomly generated cases can be obtained so as to obtain a boxplot for each $\zeta \in [31, 110]$. In this way, in total, we have 80 boxplots for different situations with different $\zeta \in [31, 110]$ and they are shown in Figure 5. From the boxplots, we can intuitively know the central location and dispersion range of the gaps for each situation, and if the gaps obtained by the ILP for different situations are stable, i.e., check if the ILP can obtain good solutions for different situations stably.

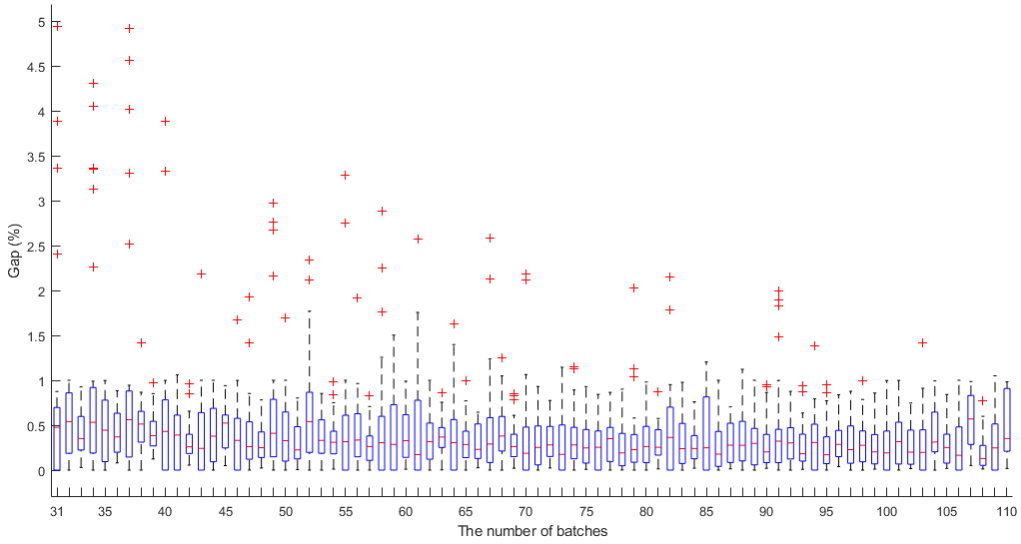

Figure 5. The boxplots for all values of $\zeta \in [31, 110]$.

It follows from the boxplots that for 91.25% of (or 73) situations (boxplots), the median number is no more than 0.5%. For the rest of the seven situations, the median number is just a little larger than 0.5%. This means that for each situation, the median number of gaps is relatively small. Further, from Figure 5, we can intuitively know that most of the gaps for each situation are no more than 1%. Meanwhile, by observing the shapes of boxplots, the gaps obtained by the ILP are quite small and stable for different situations. Further, we observe that it can solve 96.83% of the cases by the ILP for a gap of 1% within 3600 s, while only 3.17% of the cases (i.e., 76 cases) cannot be solved. For these unsolved cases, we have that the average gap of these cases is 2.1827% and the maximum gap is 4.9416%. Therefore, based on the data analysis for the much larger size problems, good solutions for different situations can be obtained by CPLEX within a reasonable time, i.e., the established approach is efficient and effective in terms of practical applications.

To understand the reason behind the data analysis results, we also try to analyze the ILP. For real application problems from an AGMS in China, the size of the ILP is directly affected by the number of batches to be produced since the number of machines at Step 1 is determined. Then, by adjusting the number of batches in BT1 and BT2, the changes of the number of variables and constraints of the ILP are shown in Figures 6 and 7, respectively. From Figures 6 and 7, we can observe that the number of variables and constraints are linear with the number of batches for producing. This means that for real application problems, as the number of batches increases, the number of variables or constraints increases linearly rather than exponentially. This leads to the fact that even if the size of the problems increases, the ILP is still applicable for obtaining good solutions.

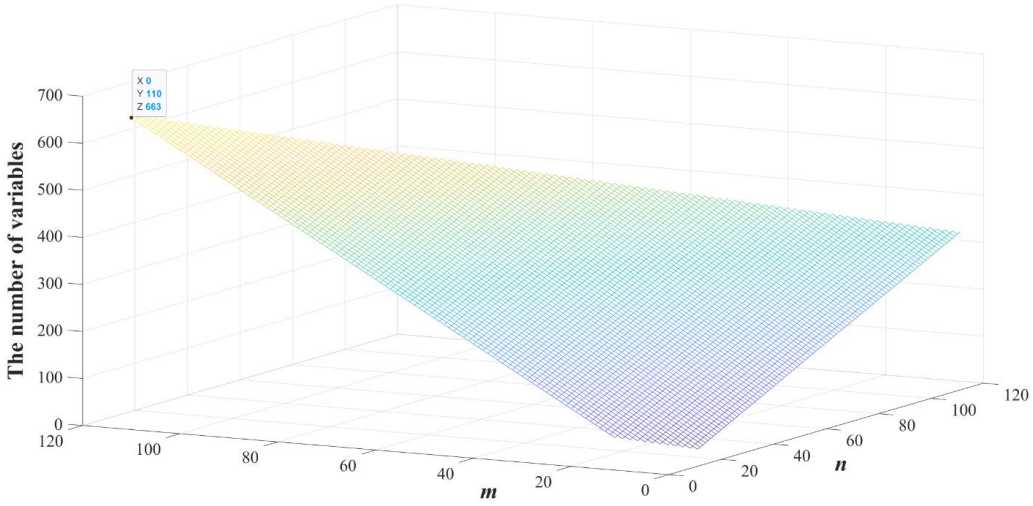

Figure 6. The number of variables of the developed ILP for different cases.

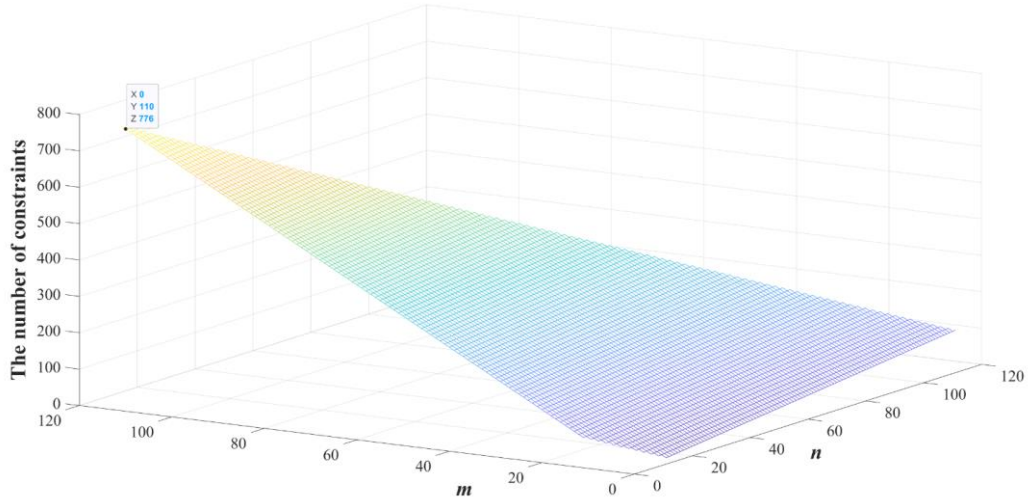

Figure 7. The number of constraints of the developed ILP for different cases.

As we have emphasized in the Introduction, if the problem is formulated by a mathematical programming model without the established feasibility conditions, we need to identify all the products for sequencing instead of just identifying the batches, leading to a huge number of binary variables being necessary to form the model. However, by the proposed two-step solution method, to formulate the ILP, we need to identify and allocate the batches only so that there are much fewer binary variables and constraints. Hence, the computational complexity for solving the problem is greatly reduced. Moreover, by Algorithm 1, we can efficiently sequence the batches for producing by a machine. This is why the proposed method can find good solutions within a reasonable time.

In practice, often a mid-term schedule is generated for an AGMS and this is hierarchically carried out in a two-step way, as follows:

(1) At the first step, according to the customer orders, a relatively rough schedule is developed for a relatively long-time horizon that typically lasts for a month. This scheduling horizon is divided into several uniform slots with each slot lasting for 5–7 days. Then, with the capacity of the AGMS considered, this schedule determines the batches to be produced in each time slot by using simple heuristic algorithms, such that the batches can be produced by the due date.

(2) At the second step, detailed schedules are generated to realize the rough schedule for each time slot. Such detailed schedules are also called short-term schedules. For each detailed schedule, it needs to schedule all the activities in an AGMS just as performed in this paper such that the process constraints are satisfied and the productivity of the system is maximized to optimize the profit.

In this way, a mid-term schedule is formed by the rough schedule and the detailed schedules together. Such a mid-term schedule is repeatedly updated to govern the operation of an AGMS and this is the routine work for an AGMS. It can be seen from the development of the scheduling method presented in this paper, it is not an easy job to generate a detailed schedule for an AGMS. However, in practice, currently this job is done manually by the production engineers. This is a great burden for such an engineer. Moreover, it is very hard to manually generate a detailed schedule to satisfy process constraints and optimize productivity. The proposed approach in this paper provides an effective way to solve this problem.

6. Conclusions

It is well known that auto models are greatly diversified, leading to a great number of glass products for autos such that automotive glass manufacturing becomes an important part of the automotive industry. An AGMS can be treated as a two-stage flow-shop. In an AGMS, there are strict process constraints: (1) setup with significant time is necessary for the machines at Stage 1; and (2) the machines at the second stage cannot be interrupted. Furthermore, an AGMS operates in customization mode such that there are frequent changeovers for producing different product types. These properties make the scheduling problem of AGMS very challenging. This work investigates this challenging problem by establishing feasible schedule existence conditions. With the conditions, a two-step approach is developed to solve the problem. At the first step, an ILP model is built to allocate the batches to machines at Stage 1. Then, a simple algorithm is given to sequence the batches for each machine. Thanks to the conditions, the size of the ILP is significantly reduced such that good solutions can be obtained in a reasonable time even for large application problems. Extensive experiments are also used to demonstrate the efficiency and effectiveness of the developed approach. For small-size problems, with the number of batches being no more than 24, by CPLEX, ILP can find optimal solutions within 3600 s. For problems with larger sizes, it fails to obtain optimal solutions for all cases within 3600 s. Nevertheless, we set 3600 s as the longest time for obtaining a solution and a gap of 1% for the lower bound of solutions. Then, results show that CPLEX can solve 96.83% cases. Moreover, we can obtain good solutions with the maximum gap 4.9416% for the unsolved cases.

Work in process has important effects on the cost for an AGMS. Hence, it is meaningful to minimize the temporary inventory between the stages. However, we do not take this factor into consideration. In the future, we aim to extend the proposed approach by considering this factor.

Author Contributions: Conceptualization, Y.Q. and N.W.; methodology, Y.Q. and Z.L.; software, Y.Q.; validation, A.-A.E.-T., H.K.; formal analysis, Y.Q.; investigation, N.W. and H.K.; resources, N.W.; writing—original draft preparation, Y.Q.; writing—review and editing, N.W., Z.L. and A.M.A.-A.; supervision, N.W.; funding acquisition, A.M.A.-A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Plan for Science, Technology and Innovation (MAARIFAH), King Abdulaziz City for Science and Technology, Kingdom of Saudi Arabia, under Grant 2-17-01-001-0008.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature

Abbreviations:

AGMS	Automotive glass manufacturing system
BT1	Batch Type 1
BT2	Batch Type 2
ILP	Integer linear program
PVB	Polyvinyl Butyral
TMFS	Two-machine flow shop
TSFS	Two-stage flow-shop
TSFFS	Two-stage flexible flow-shop

Notation:

BH_{ij}	The j-th processed batch at M_{1i}
b_{ij}	Batch size of the j-th batch (i.e., BH_{ij}) to be processed at M_{1i}
$f_1(x)$	$= a \times x/(b \times x + c)$, $a > 0$, $b > 0$, and $c > 0$
$f_2(x)$	$= x/(\alpha \times x + u \times \delta)$
g	The number of machines at Stage 1
h	The number of machines at Stage 2
M_{1i}	The i-th machine at Stage 1
M_{2j}	The j-th machine at Stage 2
m	The number of batches in BT2 whose batch size is smaller than Θ
$\mathbb{N}_k$	$= \{1, 2, \ldots, k\}$
Num_{solved}	The number of cases that ILP can solve within 3600 s
$Num_{unsolved}$	The number of cases that ILP cannot solve within 3600 s
O_i ($\geq \Theta$)	The size of the i-th batch in BT1
T_{ij}	Time point when BH_{ij} has just been completed at M_{1i}
T_{min}	$= min(T_{i\eta(i)} \mid i \in \mathbb{N}_g)$
α	Time units to complete a product by a machine at Stage 1
β	$= \mu/h$
δ	Setup time for a machine at Stage 1
Θ	$= \delta/(g\beta - \alpha)$
$\eta(i)$	The number of processed batches at M_{1i}
Λ_i	Set of batches assigned to M_{1i} obtained by solving an ILP except the one belonging to BT2 with $z_{ij} = 1$
μ	Processing time of a machine at Stage 2
ϑ_{it}	Average productivity at M_{1i} during time interval $[0, t]$
ζ	The number of batches to be scheduled
Φ_j ($< \Theta$)	The size of the j-th batch in BT2

Variables in the developed ILP:

x_{ij}	Binary variable, 1 if the j-th batch in BT1 is processed at M_{1i}, zero otherwise
y_{ij}	Binary variable, 1 if the j-th batch in BT2 is processed at M_{1i}, zero otherwise
z_{ij}	Binary variable, 1 if the j-th batch with size Φ_j is processed at M_{1i} at last, zero otherwise
ξ_i	Integer variable representing the number of extra products to be processed at M_{1i} with their type being same as the one in BH_{i1}
Γ	Time needed to complete the processing of all batches at Stage 1

References

1. Gharbi, A.; Ladhari, T.; Msakni, M.K.; Serairi, M. The two-machine flow-shop scheduling problem with sequence-independent setup times: New lower bounding strategies. *Eur. J. Oper. Res.* **2013**, *231*, 69–78. [CrossRef]
2. Lee, T.; Loong, Y. A review of scheduling problem and resolution methods in flexible flow shop. *Int. J. Ind. Eng. Comput.* **2019**, *10*, 67–88. [CrossRef]
3. Zhao, Z.; Zhou, M.; Liu, S. Iterated Greedy Algorithms for Flow-Shop Scheduling Problems: A Tutorial. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 1941–1959. [CrossRef]
4. Vasilis, S.; Nikos, N.; Kosmas, A.; Dimitris, M. A toolbox of agents for scheduling the paint shop in bicycle industry. *Procedia CIRP* **2022**, *107*, 1156–1161. [CrossRef]
5. Rooeinfar, R.; Raissi, S.; Ghezavati, V. Stochastic flexible flow shop scheduling problem with limited buffers and fixed interval preventive maintenance: A hybrid approach of simulation and metaheuristic algorithms. *Simulation* **2019**, *95*, 509–528. [CrossRef]

6. Li, J.; Bai, S.; Duan, P.; Sang, H.; Han, Y.; Zheng, Z. An improved artificial bee colony algorithm for addressing distributed flow shop with distance coefficient in a prefabricated system. *Int. J. Prod. Res.* **2019**, *57*, 6922–6942. [CrossRef]
7. Missaoui, A.; Ruiz, R. A parameter-Less iterated greedy method for the hybrid flowshop scheduling problem with setup times and due date windows. *Eur. J. Oper. Res.* **2022**, *303*, 99–113. [CrossRef]
8. Fattahi, P.; Hosseini, S.M.H.; Jolai, F. A mathematical model and extension algorithm for assembly flexible flow shop scheduling problem. *Int. J. Adv. Manuf. Technol.* **2013**, *65*, 787–802. [CrossRef]
9. Umam, M.S.; Mustafid, M.; Suryono, S. A hybrid genetic algorithm and tabu search for minimizing makespan in flow shop scheduling problem. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 7459–7467. [CrossRef]
10. Lalas, C.; Mourtzis, D.; Papakostas, N.; Chryssolouris, G. A Simulation-Based Hybrid Backwards Scheduling Framework for Manufacturing Systems. *Int. J. Comput. Integr. Manuf.* **2006**, *19*, 762–774. [CrossRef]
11. Papakostas, N.; Chryssolouris, G. A Scheduling Policy for Improving Tardiness Performance. *Asian Int. J. Sci. Technol.* **2009**, *2*, 79–89.
12. Miyata, H.H.; Nagano, M.S. The blocking flow shop scheduling problem: A comprehensive and conceptual review. *Expert Syst. Appl.* **2019**, *137*, 130–156. [CrossRef]
13. 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]
14. Mirabi, M.; Fatemi Ghomi, S.M.T.; Jolai, F. A two-stage hybrid flowshop scheduling problem in machine breakdown condition. *J. Intell. Manuf.* **2013**, *24*, 193–199. [CrossRef]
15. Gupta, D.; Goel, S.; Mangla, N. Optimization of production scheduling in two stage flow shop scheduling problem with m equipotential machines at first stage. *Int. J. Syst. Assur. Eng. Manag.* **2021**, *13*, 1162–1169. [CrossRef]
16. Chen, Z.; Zheng, X.; Zhou, S.C.; Liu, C.; Chen, H.P. Quantum-inspired ant colony optimization algorithm for a two-stage permutation flow shop with batch processing machines. *Int. J. Prod. Res.* **2019**, *58*, 5945–5963. [CrossRef]
17. Zheng, X.; Zhou, S.C.; Xu, R.; Chen, H.P. Energy-efficient scheduling for multi-objective two-stage flow shop using a hybrid ant colony optimization algorithm. *Int. J. Prod. Res.* **2019**, *58*, 4103–4120. [CrossRef]
18. Dong, J.; Pan, H.; Ye, C.; Tong, W.; Hu, J. No-wait two-stage flowshop problem with multi-task flexibility of the first machine. *Inf. Sci.* **2021**, *544*, 25–38. [CrossRef]
19. Jemmali, M.; Hidri, L.; Alourani, A. Two-stage hybrid flowshop scheduling problem with independent setup times. *Int. J. Simul. Model.* **2022**, *21*, 5–16. [CrossRef]
20. Lei, D.M.; Xi, B.J. Diversified teaching-learning-based optimization for fuzzy two-stage hybrid flow shop scheduling with setup time. *J. Intell. Fuzzy Syst.* **2021**, *41*, 4159–4173. [CrossRef]
21. Gerpott, F.T.; Lang, S.; Reggelin, T.; Zadek, H.; Chaopaisarn, P.; Ramingwong, S. Integration of the A2C algorithm for production scheduling in a two-stage hybrid flow shop environment. *Procedia Comput. Sci.* **2022**, *200*, 585–594. [CrossRef]
22. Han, J.-H.; Lee, J.-Y. Heuristics for a two-stage assembly-type flow shop with limited waiting time constraints. *Appl. Sci.-Basel* **2021**, *11*, 11240. [CrossRef]
23. Pourhejazy, P.; Cheng, C.Y.; Ying, K.C.; Nam, N.H. Meta-Lamarckian-based iterated greedy for optimizing distributed two-stage assembly flowshops with mixed setups. *Ann. Oper. Res.* **2022**. [CrossRef]
24. Talens, C.; Fernandez-Viagas, V.; Perez-Gonzalez, P.; Framinan, J.M. New efficient constructive heuristics for the two-stage multi-machine assembly scheduling problem. *Comput. Ind. Eng.* **2020**, *140*, 106223. [CrossRef]
25. Zhang, Z.; Tang, Q. Integrating flexible preventive maintenance activities into two-stage assembly flow shop scheduling with multiple assembly machines. *Comput. Ind. Eng.* **2021**, *159*, 107493. [CrossRef]
26. 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]
27. Kalczynski, P.J.; Kamburowski, J. An empirical analysis of heuristics for sovling the two-machine flow shop problem with release times. *Comput. Oper. Res.* **2012**, *39*, 2659–2665. [CrossRef]
28. Liu, P.; Lu, X. A best possible on-line algorithm for two-machine flow shop scheduling to minimize makespan. *Comput. Oper. Res.* **2014**, *51*, 251–256. [CrossRef]
29. Agrebi, I.; Jemmali, M.; Alquhayz, H.; Ladhari, T. Metaheuristic algorithms for the two-machine flowshop scheduling problem with release dates and blocking constraint. *J. Chin. Inst. Eng.* **2021**, *44*, 573–582. [CrossRef]
30. Schaller, J.; Valente, J. Branch-and-bound algorithms for minimizing total earliness and tardiness in a two-machine permutation flow shop with unforced idle allowed. *Comput. Oper. Res.* **2019**, *109*, 1–11. [CrossRef]
31. Bank, M.; Fatemi Ghomi, S.M.T.; Jolai, F.; Behnamian, J. Two-machine flow shop total tardiness scheduling problem with deteriorating jobs. *Appl. Math. Model.* **2012**, *36*, 5418–5426. [CrossRef]
32. Cheng, M.; Tadikamalla, P.R.; Shang, J.; Zhang, S. Bicriteria hierarchical optimization of two-machine flow shop scheduling problem with time-dependent deteriorating jobs. *Eur. J. Oper. Res.* **2014**, *234*, 650–657. [CrossRef]
33. Qiao, Y.; Wu, N.Q.; He, Y.F.; Li, Z.W.; Chen, T. Adaptive genetic algorithm for two-stage hybrid flow-shop scheduling with sequence-independent setup time and no-interruption requirement. *Expert Syst. Appl.* **2022**, *208*, 1–13. [CrossRef]
34. Fridman, I.; Pesch, E.; Shafransky, Y. Minimizing maximum cost for a single machine under uncertainty of processing times. *Eur. J. Oper. Res.* **2020**, *286*, 444–457. [CrossRef]

35. Ghaleb, M.; Taghipour, S.; Sharifi, M.; Zolfagharinia, H. Integrated production and maintenance scheduling for a single degrading machine with deterioration-based failures. *Comput. Ind. Eng.* **2020**, *143*, 106432. [CrossRef]
36. Goldengorin, B.; Romanuke, V. Online heuristic for the preemptive single machine scheduling problem to minimize the total weighted tardiness. *Comput. Ind. Eng.* **2021**, *155*, 107090. [CrossRef]
37. Luo, W.C.; Xu, Y.; Tong, W.T.; Lin, G.H. Single-machine scheduling with job-dependent machine deterioration. *J. Sched.* **2019**, *22*, 691–707. [CrossRef]
38. Mor, B.; Mosheiov, G. Minmax due-date assignment on a two-machine flowshop. *Ann. Oper. Res.* **2021**, *305*, 191–209. [CrossRef]
39. Perez-Gonzalez, P.; Framinan, J.M. Single machine interfering jobs problem with flowtime objective. *J. Intell. Manuf.* **2018**, *29*, 953–972. [CrossRef]
40. Wan, L.; Yuan, J. Single-machine scheduling with operator non-availability to minimize total weighted completion time. *Inf. Sci.* **2018**, *445–446*, 1–5. [CrossRef]
41. Telles, E.S.; Lacerda, D.P.; Morandi, M.I.W.; Piran, F.A.S. Drum-buffer-rope in an engineering-to-order system: An analysis of an aerospace manufacturer using data envelopment analysis (DEA). *Int. J. Prod. Econ.* **2020**, *222*, 107500. [CrossRef]
42. Telles, E.S.; Lacerda, D.P.; Morandi, M.I.W.; Ellwanger, R.; Souza, F.B.; Piran, F.S. Drum-Buffer-Rope in an engineering-to-order productive system: A case study in a Brazilian aerospace company. *J. Manuf. Technol. Manag.* **2022**, *33*, 1190–1209. [CrossRef]