A Q-learning Algorithm for Two-Stage Hybrid Flow Shop Scheduling

Yanjun Lu[1], Zhaoting Liu[2], and Qian Zhang[1]

[1]Jiangsu Open University
[2]State Grid Electric Power Research Institute Nanjing Branch

March 12, 2024

Abstract

With the increasing demand for customization, flow shop scheduling tends to process multi-variety small batch products. Thus, there are frequent switches between batches. Frequent switchovers between different sets are required during batch changeover, where machine setups may be necessary. This paper investigates scheduling optimization and work-in-process inventory optimization problems in a hybrid flow shop with rolling processing requirements. Developing continuous processing conditions to meet rolling processing requirements presents a significant challenge. In this work, continuous processing conditions are derived. Subsequently, a linear programming model is designed to find an optimal solution for this scheduling problem. Formulas for calculating work-in-process inventory are presented. To enhance work-in-process storage, a greedy algorithm with bubble sort and a Q-learning algorithm with a modified ϵ-policy are proposed to find an optimized product sequence and reduce work-in-process inventory, respectively. Numerical experiments evaluate the effectiveness and efficiency of the proposed methods.

Received <day> <Month>, <year>; | Revised <day> <Month>, <year>; | Accepted <day> <Month>, <year>

DOI: xxx/xxxx

JOURNAL

A Q-learning Algorithm for Two-Stage Hybrid Flow Shop Scheduling

Yanjun Lu*[1] | Zhaoting Liu[2] | Qian Zhang[1]

[1]School of Information Technology, Jiangsu Open University, Jiangsu, China
[2]State Grid NanRui Nanjing Control System Co., Ltd., State Grid NanRui Group, Jiangsu, China

Correspondence
*Yanjun Lu, Corresponding address. Email: luyanjun2018@126.com

Present Address
399 Jiangdong North Road, Gulou District, Nanjing City, Jiangsu Province, China

Abstract

With the increasing demand for customization, flow shop scheduling tends to process multi-variety small batch products. Thus, there are frequent switches between batches. Frequent switchovers between different sets are required during batch changeover, where machine setups may be necessary. This paper investigates scheduling optimization and work-in-process inventory optimization problems in a hybrid flow shop with rolling processing requirements. Developing continuous processing conditions to meet rolling processing requirements presents a significant challenge. In this work, continuous processing conditions are derived. Subsequently, a linear programming model is designed to find an optimal solution for this scheduling problem. Formulas for calculating work-in-process inventory are presented. To enhance work-in-process storage, a greedy algorithm with bubble sort and a Q-learning algorithm with a modified ϵ-policy are proposed to find an optimized product sequence and reduce work-in-process inventory, respectively. Numerical experiments evaluate the effectiveness and efficiency of the proposed methods.

KEYWORDS:
flow shop scheduling; rolling processing requirement; setup time;Q-learning

1 | INTRODUCTION

Due to the escalating global energy crisis and worsening environmental pollution, the automotive industry, as a critical sector of energy consumption, plays a crucial role. The introduction of new energy vehicles contributes to reducing dependence on non-renewable energy sources[1], and in recent years, their demand has been steadily increasing, presenting new development opportunities for automotive component companies. Currently, the manufacturing of automotive glass is undergoing a transition from traditional large-scale processing to small-scale processing. However, in this process, switching between different types of products often requires a significant amount of time for equipment debugging. To address this issue, it is advisable to preset the debugging time during the equipment debugging stage. This ensures that the machine can seamlessly transition between different products, achieving the goal of continuous production. This approach contributes to enhancing production efficiency, enabling the automotive glass manufacturing sector to adapt more flexibly to changes in market demand.

In accordance with the actual conditions of the glass manufacturing workshop, the research problem can be defined as a two-stage hybrid flow shop scheduling optimization problem. Automotive glass manufacturers produce a variety of glass types, including windshields for sedans, rear windshields for vehicles, and side window glass for vehicles. Glasses intended for different purposes undergo distinct production processes, and even glasses serving the same purpose have different production processes

[0]**Abbreviations:** WIP, work-in-process

due to varying product adaptations, presenting challenges and difficulties to the production process. Typically, the production of automotive glass involves five main processes: pretreatment, bending, polyvinyl butyral (PVB) stretching and coating, high pressure, and final inspection packaging.During the first-stage pretreatment process, when transitioning from processing one batch to another, equipment shutdown is required for debugging, indicating the necessity of setup time in the first stage. Through the judicious arrangement of production plans, setup time for the bending process of other products can be omitted. In summary, bending constitutes the first-stage process, while the remaining processes belong to the second stage.

In a typical manufacturing environment, a hybrid flow shop is applied in various production scenarios, including chemical, electrical appliances, and other small-batch customized industrial products[2,3,4,5]. Among the various scheduling problems in flow shops, the two-stage hybrid flow shop scheduling problem has been proven to be an NP-hard problem[6] . Due to its practical and theoretical significance, both academia and industry are continuously researching the two-stage hybrid flow shop scheduling problem. However, issues with interruptible process constraints and rolling production requirements have not been adequately explored[3]. In this paper, motivated by the scheduling challenges of real-world glass manufacturing workshops that meet interruptible process constraints and rolling production demands, we investigate a two-stage hybrid flow shop scheduling problem, considering optimization for both makespan and work-in-process (WIP) inventory.

1.1 | Literature Review

In the past few decades, scheduling problems in industrial systems have garnered widespread attention from academia and industry due to their indispensable role in improving production efficiency and reducing production costs[7,8,9,10,11,12,13]. The research by Wang et al.[8] focuses on minimizing energy consumption in a flexible job shop, characterized by its dynamic energy features. The study conducted by Wang et al.[9] concentrates on achieving both makespan and total energy consumption in a two-stage hybrid flow shop where parallel machines exist in both the first and second stages. Li et al.[10] addresses the issue of limited waiting time in a two-stage hybrid flow shop problem and proposes a hybrid membrane computing metaheuristic to minimize makespan. The research by Fan et al.[11] aims to simultaneously minimize the arrival rate of different parts at the assembly stage and the on-time delivery rate in a two-stage hybrid flow shop scheduling problem.

In addition to scheduling issues in the two-stage hybrid flow shop setting, the current trend in research focuses on increasing the complexity and practical relevance of the model[14,15,16,17]. Researchers have introduced constraints such as multi-process task scheduling[15], resource constraints[14,18] , interruptible process constraints[19] , and setup time[20,21]. These typical constraints extend the standard two-stage hybrid flow shop model to more complex yet practical problems. In situations where there is no need to wait for processing due to high decay rates, these problems resemble uninterrupted processing.Specifically, the branch-and-bound algorithm proposed by[22] has successfully addressed large-scale two-stage no-wait problems. Its application scope is broader, and computational efficiency is higher compared to[23]. Furthermore, Dong et al.[24] considered a no-wait two-stage flow shop scheduling problem with the first stage having multi-task flexibility. The objective is to minimize the maximum completion time of all jobs.

Considering its significance and complexity, the two-stage hybrid flow-shop scheduling problem has been a focal point of research for an extended period. Sheikh et al.[21] proposed a mixed binary linear optimization model and an unconstrained mathematical formulation to minimize the makespan in a multi-stage assembly flow shop. Cai et al.[25] introduced a collaborative variable search method with seven neighborhood structures and two global search operators to optimize the two-stage hybrid flow shop, aiming at maximizing the total agreement index and fuzzy makespan. Wang et al.[26] formulated a mixed-integer linear program for calculating the total processing time of all jobs in the two stages. Zhao et al.[27] investigated a non-waiting flow shop scheduling problem involving sequence-dependent setup times with the goal of minimizing the completion time. However, the product processing sequence in the aforementioned scheduling problem is fixed and cannot adapt to dynamic changes in production plans and product types.

Optimizing work-in-process (WIP) inventory has emerged as a critical objective in scheduling optimization and minimizing processing time[28,29]. Fan et al.[11] proposed a mutated firefly algorithm for addressing the two-stage hybrid flow-shop scheduling problem with two objective functions. Wang et al.[26] employed a technique known as lot streaming, dividing large batches of products into multiple smaller batches to reduce WIP inventory. Scheduling strategies based on priority, cycling, and WIP were investigated to address both small-scale and large-scale flexible production lines, enhancing product productivity[30]. Zhu et al.[31] and Li et al.[32] both investigated a cluster tool scheduling problem in manufacturing systems to prevent deadlock. The objective of this problem was to optimize the maximum completion time and WIP inventory, and a nonlinear programming model was established. However, the methods mentioned in these articles cannot be directly applied to the two-stage hybrid flow-shop

scheduling problem. This paper explores the two-stage hybrid flow-shop scheduling problem with the aim of optimizing the total completion time and WIP inventory for all jobs.

From the above literature, it can be observed that most research on two-stage hybrid flow shop scheduling focuses on developing heuristic or optimal solutions to address general scheduling problems. The actual manufacturing demand is shifting towards customer customization, and production scheduling issues need to consider multiple constraints, such as resource constraints, interruptible process constraints, and setup time. Considering multiple constraints in scheduling problems poses challenges and complexities. In this paper, we investigate a two-stage hybrid flow shop problem with setup time, interruptible processing constraints, and rolling production requirements.

2 | PROBLEM DESCRIPTION AND FORMULATION

2.1 | Problem Description

The two-stage hybrid flow shop scheduling problem can be defined by the following characteristics: 1) The first and the second stages are equipped with $m \geq 1$ machines and $n \geq 1$ machines, respectively, where $m \in \mathbb{N}^+$, $n \in \mathbb{N}^+$; 2) Set-up time is required in the first stage but can be ignored in the second stage; 3) Initially, all batches undergo processing in the first stage and are then seamlessly transferred to the second stage, adhering to interruptible process constraints; 4)Post-processing debugging is necessary after each batch to ensure immediate production; and 5) The objective is to minimize the total completion time in the first stage and reduce work-in-process (WIP) inventory across the entire two-stage process.

To specify the scheduling problem, the following assumptions are made based on the actual manufacturing system: 1) The set-up time for machines in the first stage is denoted by σ; 2) There are parallel machines in each stage, where M_{1i} denotes the $i\text{-}th$ machine in the first stage($i \in \mathbb{N}_m^+$), and M_{2j} represents the $j\text{-}th$ machine in the second stage($j \in \mathbb{N}_n^+$); 3) At any given time, each machine can process only one product. Processing a single product on machine M_{1i} takes α time, while on M_{2j} it takes μ time. Since the set-up time for machines in the second stage is negligible, we can treat the n machines as a single device. Therefore, this implies that the time required to complete a single job in the second stage is $\psi = \mu/n$; 4) The initial state of the system is known, including the the processing status and quantity of jobs on the machines in the initial state, the total inventory r_i, the batches of jobs to be processed m_i, and the amount of jobs b_{ij}; 5) The same batch of jobs has the same product type, while different batches of jobs have different product types; 6) The production environment is stable, and machine failures and other emergencies are not taken into account; 7) The production rate of jobs is higher in the first stage than in the second stage, *i.e.*, $\alpha/m < \varphi$. However, machines in the first stage require set-up time. As a result, the production rate of jobs in the first stage will always be lower than in the second stage, leading to an imbalance in production.

2.2 | Feasibility Analysis of Scheduling

By appropriately assigning tasks to machines in the first stage, interruptible process constraints can be met. This allows for the resolution of the two-stage hybrid flow shop scheduling problem, enabling all machines to achieve maximum utilization. The calculation formulas for the completion time of one batch of jobs, denoted as T_{ij}, and the rolling production time, represented by H_{ij}, are as follows.

$$T_{ij} = \alpha * \sum_{1}^{j} b_{ij} + (j-1) * \sigma, i \in \mathbb{N}_m^+, j \in \mathbb{N}_n^+ \tag{1}$$

$$H_{ij} = T_{ij} + \sigma, i \in \mathbb{N}_m^+, j \in \mathbb{N}_n^+ \tag{2}$$

where b_{ij} represents the quantity of the $j\text{-}th$ batch of products processed by M_{1i} in the first stage. Simultaneously, $m(i)$ denotes the number of processing batches allocated to M_{1i}. T_{ij} represents the completion time for processing the $j\text{-}th$ batch of products on M_{1i}, while H_{ij} indicates the processing time for the $j\text{-}th$ batch of products on M_{1i} to meet the rolling production requirements. It is worth nothing that b_{i0} represents the initial batch in the system , $b_{i0} = 0$. $T_{im}(i)$ represents the completion time for M_{1i} processing the $j\text{-}th$ batch of products, and $H_{im}(i)$ represents the time during which M_{1i} can produce jobs in a rolling manner. In the time interval $[0, t]$, let $T_{min} = min(T_{im}(i))$, where $i \in \mathbb{N}_m^+$. v_{it} denotes the average production rate of processing jobs on M_{1i}, where $i \in \mathbb{N}_m^+$. Based on the above analysis, we can derive the following lemma, which outliness the conditions for interruptible processing.

Lemma 1. If $\sum_1^m \geq 1/\psi$ holds within the time interval $\left[0, T_{min}\right]$, then the two-stage hybrid flow shop will have the capability to maintain interruptible processing for any $t \in \left[0, T_{min}\right]$.

Proof: When the production rate $\sum_1^m$ of products in the first stage is greater than or equal to the reciprocal of products in the second stage $1/\psi$, that is $\sum_1^m \geq 1/\psi$, the products in the first and second stages can maintain interruptible processing. Therefore, this lemma holds.

Lemma 1 indicates that, within the time interval $\left[0, T_{min}\right]$, continuous processing of the two stages can be guaranteed as long as the condition $\left[0, T_{min}\right]$ for processing jobs in the first stage is satisfied. Subsequently, through a predetermined machine set-up time, the rolling production of the two-stage hybrid flow shop scheduling can be realized. Let $\Delta p = \sigma/(m\psi - \alpha)$, and the following lemma can be derived.

Lemma 2. $\sum_1^m \geq 1/\psi$ holds if any batch satisfies $b_{ij} \geq \Delta p$, $i \in \mathbb{N}_m^+$, $j \in \mathbb{N}_{m(i)}^+$, for any $t \in \left(0, T_{min}\right]$.

Proof: Given that the production rate of jobs in the first stage is greater than that in the second stage, *i.e.*, $\alpha/m < \psi$, and considering that the initial state of the system satisfies the continuous processing of jobs between the two stages, at time $t_0 \leq \vartheta_i\alpha + \sigma$, the average production rate is $\upsilon_{it} \geq 1/\alpha > 1/m\psi$ on M_{1i}. Here, ϑ_i represents the quantity of products being processed on M_{1i} in the initial state, where $i \in \mathbb{N}_m^+$. Additionally, r_{1i} denotes the inventory on Machine M_{1i} in the first stage, where $i \in \mathbb{N}_m^+$.

When the time t is within the range $\vartheta_i\alpha + \sigma < t_1 \leq H_{i1} - \sigma$, we can derive that the average production rate υ_{it_1} on M_{1i} is $\left[t_1 - (\vartheta_i\alpha + \sigma)\right]/\left[t_1 - (\vartheta_i\alpha + \sigma)\right] = 1/\alpha > 1/m\psi$. This confirms the results obtained under this condition.

When the time t is within the range $H_{ik} - \sigma < t_2 \leq H_{ik}$, $1 \leq k \leq m(i) \geq 2$, based on Formulas 1 and 2, we can obtain that the average production rate is $\upsilon_{it_2} = (r_{1i} + \vartheta_i + \sum_1^k b_{ik})/t_2$. Let $b_{ij} \geq \Delta p$, since $\upsilon_{it_0} = (r_{1i} + \vartheta_i)/(\vartheta_i\alpha + \sigma) \geq 1/\alpha$, we have $r_{1i} + \vartheta_i \geq (\vartheta_i\alpha + \sigma)/\alpha$, $t_2 - (H_{ik} - \sigma) < \sigma$, then $\upsilon_{it_2} = \left[(\vartheta_i\alpha + \sigma)/\alpha + k\Delta p\right]/(\vartheta_i\alpha + k\Delta p\alpha + k\alpha + \sigma) = 1/\alpha\left[1 - k\Delta p/(k\Delta p\alpha + k\alpha)\right] = \sigma/(\Delta p\alpha + \sigma)$. This confirms the results obtained under this condition.

When the time t is within the range $H_{ik} < t_3 < H_{ik+1} - \sigma$, $1 \leq k \leq m(i) - 1$, $m(i) \geq 2$, based on Formulas 1 and 2, we have $\upsilon_{it_3} = [r_{1i} + \vartheta_i + \sum_1^k b_{ik} + (t_3 - (\vartheta_i\alpha + \alpha\sum_1^k b_{ik} + (k+1)\sigma))/\alpha]/t_3$. Since $\upsilon_{it_0} = (r_{1i} + \vartheta_i)/t_0 = (r_{1i} + \vartheta_i)/(\vartheta_i\alpha + \sigma) \geq 1/\alpha$, $r_{1i} + \vartheta_i \geq (\vartheta_i\alpha + \sigma)/\alpha$ holds. Let $b_{ik} \geq \Delta p$, the average production rate υ_{it_3} can be simplified $\upsilon_{it_3} \geq [(t_3 - \alpha k\Delta p - k\sigma)/\alpha + k\Delta p]/t_3$ $= 1/\alpha(1 - k\sigma/t_3)$. Since $k\sigma/t_3 < (k+1)\sigma/t_3 = (k+1)\sigma/[\alpha(k+1)\Delta p + (k+1)\sigma] = \sigma/(\alpha\Delta p + \sigma)$, $\upsilon_{it_3} \geq 1/\alpha(1 - \sigma/(\alpha\Delta p + \sigma))$ $\geq 1/m\psi$ holds.

Hence, the production rate for the first stage is $\sum_1^m \geq 1/\psi$, for any $t \in \left(0, T_{min}\right]$.

Corollary 1. $\sum_1^m \geq 1/\psi$ holds if all batches satisfy $\sum_1^j b_{ij} \geq j\Delta p$, $i \in \mathbb{N}_m^+$, $j \in \mathbb{N}_{m(i)}^+$, for any $t \in \left(0, T_{min}\right]$.

Proof: It follows immediately from Lemma 2.

According to Lemma 2 and Corollary 1, any batch satisfies condition $b_{ij} \geq \Delta p$, $i \in \mathbb{N}_m^+$, $j \in \mathbb{N}_{m(i)}^+$, while all batches meet with condition $\sum_1^j b_{ij} \geq j\Delta p$, $i \in \mathbb{N}_m^+$, $j \in \mathbb{N}_{m(i)}^+$.

In a two-stage hybrid flow shop, jobs can be processed continuously. Therefore, when the conditions of Lemma 2 and Corollary 1 are satisfied, the shop can achieve rolling processing.

2.3 | Mathematical Model

This section discusses the problem of job allocation to achieve rolling production in a two-stage hybrid flow shop. To schedule the two-stage rolling production workshop, each batch of jobs from the first stage needs to be assigned to machine M_{1i} for processing, where $i \in \mathbb{N}_m^+$. In practical manufacturing scheduling plans, there are cases such as $b_{ij} \geq \Delta p$, $i \in \mathbb{N}_m^+$, $j \in \mathbb{N}_{m(i)}^+$, and in other situations where $b_{iu} < \Delta p$, $i \in \mathbb{N}_m^+$, $j \in \mathbb{N}_{m(i)}^+$, and $j \neq u$. According to Corollary 1, when $\sum_1^b b_{ij} \geq j\Delta p$ holds for $i \in \mathbb{N}_m^+$, $j \in \mathbb{N}_{m(i)}^+$, jobs can maintain rolling processing within the time interval $\left(0, T_{min}\right]$.

If the above conditions are not satisfied, additional jobs will be allocated to the first stage to ensure $\sum_1^j b_{ij} \geq j\Delta p$ holds, for any $i \in \mathbb{N}_m^+$, $j \in \mathbb{N}_{m(i)}^+$. Without loss of generality, let δ_i denote the number of additional jobs. Based on Lemmas 1 and 2, a linear programming model with 0-1 decision variables is established in this section. Here, x_{ij} and x_{iu} are 0-1 decision variables, representing where the *j-th* batch of jobs or the *u*-th batch of jobs, respectively, is processed at M_{1i}. Specifically, $x_{ij} = 1$ if the *j-th* batch of jobs is processed at M_{1i}, otherwise $x_{ij} = 0$. Similarly, $x_{iu} = 1$ if the *u-th* batch of jobs is processed at M_{1i}, otherwise $x_{iu} = 0$. The model is described as follows:

$$minC_{max} \tag{3}$$

$$\sum_{1}^{m}(x_{ij}+x_{iu})=1, i\in\mathbb{N}_{m}^{+}, j\neq u, j\in\mathbb{N}_{m(i)}^{+}, u\in\mathbb{N}_{m(i)}^{+} \tag{4}$$

$$\sum_{j=1}^{m(i)}[x_{ij}(b_{ij}\alpha+\sigma)]+\sum_{u=1}^{m(i)}[x_{iu}(b_{iu}\alpha+\sigma)]+\alpha\times max\{(\delta_i-r_{1i}),0\}\leq C_{max}, i\in\mathbb{N}_{m}^{+}, j\neq u \tag{5}$$

$$\sum_{j=1}^{m(i)}x_{ij}b_{ij}\geq\sum_{j=1}^{m(i)}x_{ij}\Delta p, i\in\mathbb{N}_{m}^{+} \tag{6}$$

$$\sum_{u=1}^{m(i)}x_{iu}b_{iu}\geq\sum_{j=1}^{m(i)}x_{iu}\Delta p, i\in\mathbb{N}_{m}^{+} \tag{7}$$

$$\sum_{1}^{m}r_{1i}=r, i\in\mathbb{N}_{m}^{+} \tag{8}$$

$$x_{ij}=0\ or 1, i\in\mathbb{N}_{m}^{+}, j\in\mathbb{N}_{m(i)}^{+}, \tag{9}$$

$$x_{iu}=0\ or 1, i\in\mathbb{N}_{m}^{+}, u\in\mathbb{N}_{m(i)}^{+}, \tag{10}$$

The objective function (3) aims to minimize the total completion time for all batches of jobs completed in the first stage; Constraint (4) indicates that each batch of jobs can only be assigned to a machine M_{i1},$i\in\mathbb{N}_{m}^{+}$ in the first stage for processing; Constraint (5) specifies that the total completion time of all batches is not less than C_{max}; Constraint (6) stipulates that within the time interval [0, $H_{im(i)}$], the average production rate of M_{i1} is greater than or equal to $1/m\psi$, ensuring the rolling production of a two-stage hybrid flow shop; Constraint (7) represents that when the product batch does not meet the conditions for continuous production, additional inventory δ_i can be added to ensure continuous production in the two-stage hybrid flow shop; Constraint (8) means that the amount of inventory allocated for all machines in the initial state is r; and Constraints (9) and (10) are the range of values for decision variables.

By solving the linear programming problem, the jobs allocated to machine M_{i1},$i\in\mathbb{N}_{m}^{+}$ can be determined. However, the processing sequence for the jobs assigned to each machine has not been specified in accordance with the objectives. Due to the adopted batch sorting schemes, the corresponding WIP inventory as the objective and further optimizes the scheduling scheme to minimize it. A linear relationship exists between the total completion time and the number of jobs processed at machine M_{i1},$i\in\mathbb{N}_{m}^{+}$. Then, we obtain the following lemma.

Lemma 3. In a two-stage hybrid flow shop scheduling problem, if all machines M_{i1},$i\in\mathbb{N}_{m}^{+}$ satisfy a linear relationship between time t and product quantity, the peak WIP inventory occurs when each batch is completed.

Proof: Without loss of generality, the stock is zero at the initial state. Let the number of jobs be a at M_{1i}, the processing time of an individual job is α, the time required to complete a single job in the second stage is $\psi=\mu/n$, and the machine set-up time is represented by σ. When time t satisfies $0\leq t\leq a\alpha$, the calculation for the WIP inventory is $t/\alpha-t/\psi$. Thus, the number of jobs in the process decreases linearly with time t. When time t satisfies $a\alpha\leq t\leq a\alpha+\sigma$, the calculation for the WIP inventory is $a-t/\psi$, and the number of jobs in the process decreases linearly with time t. Therefore, the peak WIP inventory occurs when each batch finishes processing.

Based on Lemma 3, it can be known that for analyzing the number of jobs in the two-stage rolling production workshop, we only need to focus on the nodes when each batch of jobs is processed. Letting the system time be t, $wip(i,t)$ denotes the WIP inventory at M_{1i}, $i\in\mathbb{N}_{m}^{+}$, at time t, calculated by the formulas as follows:

$$wip(i,t)=t/\alpha-t/\psi+r_{1i}+max\{(\delta_i-r_{1i}),0\}, 0<t\leq\vartheta_i\alpha \tag{11}$$

$$wip(i,t)=\vartheta_i-t/\psi+r_{1i}+max\{(\delta_i-r_{1i}),0\}, \vartheta_i\alpha<t\leq\vartheta_i\alpha+\sigma \tag{12}$$

$$wip(i,t)=t/\alpha-t/\psi+\vartheta_i+r_{1i}+max\{(\delta_i-r_{1i}),0\}, \vartheta_i\alpha+\sigma<t\leq T_{i1} \tag{13}$$

$$wip(i,t)=b_{i1}-t/\psi+\vartheta_i+r_{1i}+max\{(\delta_i-r_{1i}),0\}, T_{i1}<t\leq H_{i1} \tag{14}$$

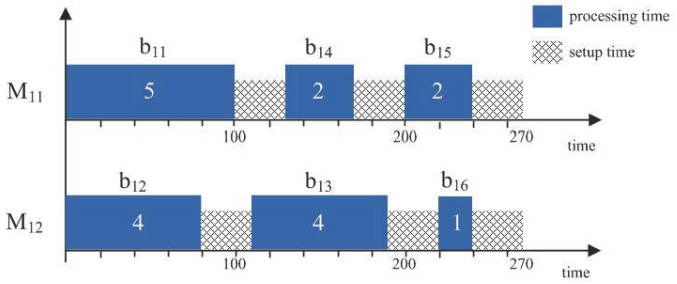

FIGURE 1 The Gantt chart for the parallel machines scheduling at Phase 1

$$wip(i,t) = \sum_{j=1}^{m(i)} b_{ij} - t/\psi + \vartheta_i + r_{1i} + max\{(\delta_i - r_{1i}), 0\}, H_{ij} < t \leq H_{i(j+1)} - \sigma \tag{15}$$

$$\begin{aligned} wip(i,t) = &\sum_{j=1}^{m(i)} b_{ij} + (t - H_{i(j+1)})/\alpha - t/\psi + \vartheta_i + \\ &r_{1i} + max\{(\delta_i - r_{1i}), 0\}, H_{i(j+1)} - \sigma < t \leq H_{i(j+1)} \end{aligned} \tag{16}$$

Equation (11) represents the calculation formula for WIP inventory of each machine in the initial state of the system within the time range $0 \leq t \leq \vartheta_i\alpha$; Equation (12) represents the calculation formula for WIP inventory of each machine before processing the first batch of jobs during $\vartheta_i\alpha \leq t \leq \vartheta_i\alpha + \sigma$; Equation (13) represents the calculation formula for WIP inventory of each machine before the completion of the first batch of jobs processing with the time range of $\vartheta_i\alpha + \sigma$ to T_{i1}; Equation (14) represents the calculation of each machine after the completion of the first batch of jobs processing during $T_{i1} < t < H_{i1}$; Equation (15) represents the calculation of each machine before the completion of the j-th batch of jobs processing with the range of H_{ij} to $H_{i(j+1)} - \sigma$; and Equation (16) indicates the calculation equation for the WIP inventory of each machine in the time range from $H_{i(j+1)} - \sigma$ to $H_{i(j+1)}$, from the completion of the j-th batch of jobs to the start of processing of the $(j+1)$-th batch of jobs.

To illustrate the operation of Equations (11) to (16), a simple example is provided. In this example, there are two machines in the first stage and one machine in the second stage, *i.e.*, $m = 2$, and $n = 1$. The processing time of a job by machine M_{1i}, $i \in \{1, 2\}$, and M_{11}, M_{12}, and M_{21} for Stages 1 and 2 is 20s, 20s, and 15s, respectively ($\alpha = 20s$ and $\mu = 15s$). The set-up time for machine M_{1i}, $i \in \{1, 2\}$, is 30s($\sigma = 30s$). Machine M_{21} has no set-up time. Therefore, we have $\alpha/m = 10s < \varphi = 15s$ and $\Delta p = \sigma/(m\psi - \alpha) = 3$. Assume there are six batches, *i.e.*, $B = \{b_{11}, b_{12}, b_{13}, b_{14}, b_{15}, b_{16}\}$. The sizes of these batches are 5, 4, 4, 2, 2, and 1, respectively. According to Lemma 2, we can find a solution to assign different batches processed at M_{11} and M_{12}, respectively. Batch $b_{11} = 5$, $b_{14} = 2$, and $b_{15} = 2$ are sequentially processed at M_{11}, $b_{12} = 4$, $b_{13} = 4$, and $b_{16} = 1$ are sequentially processed at M_{12}. The Gantt chart for the obtained schedule is shown in Fig.1 .

3 | PROPOSED ALGORITHM

3.1 | Greedy Algorithm

While the linear programming model developed in Section 2 can determine the total completion time of all jobs in the first stage, it does not specify the processing sequence of different batches on the same machine. Altering the processing sequence directly influences the WIP inventory. Therefore, to minimize overall production cost, this paper utilizes a greedy algorithm to optimize the processing sequence of jobs in the first stage. The choice of the greedy algorithm is based on its high efficiency and straightforward implementation for combinatorial optimization tasks. To ensure continuous job processing while reducing WIP inventory, the greedy algorithm directly selects the solution with the lowest local fitness.

The greedy algorithm is a simpler and faster design technique for addresing certain optimization problems. It progresses step by step, typically based on the current situation and optimal choices according to a specific optimization metric, without considering various possible overall scenarios. This saves considerable time that would otherwise be spent exhaustively all possibilities to find the optimal solution. The greedy algorithm adopts a top-down, iterative approach to make successive greedy

choices. Each time a greedy choice is made, the problem being solved is reduced to a smaller sub-problem. Through each step of greedy choice, an optimal solution to the problem can be obtained.

Next, we discuss how to use a greedy algorithm to generate a schedule that complies with the interruptible process constraints in a two-stage hybrid flow shop scheduling problem. As long as Lemmas 2 and 3 proposed in Section 2 are satisfied, we can address the interruptible process constraints in the scheduling problem. Since Equations 1 and 2 take into account the requirements of rolling production, satisfying Lemmas 2 and 3 will enable rolling production. In the greedy algorithm, local fitness refers to the WIP inventory levels of different schedules. The proposed greedy algorithm allows for the efficient determinaion of approximate optimal solutions, as shown in Algorithm 1. In Algorithm 1, the task sequence matrix $B = b_{ij}(m \times J)$ is obtained

Algorithm 1 A greedy algorithm using bubble sort for the WIP inventory optimization

1: **procedure** GREEDYALGORITHM
2: **Input**: Task sequence matrix $B_{m\times J}$;
3: **Output**:Task sequence matrix $B'_{m\times J}$ and the WIP inventory;
4: Bubble sort in descending order of matrix $B_{m\times J}$, for all batches $i \in \mathbb{N}_m^+$;
5: **for** $i = 1$ to m **do**
6: **for** $k = 1$ to J **do**
7: **for** $j = 1$ to J **do**
8: **if** k==1 **then**
9: %Lemma 2 holds;
10: **if** $b_{ij} >= \Delta p$ **then**
11: $b'_{ik} \leftarrow b_{ij}$;
12: break;
13: **end if**
14: **else**
15: %Lemma 3 holds;
16: **if** $\sum_1^{k-1} b_{i(k-1)} + b_{ij} \geq k\Delta p$ **then**
17: $b_{i(k-1)} \leftarrow b_{ij}$;
18: break;
19: **end if**
20: **end if**
21: **end for**
22: **end for**
23: **end for**
24: Compute the WIP inventory by Task sequence matrix $B'_{m\times J}$ and Equations 11 to 16;
25: **end procedure**

through linear programming as presented in Section 2, where m represents the number of machines in the first stage, and $J = max\{m(i)|i \in \mathbb{N}_m^+\}$ indicates the number of batches allocated to the first stage. In Algorithm 1, Statement 1) conducts a bubble sort on the $B_{m\times J}$ rows in descending order, where Δp denotes the minimum batch size that satisfies the continuous process constraint between two stages. Statements 2) - 9) select the minimum batch size on each machine meeting continuity and schedule them first. Statements 10) - 17) sequentially choose subsequent batches, with 11) providing the minimum batch size condition from Lemma 1. Algorithm 1 rearranges batches via bubble sort, enabling the quick selection of the minimum batch in each step for local optima. With the sequence obtained from Algorithm 1, the WIP inventory is calculated. Therefore, we first minimize the completion time globally through linear programming. Then, we locally optimize the WIP inventory using Algorithm 1, thus optimizes the overall production schedule in a two-stage hybrid flow shop scheduling problem.

3.2 | The Designed Q-learning Algorithm

This paper utilizes Algorithm 1 and the WIP inventory formulas to efficiently obtain optimized scheduling solutions. However, the greedy algorithm does not consider the perspective of overall optimality; thus the solutions obtained cannot be guaranteed to be globally optimal. To validate the effectiveness of Algorithm 1 and further enhance the scheduling plan, Q-learning is applied to the two-stage hybrid flow shop rolling production scheduling problem in this paper.

Q-learning is a renowned reinforcement learning algorithms [33]. During the interaction with the environment, the agent receives rewards R corresponding to executed action a. The algorithm combines states s and actions a to construct a Q-table, recording updated Q values. Subsequently, it selects actions that maximize rewards according to the Q-table. After each action, the Q-table is simultaneously updated according to the following formula.

$$Q(s_{ite}, a_{ite}) = Q(s_{ite}, a_{ite}) + \phi[R_{ite} + \gamma max Q(s'_{ite}, a'_{ite}) - Q(s_{ite}, a_{ite})] \tag{17}$$

where s_{ite} denotes the state at iterations ite, a_{ite} represents the action taken at iterations ite, and $Q(s_{ite},a_{ite})$ indicates the updated Q value when taking action a_{ite} in state s_{ite}. R_{ite} denotes the reward at iteration ite, and $\phi \in [0, 1]$ is the learning rate. γ is the discount factor. If γ is close to zero, it focuses more on short-term rewards. If γ is close to 1, it cares more about long-term cumulative rewards. Notably, Q-learning updates do not necessarily use the data sampled from $maxQ(s_{ite},a_{ite})$. For exploration, the ε-greedy algorithm is generally used to select the action with the highest expected reward estimation. Specifically, with probability $\varepsilon = 1$, the action with the highest empirical expected reward is chosen, and with probability ε a random action is performed.

Consequently, with an increasing number of explorations, our reward estimates for different actions improve, obviating the necessity for extensive search. To this end, this paper refines the ε-greedy policy for action selection, as illustrated below.

$$\varepsilon = \frac{0.5}{1 + e^{\frac{ite - \theta \times ite_{max}}{ite_{max}}}} \tag{18}$$

$$a_{ite} = \begin{cases} random\ A, & rand > 1 - \varepsilon \\ a^*, & otherwise \end{cases} \tag{19}$$

$$a^* = arg\ max a_j \{Q(q_{ij}) | i \in \mathbb{N}^+_m, j \in \mathbb{N}^+_{m(i)}\} \tag{20}$$

where Equation 18 is used to calculate the value ε, the modified ε-greedy policy exhibits better learning effects than a fixed ε value. The value ite_{max} represents the termination condition of the algorithm, which is the maximum iterations and θ denoting a random number in the range of $\theta \in [0, 1]$. Formula 19 specifies that if a random number rand $\in [0, 1]$, $rand > 1 - \varepsilon$, then action a is randomly selected from the action set A; otherwise action a^* is chosen. Here, a^* represents the action a corresponds to the maximum value q_{ij}, calculated by Formula 20. Subsequently, Q-learning is carried out following the steps outlined in Procedure 1.

Procedure 1: The designed Q-learning algorithm is outlined as follows.

1. Initialize the parameters of the Q-learning algorithm, input process information of the two-stage hybrid flow shop, and obtain the initial schedule B_{ite} from the linear programming model, setting $ite = 1$.
2. Update the ε value based on Equation 18.
3. Choose an action a_{ite} in the current world state s based on Formulas 19 and 20.
   (a) If the random value $rand \in [0, 1]$, $rand > 1 - \varepsilon$, then generate action a_{ite} randomly from action set A.
   (b) If the random value $rand \in [0, 1]$, $rand \geq 1 - \varepsilon$, then action a_{ite} is a^* based on the maximum q value.
4. Execute action a and observe the resulting state s'_{ite}.
5. Choose a row $B_{:,i}$ in $B_{ite} = b_{ij}(m \times j)$, where $i = 1$.
6. Update the reward $R_{ite} = d_{kj}(m(i) \times m(i))$ based on Corallary 1.
   (a) If $b_{ij} = b_{i(j+1)} = \Delta p$, $d_{kj} = +M$;% $+ M$ is a positive number.
   (b) If $b_{ij} \geq \Delta p$, $b_{ij} + b_{i(j+1)} > 2 * \Delta p$, $d_{kj} = min\{2 * \Delta p/(b_{ij} + b_{i(j+1)}), \Delta p\}$;

TABLE 1 Main parameters of algorithms

Algorithm	parameter	value
PSO[34]	c_1	2
	c_2	2
	1	Linearly decreases from 0.6 to 0.3
VNS[35]	Inloop_max	100

(c) Else $d_{kj} = -M$;% $-$ M is a negative number.

7. $i = i + 1$, if $i = m$ goes to Step 8), otherwise go to Step 5).

8. Update the $Q-$table based on Equation 17.

9. Update the schedule B'_{ite} based on the $Q-$table.

10. Check whether schedule B'_{ite} meets the interruptible process constraints, *i.e.*, Lemma 2 and Corollary 1.

    (a) If Lemma 2 or Corollary 1 holds, update $B_{ite} = B'_{ite}$.

        i. Compute the WIP inventory based on Equations 11 to 16;

        ii. Modify the reward R_{ite}, $d_{kj} = d_{kj} + |d_{kj}|$;

    (b) If Lemma 2 or Corollary 1 do not hold, do not update B_{ite}.

        i. Modify the reward R_{ite}, $d_{kj} = min\{d_{kj}, 0\}$.

11. $ite = ite + 1$, if $ite = ite_{max}$, indicating that the termination condition is met, the designed $Q - learning$ will output a solution with the highest reward; otherwise, go to Step 2).

4 | COMPUTATIONAL EXPERIMENTS

4.1 | Parameter Selection and Design

To verify the effectiveness of the linear programming model and algorithm in solving the problem, the linear programming model and algorithm are coded in MATLAB. The linear programming model is solved by calling the ILOG CPLEX 12.6 software. The greedy algorithm (GA), Q-learning, particle swarm optimization (PSO)[35], and variable neighborhood search (VNS) are all coded in MATLAB using the same computer. The experiments are run on a personal computer with Windows 10, Intel CoreTM i5 3.3 GHz processor, and 8GB RAM, and the experimental computational time refers to the CPU running time of the computer. In this way, the linear programming model obtains the optimal completion time. Then, various algorithms are used to optimize the machining sequence of jobs to reduce WIP inventory. The parameters in PSO and VNS are shown in Table 1 .

Q-learning involves three parameters: the number of iterations ite, the learning factor ϕ, and the discount factor γ. Through thousands of experiments, we narrow down the selection of ite, ϕ, and γ parameters within specific ranges, setting them to three levels respectively, where $ite = \{500, 700, 900\}$, $\phi = \{0.35, 0.37, 0.39\}$, and $\gamma = \{0.30, 0.35, 0.40\}$. The Taguchi parameter design method can pick out parameter combinations that stabilize experimental results with low variability and utilize orthogonal tables to arrange experimental parameters for experiments.

Through the Taguchi parameter design method, the orthogonal table for arranging parameters of the designed Q-learning is shown in Table 2 . As shown in Table 3 , each parameter set runs independently five times to obtain a non-dominated solution set $E_i = \{i|i = 1, 2, 3,\ldots,9\}$. E_1 to E_9 form a final set FE. $E_i = \{wip_1, wip_2, wip_3, wip_4, wip_5\}$, where wip represents the WIP inventory with the parameter set E_i. The contribution ratio $CR(i)$ of E_i is calculated as follows:

$$CR(i) = \sum_{i=1}^{|E'_i|} e_i / \sum_{i=1}^{|FE'|} wip_i \tag{21}$$

TABLE 2 Computational test results

dataset	minimum completion times(s)	upper bound(s)	low bound(s)
ta001	1266	1278	1232
ta002	1298	1359	1290
ta007	1229	1239	1226
ta008	1182	1170	1206
ta010	1098	1082	1108
ta011	1522	1582	1448
ta012	1505	1659	1479
ta014	1337	1377	1308
ta018	1396	1538	1363
ta019	1566	1593	1472

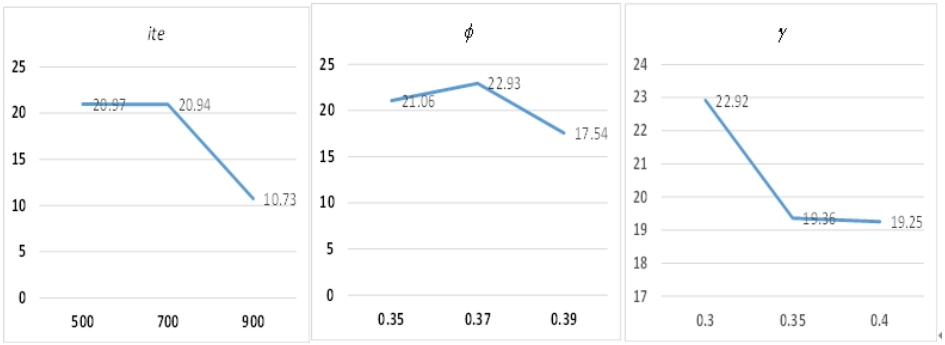

FIGURE 2 Influence chart of control parameters

TABLE 3 Orthogonal experimental design for parameters

No.	ite	ϕ	γ	$CR(\%)$
1	500	0.35	0.3	26.39
2	500	0.37	0.4	26.25
3	500	0.39	0.35	20.97
4	700	0.35	0.4	20.77
5	700	0.37	0.35	21.03
6	700	0.39	0.3	20.93
7	900	0.35	0.35	16.09
8	900	0.37	0.3	21.50
9	900	0.39	0.4	10.73

where $E_i' = E_i \cap FE$, $FE' = \cup_{i=1}^{|9|} E_i'$, it can be easily understood that FE' represent the set after removing deplicate elements from FE. According to Table 2 and Figure 2 , the selected algorithm parameters with the largest average contribution value (ACV) are $ite = 500$, $\phi = 0.37$, $\gamma = 0.3$.

Due to the limited range of ϕ and γ values in the algorithm, it is easy to see that at the 0.05 significance level, these two parameters do not significantly differ in the amount of WIP inventory. The experimental data consists of 30 batches of jobs randomly generated, with each parameter set running independently 30 times, totaling $30 * 9 = 270$ experiments. The number of

TABLE 4 ANOVA table

Source	SS	df	MS	F	P-value	F crit
Factor	5299.919	8	662.4898	6.448529	$1.17E-07$	1.973975
Error	26813.84	261	102.735			
Total	32113.76	269				

iterations *ite* significantly influences the WIP inventory amount, as indicated by the significant difference for *ite* in the analysis of variance results shown in Table 3 , with a P-value less than 0.05. Furthermore, this means a significant difference for *ite*.

4.2 | Efficiency and Effectiveness of the Proposed Algorithm

To validate the proposed algorithm design, this study selected 5 test cases of 20 jobs on five machines and another five test cases of 20 jobs on 10 machines from the Taillard benchmark dataset for flow shop scheduling problems. The proposed algorithm is utilized to calculate the minimum completion time for each job, and the output results are examined to be within the lower and upper bounds of the feasible region, demonstrating the effectiveness and feasibility of the proposed algorithm. The computational results given 1000 generations take only around $1s$, as shown in Table 4 . The benchmark dataset is available at https://cuglirui.github.io/Dataset/Taillard-PFSP.rar for download.

In the experimental analysis, a GA, VNS, PSO, and the designed Q-learning algorithm are adopted. Among them, VNS and PSO serve as benchmark algorithms for comparison. Each test case is run independently 30 times, and the experimental results, including the average gap and minimum gap WIP inventory for different algorithms, are presented in Table 5 and Table 6 , respectively. The gap value calculation formula is shown in Eq.(22).

$$Gap(\%) = (D - Ave_{GA})/Ave_{GA} \times 100\% \tag{22}$$

where D denotes the average WIP inventory when operating VNS, PSO, and Q-learning algorithms, respectively. Ave_{GA} represents the average WIP inventory resulting from the operation of a greedy algorithm.

By Eq.(22), we can conclude that, as the data in Table 5 shows, when the experimental result is greater than 0, the closer the number is to 0, indicating that GA has a better operation effect, and when the experimental result is less than 0, it means that the other algorithm has a better operation effect. Additionally, as exhibited in Table 5 , among 13 test cases, the number of times that GA, Q-learning, and VNS achieved the minimum average WIP inventory are 9, 2, and 2, respectively. In particular, the number of times that GA and Q-learning attained the second minimum values are 4 and 9, respectively. Likewise, as the data in Table 6 exhibit, among the 13 test cases, the number of times that GA, Q-learning, and VNS attained the minimum WIP inventory are 5, 5, and 3, respectively. Concurrently, the number of times that these three algorithms obtained the second minimum values are 6, 6, and 1, respectively. The results demonstrate that this paper proposes GA and Q-learning algorithms in this paper can both acquire optimized and near-optimized solutions.

5 | CONCLUSION

In response to the practical demand for optimizing production scheduling in small-batch manufacturing at glass plants, this paper proposes a two-stage hybrid flow shop scheduling model with sequence-dependent set-up time and rolling processing constraints to minimize the total completion time in the first stage and work-in-process (WIP) inventory for multi-variety small-batch processing. A linear programming model is formulated to optimize the scheduling plan. To further improve the scheduling results, a greedy algorithm and a Q-learning algorithm are designed to obtain better solutions by determining the optimized product sequence. The proposed methods demonstrate effectiveness in reducing WIP inventory through numerical experiments. The scheduling model and algorithms provide a viable framework for multi-variety mixed-model scheduling in practical manufacturing systems. In the future, more complex constraints and objectives can be incorporated, or metaheuristic optimization methods can be applied to solve large-scale problems.

TABLE 5 Difference of experimental average gap WIP between different algorithms

No.	VNS and GA(%)	PSO and GA(%)	Q-learning and GA(%)
1	17.88	15.78	0.08
2	18.33	18.69	8.49
3	27.36	21.56	12.81
4	1.91	5.78	−3.96
5	−0.13	3.09	0.99
6	7.41	8.04	6.42
7	1.70	4.24	0.95
8	2.53	20.91	1.86
9	5.28	14.46	0.52
10	−0.62	5.65	2.30
11	0.43	12.97	−7.31
12	2.25	6.67	0.43
13	6.41	8.36	0.13

TABLE 6 Difference of experimental minimum WIP between different algorithms

No.	VNS and GA(%)	PSO and GA(%)	Q-learning and GA(%)
1	17.78	10.32	−3.16
2	16.48	14.70	7.25
3	24.63	16.14	9.90
4	−0.68	3.48	−5.25
5	−1.30	1.57	0.09
6	5.32	7.80	4.32
7	1.47	2.84	−0.41
8	−5.16	20.91	0.69
9	3.48	7.63	0.16
10	−1.68	0.96	1472
11	0.42	4.76	−8.40
12	2.25	2.64	0.01
13	1.27	2.25	−0.24

In the future, more complex constraints and objectives can be incorporated, or metaheuristic optimization methods can be applied to solve large-scale problems.

6 | ACKNOWLEDGEMENTS

6.1 | Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

7 | BIBLIOGRAPHY

References

1. Vaka M, Walvekar R, Rasheed A, Khalid M. A review on Malaysia's solar energy pathway towards carbon-neutral Malaysia beyond Covid'19 pandemic. *Journal Cleaner Production* 2020; 273: 122834.

2. Ribas I, Leisten R, Framiñan J. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research* 2010; 37(8): 1439–1454.

3. Qiao Y, Wu N, He Y, Li Z, Chen T. Adaptive genetic algorithm for two-stage hybrid flow-shop scheduling with sequence-independent setup time and no-interruption requirement. *Expert Systems with Applications* 2022; 208: 11608.

4. Engin O, Guclu A. A New Hybrid Ant Colony Optimization Algorithm for Solving the No-Wait Flow Shop Scheduling Problems. *Applied Soft Computing* 2018; 72: 166–176.

5. Rossit DA, Tohmé F, Frutos M. The non-permutation flow-shop scheduling problem: a literature review. *Omega* 2018; 77: 143–153.

6. Gong H, Tang L, Duin C. A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times. *Computers & Operations Research* 2010; 37(5): 960–969.

7. Nazari S, Keshavarz MM P, Sareh P. A multi-objective optimization approach to designing window and shading systems considering building energy consumption and occupant comfort. *Engineering Reports* 2023; 5(10): e12726.

8. Wang H, Jiang Z, Wang Y, Zhang H, Wang Y. A two-stage optimization method for energy-saving flexible job-shop scheduling based on energy dynamic characterization. *Journal of cleaner production* 2018; 188: 575–588.

9. Wang S, Wang X, Chu F, Yu J. An energy-efficient two-stage hybrid flow shop scheduling problem in a glass production. *International Journal of Production Research* 2020; 58(8): 2283–2314.

10. Li Y, Dai Z. A two-stage flow-shop scheduling problem with incompatible job families and limited waiting time. *Engineering Optimization* 2019.

11. Fan B, Yang W, Zhang Z. Solving the two-stage hybrid flow shop scheduling problem based on mutant firefly algorithm. *Journal of Ambient Intelligence and Humanized Computing* 2019; 10: 979–990.

12. Neufeld J, Schulz S, Buscher U. A systematic review of multi-objective hybrid flow shop scheduling. *European Journal of Operational Research* 2023; 309(1): 1–23.

13. Li J, Sang H, Han Y, Wang C, Gao K. Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions. *Journal of Cleaner Production* 2018; 181: 584–598.

14. Fu Y, Li H, Huang M, Xiao H. Bi-objective modeling and optimization for stochastic two-stage open shop scheduling problems in the sharing economy. *IEEE Transactions on Engineering Management* 2021.

15. Kahraman C, Engin O, Kaya I, Öztürk R. Multiprocessor task scheduling in multistage hybrid flow-shops: a parallel greedy algorithm approach. *Applied Soft Computing* 2010; 10(4): 1293–1300.

16. Moradinasab N, Shafaei R, Rabiee M, Ramezani P. No-wait two stage hybrid flow shop scheduling with genetic and adaptive imperialist competitive algorithms. *Journal of Experimental & Theoretical Artificial Intelligence* 2013; 25(2): 207–225.

17. Azaiez M, Gharbi A, Kacem I, Makhlouf Y, Masmoudi M. Two-stage no-wait hybrid flow shop scheduling with inter-stage flexibility: A mathematical model. In: . 1 of *2021 IEEE International Conference on Networking, Sensing and Control (ICNSC)*. ; 2021: 1–5.

18. Costa A, Fernandez-Viagas V, Framiñan J. Solving the hybrid flow shop scheduling problem with limited human resource constraint. *Computers & Industrial Engineering* 2020; 146: 106545.

19. Aqil S, Allali K. Local search metaheuristic for solving hybrid flow shop problem in slabs and beams manufacturing. *Expert Systems with Applications* 2020; 162: 113716.

20. Gupta J, Tunc E. Scheduling a two-stage hybrid flowshop with separable setup and removal times. *European Journal of Operational Research* 1994; 77(3): 415–428.

21. Sheikh S, Komaki G, Kayvanfar V, Teymourian E. Multi-Stage assembly flow shop with setup time and release time. *Operations Research Perspectives* 2019; 6: 100111.

22. Wang S, Liu M, Chu C. A branch-and-bound algorithm for two-stage no-wait hybrid flow-shop scheduling. *International Journal of Production Research* 2015; 53(4): 1143–1167.

23. Wang S, Liu M. A genetic algorithm for two-stage no-wait hybrid flow shop scheduling problem. *Computers & Operations Research* 2013; 40(4): 1064–1075.

24. Dong J, Pan H, Ye C, Tong W, Hu J. No-wait two-stage flowshop problem with multi-task flexibility of the first machine. *Information Sciences* 2021; 544: 25–38.

25. Cai J, Zhou R, Lei D. Fuzzy distributed two-stage hybrid flow shop scheduling problem with setup time: collaborative variable search. *Journal of Intelligent & Fuzzy Systems* 2020; 38(3): 3189–3199.

26. Wang S, Kurz M, Mason S, Rashidi E. Two-stage hybrid flow shop batching and lot streaming with variable sublots and sequence-dependent setups. *International Journal of Production Research* 2019; 57(22): 6893–6907.

27. Zhao F, Jiang T, Wang L. A reinforcement learning driven cooperative meta-heuristic algorithm for energy-efficient distributed no-wait flow-shop scheduling with sequence-dependent setup time. *IEEE Transactions on Industrial Informatics* 2022.

28. Shahvari O, Logendran R. A comparison of two stage-based hybrid algorithms for a batch scheduling problem in hybrid flow shop with learning effect. *International Journal of Production Economics* 2018; 195: 227–248.

29. Okpoti E, Jeong IJ. Decentralized Production Planning Using Collaborative Agents for a Bi-Objective Reentrant Hybrid Flow Shop. In: ; 2019: 198–202.

30. Park K, Li J, Feng S. Scheduling policies in flexible Bernoulli lines with dedicated finite buffers. *Journal of manufacturing systems* 2018; 48: 33–48.

31. Zhu Q, Yuan J, Wang G, Hou Y. Scheduling a Single-Arm Two-Cluster Tool With a Process Module Failure Subject to Wafer Residency Time Constraints. *IEEE Transactions on Automation Science and Engineering* 2023.

32. Li J, Qiao Y, Zhang S, Li Z, Wu N, Song T. Scheduling of single-arm cluster tools with residency time constraints and chamber cleaning operations. *Applied Sciences* 2021; 11(19): 9193.

33. Tan J, Guan W. Resource allocation of fog radio access network based on deep reinforcement learning. *Engineering Reports* 2022; 4(5): e12497.

34. Han Y, Yuan H, Shao Y, Li J, Huang X. Capacity Consistency Prediction and Process Parameter Optimization of Lithium-Ion Battery based on Neural Network and Particle Swarm Optimization Algorithm. *Advanced Theory and Simulations* 2023; 6(8): 2300125.

35. Clerc M, Kennedy J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE transactions on Evolutionary Computation* 2002; 6(1): 58–73.

How to cite this article: Williams K., B. Hoskins, R. Lee, G. Masato, and T. Woollings (2022), A regime analysis of Atlantic winter jet variability applied to evaluate HadGEM3-GC2, *Q.J.R. Meteorol. Soc.*, *2017;00:1–6*.