JOURNAL OF INDUSTRIAL AND PRODUCTION ENGINEERING
https://doi.org/10.1080/21681015.2024.2337029

ARTICLE

Dynamic task allocation of hybrid flow shop for machines in parallel with different speeds based on an MLD prediction model

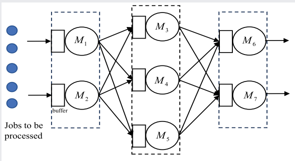
Jianbin Xin[a,b], Sixuan Li[a], Yanjie Zhou[c] and Andrea D'Ariano[d]

[a]Department of Logistics Management, School of Electrical and Information Engineering, Zhengzhou University, Zhengzhou, PR China; [b]State Key Laboratory of Intelligent Agricultural Power Equipment, Luoyang, China; [c]School of Management, Zhengzhou University, Zhengzhou, PR China; [d]Dipartimento di Ingegneria, Università Degli Studi Roma Tre, Italy

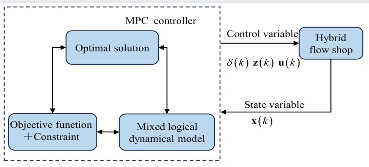
ABSTRACT
The current hybrid flow shop task allocation usually assumes a static manufacturing environment, which cannot effectively handle uncertain events in the production process. To address this, the dynamic task allocation problem for machines in parallel with different speeds is studied, and a Mixed Logical Dynamical (MLD) model for predicting the state of the production system is established. A dynamic allocation method based on Model Predictive Control (MPC) is proposed. The proposed novel method integrates dynamic model with rolling optimization to better deal with uncertain events in production process by decomposing the overall planning problem into smaller local planning models. Numerical results show that the proposed method outperform the traditional global planning method and rule-based allocation method in terms of time and job processing rate. In addition, through the Plant Simulation software, the simulation results are consistent with the numerical results, which fully proves the effectiveness of the proposed method.

ARTICLE HISTORY
Received 27 November 2023
Revised 4 March 2024
Accepted 25 March 2024

KEYWORDS
Hybrid flow shop; task allocation; mixed logical dynamical model; model predictive control



Schematic diagram of a three-stage hybrid flow shop with parallel machines of different speeds.


MPC controller
Optimal solution
Objective function +Constraint
Mixed logical dynamical model
Control variable
$\delta(k)$ $\mathbf{z}(k)$ $\mathbf{u}(k)$
Hybrid flow shop
State variable
$\mathbf{x}(k)$

Framework of the proposed dynamic allocation method.

Method

A mixed logical dynamical model has been developed to predict the state of the production system in the hybrid flow shop. An innovative dynamic allocation method, utilizing model predictive control, has been introduced. The proposed method integrates a dynamic model with rolling optimization to more effectively address uncertain events in the production process by breaking down the overall planning problem into smaller local planning models.

1. Introduction

Hybrid Flow Shop Scheduling Problem (HFSP) is a classic production scheduling problem that involves multiple stages with one or more parallel machines at each stage [1,2], where the goal is to schedule jobs to optimize one or more performance criteria. HFSP, as a generalized problem in modern production environments such as semiconductor manufacturing processes [3] and radio frequency identification tag processes [4], has been extensively studied over the past few decades. The HFSP consists of two sub-problems: job sequencing and job allocation.

As an essential branch of the hybrid flow shop scheduling problem, the job allocation problem has received much attention from many scholars. The task allocation problem in a hybrid flow shop was initially proposed as an optimization problem for two-stage and three-stage production schedules [5], and the allocation process was offline [6]. Subsequently, scholars extended the task allocation problem to multiple stages [7], with the allocation process being online.

Currently, most methods for solving task allocation problems assume a static manufacturing environment, where it is not possible to make timely adjustments based on the progress of tasks in actual task allocation. In order to deal with uncertain events in the production process, such as machine failures or new orders arrival, it is necessary to establish a dynamic allocation mathematical model that can allocate jobs in a flexible manner and enhance production efficiency.

The dynamic task allocation problem in a hybrid flow shop has several challenges for providing efficient models and algorithms. The objective of this problem is to find an optimal allocation scheme that allows tasks to be processed as quickly as possible. The first challenge is the inability to anticipate uncertain events in the production process, such as machine failures or the arrival of new jobs. Another challenge is that employing a completely rescheduling strategy results in significantly increased computation time.

CONTACT Yanjie Zhou ieyjzhou@zzu.edu.cn School of Management, Zhengzhou University, No.100 Science Avenue, Zhengzhou, Henan Province 450001, PR China

The contributions of this paper are given as follows:

- A mixed logical dynamical model for predicting the state of the production system is established. This model considers the allocation and processing of jobs as a dynamic system. Specifically, the MLD model describes the dynamic allocation process for processing jobs in a hybrid flow shop by incorporating linear dynamic inequalities with discrete variables. By using advanced control methods like model predictive control, the MLD model further improves the efficiency of job allocation.
- Additionally, model predictive control is introduced to handle uncertainties such as machine failures and the arrival of new jobs. The proposed method integrates a dynamic model with rolling optimization to better deal with uncertain events in the production process by decomposing the overall planning problem into smaller local planning models.

The remainder of the paper is organized as follows: a literature review is presented in Section 2. Section 3 introduces the hybrid flow shop scheduling problem with parallel machines of different speeds and presents the mixed logical dynamic model. Section 4 explains the principles of model predictive control. Section 5 validates the proposed method, analyzes the results in different scenarios, and conducts simulation verification using the Plant Simulation software. Finally, Section 6 provides a summary of the paper and outlines future research plans.

2. Literature review

This section reviews the classification based on machine characteristics and the approaches for static and dynamic task allocation.

2.1. Problem classifications

Based on the characteristics of the parallel machines, HFSP can be divided into three categories [8]: identical machines in parallel (Pm), where machines in the same stage possess the same processing capabilities and speeds [9,10]; machines in parallel with different speeds (Qm), where machines in the same stage possess the same processing capabilities but have different speeds [11,12]; unrelated parallel machines (Rm), where machines in the same stage have different processing capabilities and speeds [13,14].

In practical production processes, various uncertainties may arise, making it crucial to allocate jobs to machines effectively to enhance processing efficiency. Moreover, in actual production processes, machines with the same processing function within the same stage may have different processing rates due to different models or varying degrees of wear. However, most of the literature on the HFSP considers parallel identical machines, while there is limited research on HFSP with parallel machines of different speeds. Therefore, studying the task allocation problem in the hybrid flow shop with parallel machines of different speeds is necessary [12].

2.2. Static approaches

Most methods for solving the task allocation problem assume a static manufacturing environment, and commonly used methods include exact algorithms, heuristic algorithms, and metaheuristic algorithms [15]. In terms of exact algorithms, the branch and bound is the preferred method for solving the optimality of such problems. Moursli et al. [16] proposed a branch and bound algorithm to minimize the makespan for the hybrid flow shop scheduling problem .

In recent years, heuristic and metaheuristic algorithms have been commonly used to solve this type of problem. For example, some studies have utilized multi-objective evolutionary algorithms [17], an improved Nawaz, Enscore, and Ham heuristic algorithm [18], and heuristic methods with local search strategies to solve the hybrid flow shop scheduling problem [19]. Zheng et al. [20] combine the distribution estimation algorithm with iterative greedy search and propose a coevolutionary algorithm with a specific problem strategy to solve such problems. Some researchers have studied the no-wait flow shop problem without considering the buffer constraints [21]. Additionally, some studies have considered the presence of buffers [22–24], taking buffers into account can enhance production flexibility and optimize task allocation in hybrid flow shop scheduling.

With the development of artificial intelligence, learning algorithms based on agent systems and machine learning have been applied to HFSP. Babayan et al. [25] utilized an agent-based approach to solve the three-stage HFSP problem. Han et al. [26] combined reinforcement learning methods, abstracted HFSP into a Markov decision process, designed special states, actions, and reward functions for the decision process, and successfully solved the production scheduling problem in a metal processing workshop of an automobile engine factory.

2.3. Dynamic approaches

The production plan is influenced by real-time disturbances, which can cause deviations from the expected results of static planning. Therefore, online allocation is necessary to ensure production efficiency. Currently, research on dynamic HFSP can be divided into two categories: dynamic HFSP considering random or probabilistic machine failures, and dynamic HFSP considering random arrival of new jobs.

For the occurrence of new orders during the production process, Luo [27] proposed an efficient energy-saving dynamic flexible flow shop scheduling model that considers peak power. They also introduced a hybrid parallel genetic algorithm based on graphics processing units and employed a complete rescheduling strategy to solve the problem. This strategy performs well in maintaining optimal solutions but is rarely implemented in practice due to long computation times. Another approach proposed by Tliba et al. [28] is a digital dual-driven dynamic scheduling method that combines optimization and simulation to achieve dynamic allocation in a hybrid flow shop environment.

For dynamic HFSP concerning machine failures, Guo et al. [29] presented a variable priority dynamic scheduling optimization approach based on genetic algorithms, which dynamically generates prescheduling and rescheduling plans for sustainable HFSP. Currently, most online methods for task allocation rely on rescheduling techniques. However, employing complete rescheduling strategies significantly increases computation time and reduces allocation efficiency in the face of dynamic events.

2.4. Overall assessment of literature

As a summarizing remark, the dynamic task allocation problems in HFSP have not been sufficiently investigated in the literature, and a dynamical mathematical model for better predicting the state of the production system is needed. Additionally, an advanced control algorithm is needed to handle uncertainties such as machine failures and the arrival of new jobs.

3. Mixed logic dynamic model

This section introduces the research on task allocation in hybrid flow shop scheduling problems with parallel machines of different speeds. It formalizes the problem as a Mixed Integer Linear Programming (MILP) model and provides the mathematical expressions for the allocation problem based on the mixed logical dynamic model.

3.1. Problem description

The dynamic task allocation problem in a hybrid flow shop with parallel machines of different speeds is studied in this paper, as shown in Figure 1, where $M_1, M_2, \ldots, M_7$ represent the machine numbers. It is assumed that there are a total of N jobs to be processed through H stages. Each stage consists of one or more machines with the same functionality but potentially different processing speeds. At least one stage

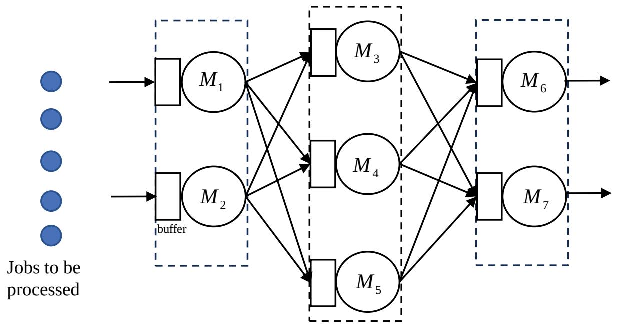


Figure 1. Schematic diagram of the three stage hybrid flow shop.

has more than one parallel machine. The objective is to provide the optimal allocation scheme for these jobs, considering a given optimization goal. The jobs are processed in a fixed sequence, and they may need to wait in buffers before the machines for processing. All jobs leave the system after completing processing in the final stage.

This paper makes the following important assumptions:

- All jobs are the same.
- The buffer capacity in front of each machine is large enough.
- The transportation time of the job between the two stages is far less than the processing time and is ignored.
- All jobs leave the system after being processed in the final stage.

3.2. Model construction

In this subsection, we construct a mixed logic dynamic model to describe the dynamic task allocation of the hybrid flow shop. The relevant model variables and symbols are shown in Table 1.

3.2.1. Machine $j \in S_h (h = 1)$

To achieve dynamic task allocation, it is necessary to describe the changes in the number of jobs in the buffer of machine j from the time instant k to $k + 1$. Taking into account the stage in which the machine is located, we first describe the case of machine $j \in S_1$, as follows: $k, j \in S_1$,

$$x_j(k+1) = (x_j(k) - v_j(k) \cdot \Delta t) \cdot b_j(k), \tag{1}$$

$$b_j(k) = \begin{cases} 1 & x_j(k) - v_j(k) \cdot \Delta t \geq 0 \\ 0 & x_j(k) - v_j(k) \cdot \Delta t < 0. \end{cases} \tag{2}$$

Equation (1) gives the change in the number of jobs in the buffer of machine j. Since the number of remaining jobs in the buffer is not equal to the processing rate multiplied by the unit time Δt, when the number of remaining jobs to be processed is less than the

Table 1. Representation of related variables and symbols.

State Variable	Description
$x_i(k)$	Number of jobs in the buffer of machine i at time instant k;
Decision variable	**Description**
$C_{ij}(k)$	Number of jobs transported from upstream machine i to downstream machine j from time instant k to $k+1$;
$D_j(k)$	Number of processed jobs by machine j from time instant k to $k+1$;
$b_j(k)$	If the number of jobs in the buffer of machine j at time instant k is greater than the processing capacity of the machine, $b_j(k)=1$; otherwise, it is 0;
Symbol	**Description**
M	A big positive number;
ε	A very small positive number;
N	Total number of jobs to be processed at the initial moment;
Δt	Time interval;
h	Stage index;
i,j	Machine index;
S_h	The set of machines in stage h;
S	Set of all machines;
K	Predictive horizon.;
k	Time instant $k\in\{0,1,\ldots,K-1\}$;
$v_j(k)$	Processing rate of machine j from time instant k to $k+1$.

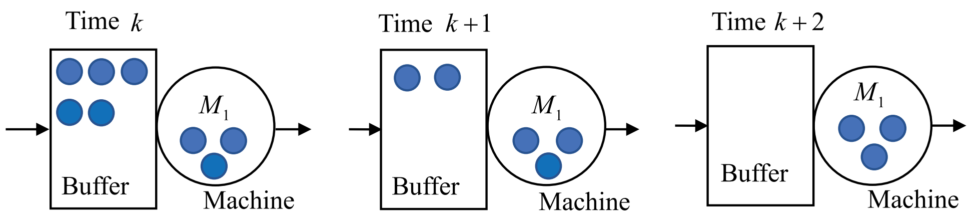


Figure 2. Diagram of the relationship between the buffer and the machine processing capacity (The number of jobs in the machine indicates the processing capability in a time unit).

maximum processing capacity of the machine, they will be subtracted negatively, but this is not in line with the actual situation, to ensure that the number of jobs in the buffer before machine j becomes zero when all the jobs in that buffer have been processed, we introduce variable $b_j(k)$ and equation (2).

Figure 2 shows a diagram of the job from buffer to machine processing. When the number of jobs in the buffer is sufficient, the number of jobs reduced in the buffer is equal to the processing capacity of the machine; When the number of jobs in the buffer is less than the processing capacity of the machine, the number of jobs processed by the machine is equal to the number of jobs remaining in the buffer.

According to the transformation of constraints [30], equation (2) can be equivalently expressed as the following two inequalities:

$$x_j(k)-v_j(k)\cdot\Delta t\le b_j(k)\cdot M-\varepsilon \tag{3}$$

$$x_j(k)-v_j(k)\cdot\Delta t\ge M\cdot(b_j(k)-1) \tag{4}$$

Considering that the term $x_j(k)\cdot b_j(k)$ is nonlinear, we introduce the variable $z_j(k)$, let $z_j(k)=x_j(k)\cdot b_j(k)$. This equivalence can be expressed by the following inequalities:

$$z_j(k)\le M_b\cdot b_j(k) \tag{5}$$

$$z_j(k)\ge m_b\cdot b_j(k) \tag{6}$$

$$z_j(k)\le x_j(k)-m_b\cdot\left(1-b_j(k)\right) \tag{7}$$

$$z_j(k)\ge x_j(k)-M_b\cdot\left(1-b_j(k)\right) \tag{8}$$

The constant M_b and m_b is the upper and lower limit of the buffer capacity respectively. The nonlinear constraint $z_j(k)=x_j(k)\cdot b_j(k)$ can be expressed by a series of linear inequalities as shown in equations (5)-(8).

Meanwhile, equation (1) is updated as follows:

$$x_j(k+1)=z_j(k)-v_j(k)\cdot\Delta t\cdot b_j(k),\forall k,j\in S_1 \tag{9}$$

3.2.2. *Machine $j\in S_h(h\ge 2)$*

Next, we consider the case of machine $j\in S_h, h=2,\ldots,H$, meaning that machine j is not in the first stage. In this case, the buffer of machine j contains not only incoming jobs from the upstream waiting to be processed but also jobs that have been processed and are being sent downstream.

Firstly, we define $D_j(k)$ as the number of processed jobs leaving the buffer of machine j from time instant k to $k+1$. Similar to equations (1)-(2), $D_j(k)$ can be expressed as: $j\in S_h, h=1,\ldots,H$,

$$D_j(k)=v_j(k)\cdot b_j(k)\cdot\Delta t+x_j(k)\cdot(1-b_j(k)), \tag{10}$$

Equation (10) represents the constraint on the number of processed jobs leaving the buffer of machine j from time instant k to $k+1$, which is determined by the relationship between the remaining jobs in the buffer and the processing capacity. Here, $b_j(k)$ follows the expression in equation (2). When the number of jobs in the buffer is sufficient, $D_j(k)$ can be directly expressed as the number of jobs processed per unit time, denoted as $v_j(k) \cdot \Delta t$. Otherwise, it is equal to the remaining number of jobs in the buffer. The equivalent transformation of $b_j(k)$ is given by equations (3) and (4).

Based on $D_j(k)$ and the definition of $C_{i,j}(k)$, the dynamic model of the change in the number of jobs in the buffer of machine $j \in S_h$ $(h = 2, .., H)$ can be described as: $j \in S_h, h = 2, .., H,$

$$x_j(k+1) = x_j(k) + \sum_{i \in S_{h-1}} C_{i,j}(k) - D_j(k), \quad (11)$$

In Figure 3, the number of jobs within the machine represents the number of jobs that can be processed by the machine in a unit time interval. From Figure 3, we can observe that the number of jobs in the buffer at each moment is composed of three parts: the initial number of jobs in the buffer, the incoming jobs into the buffer, and the jobs leaving the buffer to be processed inside the machine.

Furthermore, the variable $C_{i,j}(k)$ maintains the following balance equation between the upstream and downstream:

$$\sum_{j \in S_{h+1}} C_{i,j}(k) = D_i(k), \forall i \in S_h, h = 1, .., H-1, \quad (12)$$

The completed jobs processed by machine i are transferred to the buffer of the downstream machine. The sum of the jobs transferred to the buffers of all downstream machines represents the total number of jobs processed by machine i at that moment, denoted as $D_i(k)$. As shown in Figure 4, the sum of the jobs transferred from machine 1 to machines 3 and 4 represents the total number of jobs processed by machine 1.

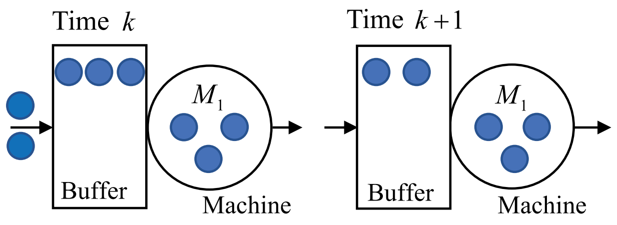


Figure 3. Diagram of machine buffer state changes considering processing inputs and outputs.

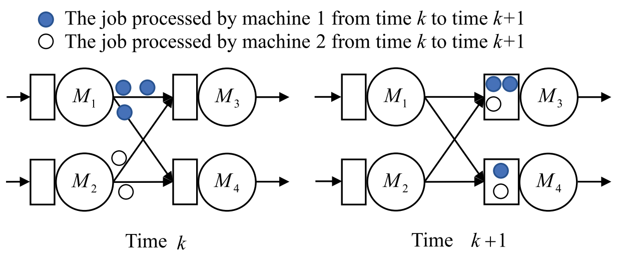

● The job processed by machine 1 from time k to time k+1
○ The job processed by machine 2 from time k to time k+1
M_1
M_2
M_3
M_4
Time k
M_1
M_2
M_3
M_4
Time $k+1$

Figure 4. Diagram of upstream processed jobs entering downstream buffers.

2.2.3. MLD model

In this section, the MLD model of dynamic task assignment is presented by defining appropriate variable vectors.

Let: $\mathbf{x}^T(k) = [x_1(k), x_2(k), ..., x_{|S|}(k)],$

$$\mathbf{u}^T(k) = [C_{1,|S_1|+1}(k), \ldots, C_{|S_1|,|S_1|+|S_2|}(k), \ldots, C_{|S|-|S_H|,|S|}(k), D_1(k), \ldots, D_{|S|}(k)]$$
$$\boldsymbol{\delta}^T(k) = [b_1(k), ..., b_{|S|}(k)]$$
$$\mathbf{z}^T(k) = [z_1(k), ..., z_{|S|}(k)],$$

Where $\mathbf{x}(k)$ is the system state vector, $\mathbf{u}(k)$ is the control variable vector, $\boldsymbol{\delta}(\mathbf{k})$ is the logical decision variable vector, and $\mathbf{z}(k)$ is the auxiliary decision variable vector. Based on the vector defined above [31], the hybrid flow shop dynamic task allocation model expressed by equations (3)-(11) can be written as the following more compact mixed logic dynamic model:

$$\mathbf{x}(k+1) = \mathrm{A}\mathbf{x}(k) + \mathrm{B}_1\boldsymbol{\delta}(k) + \mathrm{B}_2\mathbf{z}(k) + \mathrm{B}_3\mathbf{u}(k) \quad (13)$$

$$\mathrm{E}_1\boldsymbol{\delta}(k) + \mathrm{E}_2\mathbf{z}(k) \le \mathrm{E}_3\mathbf{x}(k) + \mathrm{E}_4\mathbf{u}(k) + \mathrm{E}_5 \quad (14)$$

Matrices A, B_1-B_3 and E_1-E_4 in equations (13)-(14) represent real parameter matrices, and E_5 is a real-valued vector. The MLD model is a dynamic model, which can describe the dynamic distribution process of hybrid flow shop by introducing linear dynamic inequality of discrete variables. Advanced control methods such as model predictive control can be further used to improve the efficiency of job allocation.

To facilitate the expression of the matrices, we let $U_j(k) \overset{\Delta}{=} v_j(k) \cdot \Delta t$, and the explicit expressions of matrices A, B_1-B_3 and E_1-E_5 are shown in Appendix 1.

4. Model predictive controller design

In this section, a model predictive controller for dynamic task allocation scheduling optimization of a hybrid flow shop is introduced. First, the basic principle of the controller is introduced, and then the mathematical description of MPC optimization is given.

4.1. Basic principle

Model Predictive Control originates from discrete-time optimal control methods and employs rolling optimization strategies within a finite time horizon for online optimization and control. It is widely applied in various fields such as chemical engineering, transportation, and logistics [32]. MPC utilizes dynamic models to predict future states in constrained and complex dynamic systems, considering constraints and minimizes an objective

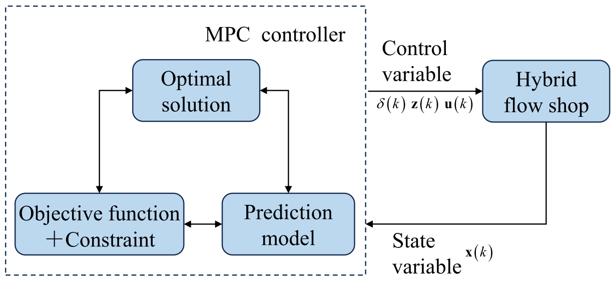


Figure 5. Online task allocation strategy for hybrid flow shop based on MPC.

function to compute decision variables within a limited predictive horizon. MPC is capable of handling varying constraints and nonlinear systems, providing stable and optimized control effects over the long term.

Figure 5 illustrates the online task allocation strategy in a hybrid flow shop based on MPC. At any given time instant k, the MPC controller solves an optimization problem that maximizes production efficiency based on the detected system state $\mathbf{x}(\mathbf{k})$, using the predictive MLD model and constraint conditions. The proposed MPC employs a local planning strategy, continuously solving optimization problems for different time instants k in a rolling manner, decomposing the global planning into smaller local planning problems to reduce computational complexity. Furthermore, this online optimization strategy can handle uncertainties in the production system, such as machine failures or newly arrived jobs.

4.2. MPC optimization problem description

The MPC optimization problem is formulated based on the proposed MLD model and is solved through rolling optimization within a predictive horizon, N_p, to achieve dynamic task allocation in the hybrid flow shop. The constructed MPC optimization control problem is represented as follows:

$$\min_{\mathbf{x}(k+l),\mathbf{u}(k+l),\boldsymbol{\delta}(k+l),z(k+l)} \sum_{l=0}^{N_p-1} J(k+l+1) \tag{15}$$

Be bound to:

$$\begin{aligned}
&\mathbf{x}(k+l+1) = \mathbf{A}\mathbf{x}(k+l) + \mathbf{B}_1\mathbf{u}(k+l) + \mathbf{B}_2\boldsymbol{\delta}(k+l) \\
&\qquad\qquad + \mathbf{B}_3\mathbf{z}(k+l) \\
&\mathbf{E}_2\boldsymbol{\delta}(k+l) + \mathbf{E}_3\mathbf{z}(k+l) \le \mathbf{E}_1\mathbf{u}(k+l) + \mathbf{E}_4\mathbf{x}(k+l) + \mathbf{E}_5 \\
&\mathbf{u}_{\min} \le \mathbf{u}(k+l) \le \mathbf{u}_{\max} \\
&\mathbf{x}_{\min} \le \mathbf{x}(k+l+1) \le \mathbf{x}_{\max}, l = 0, 1, \ldots, N_p - 1
\end{aligned} \tag{16}$$

Where, $\mathbf{x}(k+l+1)$ is the predicted state of the system at time $k+l+1$ based on decision variable, $\mathbf{u}_{\min}$, $\mathbf{u}_{\max}$, $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$ respectively represent the boundaries of the system's inputs and states. The state variable $\mathbf{x}(k+1)$ is directly influenced by the state variable $\mathbf{x}(k)$ and the inputs $\boldsymbol{\delta}(k)$, $\mathbf{u}(k)$ and $\mathbf{z}(k)$. And the objective function $J(k+1)$ is expressed as:

$$J(k+1) = \sum_{i\in S} x_i(k) \tag{17}$$

The objective function of the MPC optimization problem (equation (15)) aims to process all jobs as quickly as possible, which is translated into minimizing the total number of jobs in all buffers at all time instants, as shown in equation (17). Equations (15)-(16) represent a mixed integer linear programming (MILP) problem, which can be solved using commercial solvers such as Cplex or Gurobi.

In the constructed MPC optimization problem, the state of the machine buffer at the initial time $k = 0$ is set as follows:

$$x_j(0) = N/|S_1|, \quad j \in S_1, \tag{18}$$

$$x_j(0) = 0, \quad j \in S_h, h = 2, \ldots, H. \tag{19}$$

Formula (18) indicates that the number of jobs to be processed in each machine buffer of the first stage ($h = 1$) is evenly distributed as the total number of jobs N, and formula (19) indicates that the number of jobs in each machine buffer of the non-first stage ($h \ge 1$) is 0 at the initial moment $k = 0$.

5. Case studies

This section validates the effectiveness of the proposed dynamic task allocation method for the hybrid flow shop based on the MLD model. Firstly, the scenarios of the case study are configured, and an appropriate predictive horizon is selected. Then, the results of the case study under different scenarios are analyzed, and simulation validation is conducted using Plant Simulation software.

5.1. Scenario configuration

To validate the effectiveness of the proposed dynamic task allocation method, 15 scenarios of a three-stage hybrid flow shop are selected for case studies, as shown in Table 2. The three-stage setting allows for the consideration of task allocation across multiple stages while simplifying validation. Based on the reference [33], the number of jobs and the processing rates of machines are set. The processing rates of machines are randomly sampled from the range [1,12], measured in units of jobs per hour. The specific time interval between k and $k+1$ is equally denoted as the parameter Δt, which is set to be 1 hour. Three production layouts are defined: divergent (increasing the number of machines in each stage with the stage number), convergent (decreasing the number of machines in each stage with the stage number), and mixed (no clear relationship between the number of machines and the stage number).

Table 2. Machine number configuration in a three-stage hybrid flow shop scenario.

Scenario	Total number of machines	Stage 1	Stage 2	Stage 3	Layout type
Scenario 1	9	2	3	4	divergent
Scenario 2	9	4	3	2	convergent
Scenario 3	9	4	2	3	mixed
Scenario 4	20	4	6	10	divergent
Scenario 5	20	10	6	4	convergent
Scenario 6	20	4	10	6	mixed
Scenario 7	30	8	10	12	divergent
Scenario 8	30	12	10	8	convergent
Scenario 9	30	10	12	8	mixed
Scenario 10	40	10	12	18	divergent
Scenario 11	40	15	14	11	convergent
Scenario 12	40	12	10	18	mixed
Scenario 13	50	12	18	20	divergent
Scenario 14	50	20	18	12	convergent
Scenario 15	50	10	25	15	mixed

In this study, the performance of the proposed MPC method is compared to three methods (global optimization method and two rule-based methods) under 15 different scenarios listed in Table 2. Both global optimization and rule-based methods are commonly used approaches in the literature to solve the HFSP [8]. Global optimization is a commonly utilized offline planning approach for the hybrid flow shop scheduling problem. It involves deciding on decision variables that encompass all operations of the hybrid flow shop, with no updates during the entire planning process. The global optimization is formulated using constraints (3)-(13) and objective (17), with a sufficiently large number of planning horizons. This formulated problem is a mixed integer linear programming, which is then solved using the Gurobi solver.

The first rule-based method is based on the machine's processing capacity in the next stage, which is referred to as the Processing Capacity (PC) allocation method. If the allocation is not an integer, it is rounded, and the allocation is prioritized based on the processing capacity of the machines. Jobs are allocated to machines with higher processing capacity first until all jobs are allocated. Another rule-based method is a greedy approach, in which, at each moment, the job that completes processing is assigned to the machine with the minimum next-stage workload and the shortest processing time. If multiple machines have the same workload and processing time, a machine is randomly selected until all the jobs are allocated.

The comparisons are conducted for a total of 100, 200, and 400 jobs. Two important metrics, namely the completion time and the solution time, are used to evaluate the effectiveness of the proposed methods. The computation time of the proposed MPC method refers to the average solution time for solving the MILP problem.

The MLD model is implemented using the Pyomo toolbox [34] in Python language, installed on a Windows operating system. Gurobi 9.0.3 is utilized to solve the MPC optimization problems and the global offline optimization problem discussed in this paper. The computational hardware consists of an Intel Core i7-9700 (3.0 GHz) processor and 16GB of memory.

5.2. Predictive horizon selection

In model predictive control, the prediction horizon N_p is an important parameter that represents the extent to which the controller predicts future states. If N_p is chosen to be too large, the optimization problem's size increases, resulting in longer computation times. On the other hand, if N_p is chosen to be too small, the predictive model may not cover all elements of the system, resulting in reduced operational efficiency. Therefore, selecting an appropriate N_p is crucial.

To test the scenario of processing 100 jobs in Scenario 1 from Table 2, different values of N_p are used, and the completion time and computation time are recorded. The results are shown in Table 3. The results in Table 3 indicate that the completion time converges when $N_p \geq 9$. In this study, $N_p = 9$ is selected because it provides the minimum completion time with the shortest solution time.

Table 3. Performance index of different prediction horizon N_p.

N_p	Completion time (unit:hour)	Solution time (unit:second)
5	25	0.11
6	21	0.12
7	23	0.15
8	20	0.17
9	17	0.19
10	17	0.21
11	17	0.23
12	17	0.25
13	17	0.27
14	17	0.29

5.3. Experimental results

In this subsection, we first analyze the results under the assumption of no interference in the production process. Then, we discuss the superiority of the proposed dynamic task allocation method under two types of interference: machine failures and the arrival of new jobs.

5.3.1. No interference case

Table 4 presents the performance comparison of the four methods under the 15 scenarios defined in Section 5.1, assuming no interference in the production process. The results in Table 4 demonstrate that, in the absence of disturbances in the production process, the proposed MPC method achieves the same processing completion time as the global offline optimization method while significantly reducing the computation time. The MPC method achieves this by continuously solving small-scale MILP problems online within a smaller predictive horizon, thereby reducing the computational burden compared to global optimization. The PC and the greedy methods have a shorter computation time but may lead to a situation where jobs are only allocated to the machines with the highest processing capacity or the lowest workload in the next stage, due to the lower number of jobs processed in the previous stage. This results in a significant increase in the planned processing completion time compared to the previous two methods.

5.3.2. Machine failure case study

The previous section demonstrated the advantage of the proposed MPC method in terms of computation time. Now, we present the computational results of the MPC method during the production process under machine failure conditions. We select 9 out of the 15 scenarios from Table 2 for analysis. For each scenario, the occurrence time and machine number of the failures are randomly generated, as shown in Table 5. The

Table 4. Performance comparison of the proposed MPC method with three methods in different scenarios.

		Completion time(unit:hour)				Solution time(unit:second)			
Scenario	Number of jobs	Global optimization	MPC	PC	Greedy	Global optimization	MPC	PC	Greedy
1	100	15	15	17	24	0.21	0.09	0.007	0.007
1	200	27	27	32	47	0.37	0.12	0.014	0.016
1	400	52	52	62	92	0.76	0.22	0.037	0.047
2	100	27	27	27	27	0.50	0.15	0.006	0.006
2	200	52	52	52	52	0.97	0.31	0.016	0.017
2	400	102	102	102	102	1.89	0.47	0.038	0.040
3	100	17	17	17	30	0.26	0.14	0.008	0.007
3	200	31	31	31	59	0.56	0.23	0.021	0.018
3	400	60	60	60	116	1.05	0.42	0.049	0.046
4	100	12	12	24	25	0.53	0.36	0.006	0.006
4	200	22	22	44	48	1.15	0.63	0.025	0.024
4	400	42	42	86	93	2.10	1.24	0.056	0.055
5	100	22	22	35	35	0.99	0.42	0.006	0.006
5	200	42	42	69	69	1.96	0.83	0.028	0.028
5	400	82	82	135	135	4.70	2.03	0.069	0.069
6	100	15	15	20	22	0.69	0.25	0.006	0.006
6	200	27	27	38	43	1.58	0.35	0.023	0.025
6	400	52	52	77	83	5.11	0.60	0.056	0.060
7	100	15	15	21	21	1.12	0.27	0.006	0.006
7	200	27	27	40	40	2.03	0.63	0.030	0.030
7	400	52	52	79	79	16.7	1.12	0.058	0.058
8	100	12	12	25	25	1.03	0.53	0.005	0.005
8	200	21	21	48	48	2.61	1.07	0.012	0.012
8	400	39	39	91	91	6.40	3.44	0.058	0.058
9	100	12	12	23	23	1.01	0.59	0.005	0.005
9	200	22	22	45	45	3.32	0.87	0.017	0.017
9	400	42	42	87	87	30.94	2.41	0.071	0.071
10	100	12	12	22	22	1.42	0.73	0.007	0.007
10	200	22	22	41	41	2.41	1.43	0.027	0.027
10	400	42	42	82	82	28.06	2.95	0.072	0.072
11	100	11	11	51	51	5.51	0.60	0.005	0.005
11	200	19	19	102	102	20.16	1.17	0.013	0.013
11	400	36	36	201	201	68.64	2.63	0.063	0.063
12	100	11	11	39	39	1.23	0.35	0.006	0.006
12	200	19	19	75	75	2.22	0.37	0.025	0.025
12	400	36	36	149	149	7.20	0.41	0.063	0.063
13	100	10	10	35	35	1.76	0.88	0.004	0.004
13	200	19	19	67	67	3.84	2.69	0.012	0.012
13	400	35	35	138	138	49.56	6.19	0.063	0.063
14	100	10	10	43	43	1.87	1.06	0.005	0.005
14	200	17	17	81	81	6.17	1.52	0.012	0.012
14	400	31	31	159	159	86.95	5.34	0.047	0.047
15	100	12	12	35	35	3.21	0.87	0.005	0.005
15	200	22	22	68	68	4.41	0.92	0.017	0.017
15	400	42	42	136	136	116.57	1.64	0.066	0.066
Average		29.9	29.9	64.7	68.6	11.15	1.15	0.028	0.028

Table 5. Comparison of processing completion rates for machine failure scenarios.

Case	Scenario	Number of jobs	Occurrence Time of failure	Faulty machine	Processing completion rate			
					Global optimization	MPC	PC	Greedy
1	Scenario 2	50	$k=5$	M_6	60%	88%	82%	82%
2	Scenario 3	50	$k=4$	M_5	74%	82%	72%	80%
3	Scenario 4	50	$k=2$	M_5	76%	90%	84%	88%
4	Scenario 6	50	$k=3$	M_{15}	76%	92%	90%	88%
5	Scenario 7	50	$k=2$	M_{13}	82%	88%	76%	76%
6	Scenario 8	50	$k=2$	M_{23}	76%	92%	88%	88%
7	Scenario 9	50	$k=3$	M_{24}	78%	88%	88%	88%
8	Scenario 12	50	$k=2$	M_{14}	86%	92%	58%	58%
9	Scenario 15	50	$k=2$	M_{41}	68%	92%	92%	92%
Average					75.1%	89.3%	81.1%	82.2%

total number of processed jobs for each scenario is set to 50. The completion rates of job processing are compared among the global optimization, MPC, PC, and Greedy allocation methods.

In the given scenario layout of case study 1, with an initial set of 50 jobs to be processed, let's assume that machine 6 experiences a failure at time instant $k = 5$. The MPC-based method is capable of promptly adjusting the planning by redirecting the remaining jobs to other machines for processing. However, in the global optimization allocation methods, machines are unable to adapt their allocation plans based on the failure situation. Consequently, the jobs continue to be assigned to the faulty machine according to the original plan, even though the machine is incapable of processing them. As a result, the number of jobs in the buffer ahead of the faulty machine keeps accumulating, and the subsequent machines do not receive any jobs from that machine. Figure 6 illustrates the changes in the number of jobs in the buffers of each machine for the MPC-based, global optimization, PC, and Greedy methods, respectively. Both the PC and the Greedy methods update the allocation scheme and continue the allocation process based on their respective allocation principles.

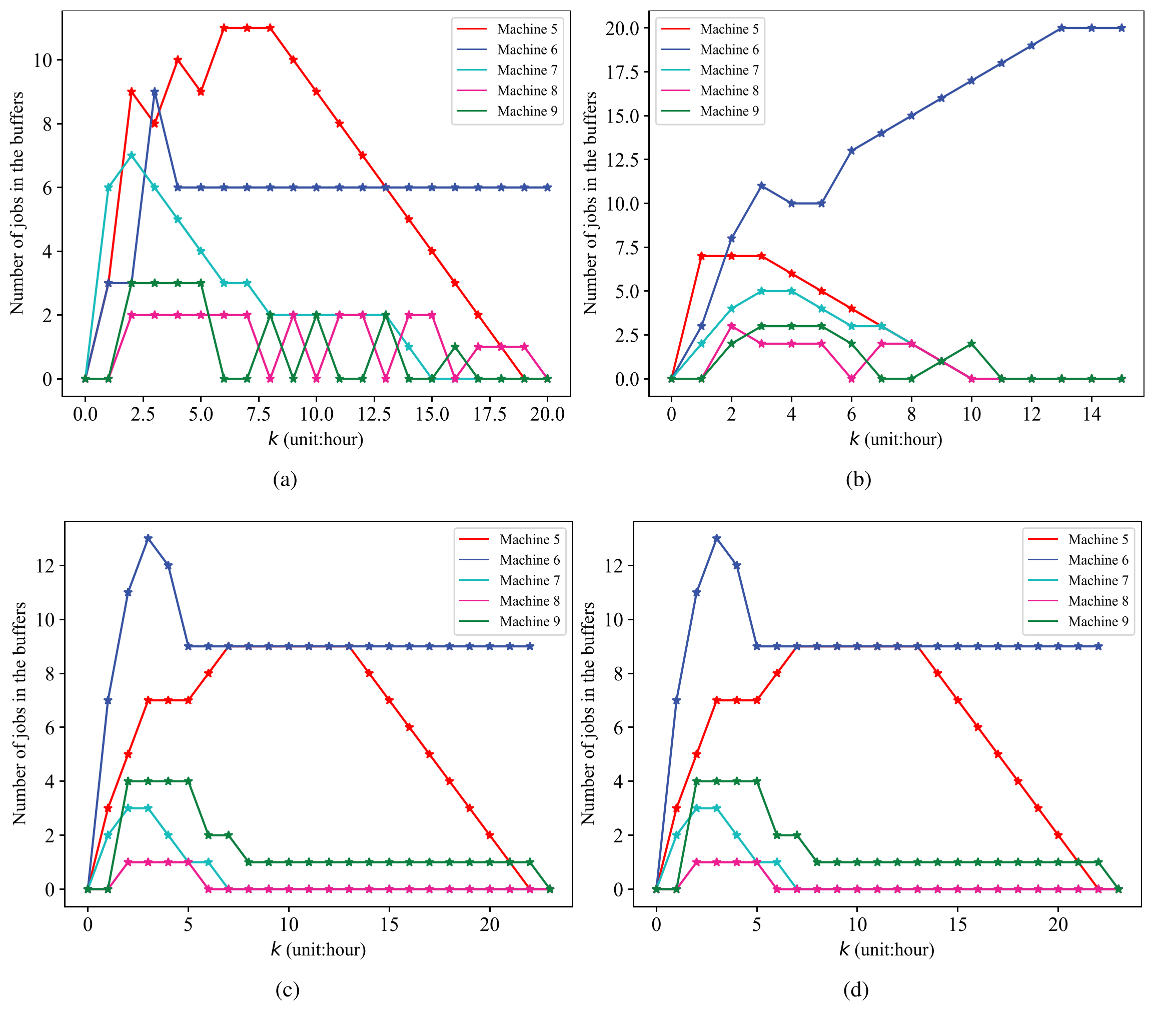


Figure 6. Number of jobs in the buffer over time of four methods for the machine failure scenario 1. (a) MPC (b) Global optimization (c) PC (d) Greedy.

From the figures, it can be observed that in the MPC-based method, after machine 6 encounters a failure, machines from the previous stage will stop allocating jobs to the faulty machine. The faulty machine is unable to process the jobs in its buffer, so the number of jobs in the buffer remains constant after time instant 5 in the figure. Both the PC and Greedy allocation methods also make adjustments to the allocation scheme in a timely manner. However, allocating jobs based on the allocation rules before a failure occurs may lead to a higher number of jobs being assigned to the faulty machine. Consequently, the job completion rate may be lower. However, in the case of global optimization, the number of jobs in the buffer of machine 6 continues to increase according to the original plan, even though the machine is experiencing a failure. As a result, the completion rates of job processing significantly decrease. By calculation, MPC achieves a completion rate of 88%, global optimization achieves a completion rate of 60%, PC allocation achieves a completion rate of 82%, and Greedy achieves a completion rate of 82%.

5.3.3. *Newly arrived job case study*

Here, we consider 9 scenarios out of the 15 scenarios listed in Table 2. For each scenario, we randomly generate the number and arrival time of newly arrived jobs, as shown in Table 6. The completion time of job processing are compared among the four methods.

In the layout scenario of Case 4, the number of jobs to be processed is 50 at the given initial moment, and 20 new jobs arrive at the fourth moment. The line plots of the change in the number of jobs within the buffer obtained by four different methods are shown in Figure 7.

In the MPC-based method, the number of newly arrived jobs can be directly added to the number of jobs to be processed to resolve the optimal distribution mode. However, once the global optimization methods start planning, there is no way to take into account the dynamic situation of the arrival of new jobs. Therefore, they can only allocate the new jobs after all the initial jobs in the buffer have been processed and left the machine. As a result, their planned completion time will be longer than those achieved with the MPC-based method. The PC and Greedy methods allocate jobs to machines with the highest processing capacity or the lowest workload and shortest processing time, according to the allocation rules, which results in a longer completion time. From Figure 7, it can be seen that in the global optimization method, the completion time for the jobs is 14 hours, while in the PC and Greedy methods, the completion time is 15 and 16 hours respectively. However, in the MPC-based method, the completion time for the jobs is only 11 hours.

5.3.4. *Multiple uncertain event case study*

Here, 6 scenarios have been selected from the 15 scenarios in Table 2, where each scenario involves multiple machine failures occurring simultaneously. Each scenario has 50 jobs to process. We also consider situations where both machine failures and new job arrivals occur simultaneously. At the initial time, there are 50 jobs waiting to be processed. The completion rates of the four methods have been compared, as shown in Tables 7 and 8. From Tables 7 and 8. it can be observed that the MPC-based allocation method outperforms the other three methods in handling multiple machine failures and scenarios where both machine failures and the arrival of new jobs are involved.

5.4. *Simulation verification*

In this section, Plant Simulation 16.0 based on discrete event simulation is used to simulate and verify the dynamic task assignment problem in the hybrid flow shop. Plant Simulation is an object-oriented discrete event simulation tool, which can realize the simulation and optimization of complex factory, production line, and production logistics processes [35]. The software can simulate the processing flow in the mixed flow workshop and complete the task assignment process.

First of all, the simulation model is established based on the layout scenario in the hybrid flow shop system. The objects required to establish the hybrid flow shop system model include material sources, buffers, stations, connectors, material terminations, etc. The allocation results obtained from the program are then implemented in the

Table 6. Comparison of the completion times for newly arrived job case studies.

					Completion time (unit:hour)			
Case	Scenario	Initial number of jobs	Number of newly arrived jobs	Arrival time	Global optimization	MPC	PC	Greedy
1	Scenario 2	50	12	$k = 7$	20	18	18	18
2	Scenario 3	50	16	$k = 4$	15	12	12	21
3	Scenario 4	50	20	$k = 3$	11	9	20	19
4	Scenario 6	50	20	$k = 4$	14	11	15	16
5	Scenario 7	50	24	$k = 2$	14	8	17	17
6	Scenario 8	50	12	$k = 5$	13	10	18	18
7	Scenario 9	50	12	$k = 4$	11	9	16	16
8	Scenario 12	50	12	$k = 3$	10	8	25	25
9	Scenario 15	50	20	$k = 2$	11	7	27	27

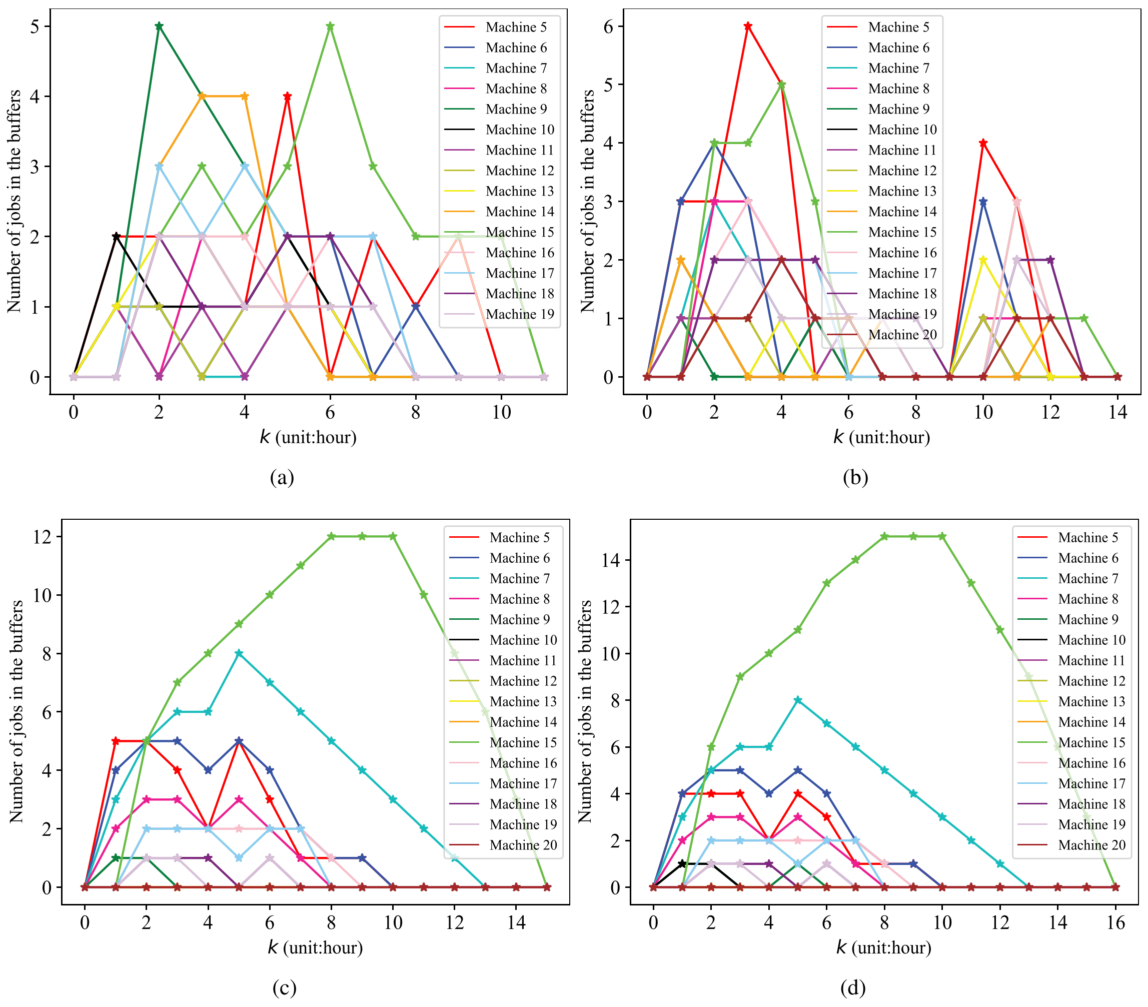


Figure 7. Number of jobs in the buffer over time of four methods for the newly arrived job scenario 6. (a) MPC (b) Global optimization (c) PC (d)Greedy.

Table 7. Comparison of the processing completion rates under multiple machine failure scenarios.

					Processing completion rate			
Case	Scenario	Number of jobs	Occurrence Time of failure	Faulty machine	Global optimization	MPC	PC	Greedy
1	Scenario 2	50	$k = 6$	M_5, M_6	58%	72%	70%	70%
2	Scenario 3	50	$k = 5$	M_6, M_7	56%	78%	52%	78%
3	Scenario 4	50	$k = 2$	M_5, M_6, M_{13}	42%	86%	74%	78%
4	Scenario 6	50	$k = 3$	M_5, M_6, M_{15}	58%	80%	76%	78%
5	Scenario 7	50	$k = 2$	M_9, M_{15}, M_{23}	68%	78%	78%	78%
6	Scenario 8	50	$k = 2$	M_{13}, M_{15}, M_{23}	58%	76%	62%	62%
Average					56.7%	78.3%	68.7%	74%

Table 8. Comparison of the processing completion rates under multiple uncertain event scenarios.

			Processing completion rate			
Case	Scenario	Uncertain events: newly arrived job and machine failure	Global optimization	MPC	PC	Greedy
1	Scenario 2	$k = 4{,}8$, M_6	46.6%	89.7%	79.3%	81.0%
2	Scenario 3	$k = 4{,}12$, M_5	59.7%	85.5%	77.4%	83.9%
3	Scenario 4	$k = 3{,}6$, M_5	73.2%	92.9%	83.9%	89.3%
4	Scenario 6	$k = 3{,}6$, M_{15}	67.9%	92.9%	91.1%	89.3%
5	Scenario 7	$k = 2{,}8$, M_{13}	70.7%	89.7%	79.3%	79.3%
6	Scenario 8	$k = 2{,}6$, M_{23}	67.9%	92.9%	89.3%	89.3%
Average			64.3%	90.6%	83.4%	85.4%

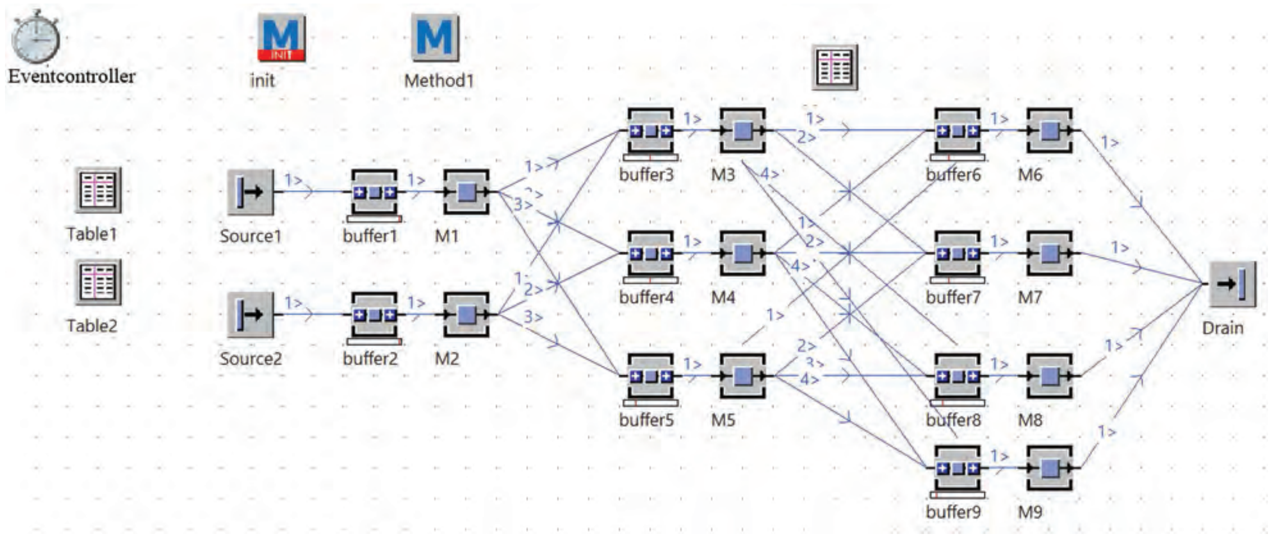


Figure 8. Schematic diagram of a three-stage hybrid flow shop modeled in plant simulation software.

simulation software. A list is set up at the material source to define the name of each job and is set at the exit of the machine so that after it is finished, it is designated to transport to the next stage of the buffer before a machine and waits for processing. The same procedure is repeated for the next stage by setting the destination buffer at the machine's exit, and so on, until all jobs are allocated according to the results. Finally, using the Simtalk programming language in Plant Simulation software, the number of jobs in each buffer is recorded hourly using the event controller's time and written into a list. The data from the list is then used to plot the line graph showing the variation in the number of jobs in each machine's buffer over time.

The MPC numerical solution results are simulated in Plant Simulation for three scenarios: no interference, machine failure, and new job arrivals. Under the configuration layout of scenario 1, with 20 jobs to be processed, the constructed Plant Simulation model is shown in Figure 8. In the absence of interference, the line graph of the number of jobs in the buffer based on the MPC numerical solution results is depicted in Figure 9(a). The corresponding line graph of the number of jobs in the buffer obtained through the simulation in Plant Simulation is shown in Figure 9(b). By comparing Figures 9(a,b) we can observe that the obtained line graphs of the number of jobs in the buffer are consistent, validating the correctness of the theoretical part.

The simulation validation in Plant Simulation is conducted in the case of machine failure. Under the layout configuration of scenario 3, let's assume that machine 5 experiences a failure at time instant 2. The line graph of the number of jobs in the buffer based on the MPC numerical solution results is shown in Figure 10(a). The corresponding

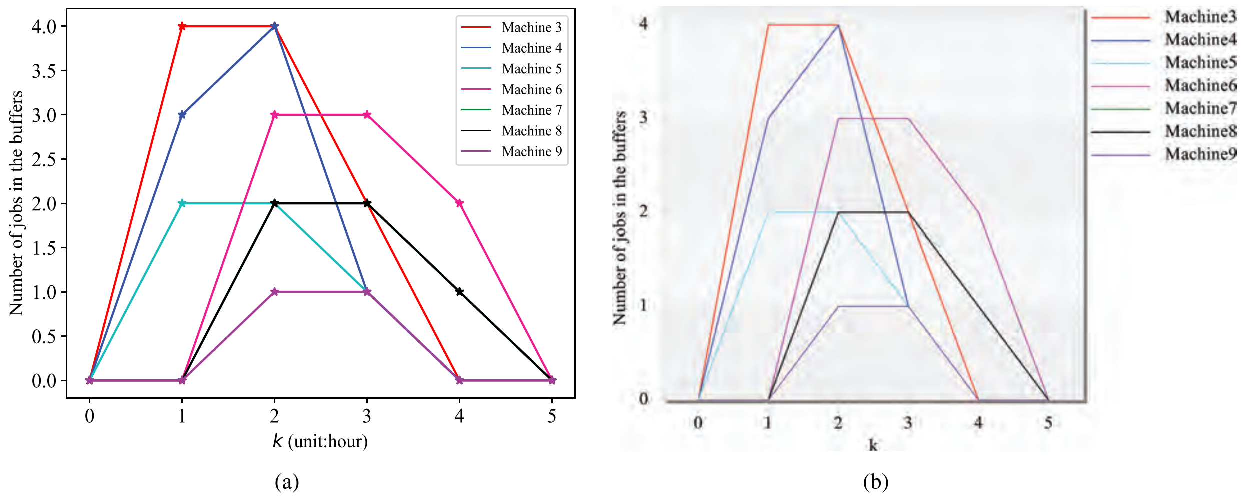


Figure 9. Curves of the number of jobs in the buffer over time obtained by the MPC method and plant simulation for scenario 1 without disturbances. (a) MPC. (b) Plant Simulation.

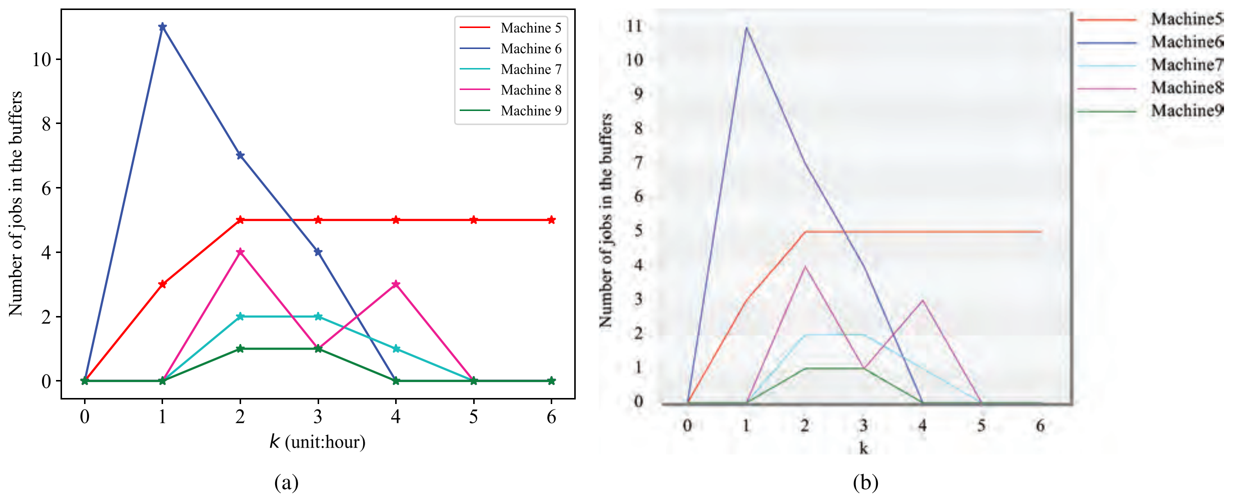


Figure 10. Curves of the number of jobs in the buffer over time by the MPC method and plant simulation for machine failure's scenario 3. (a) MPC. (b) Plant Simulation.

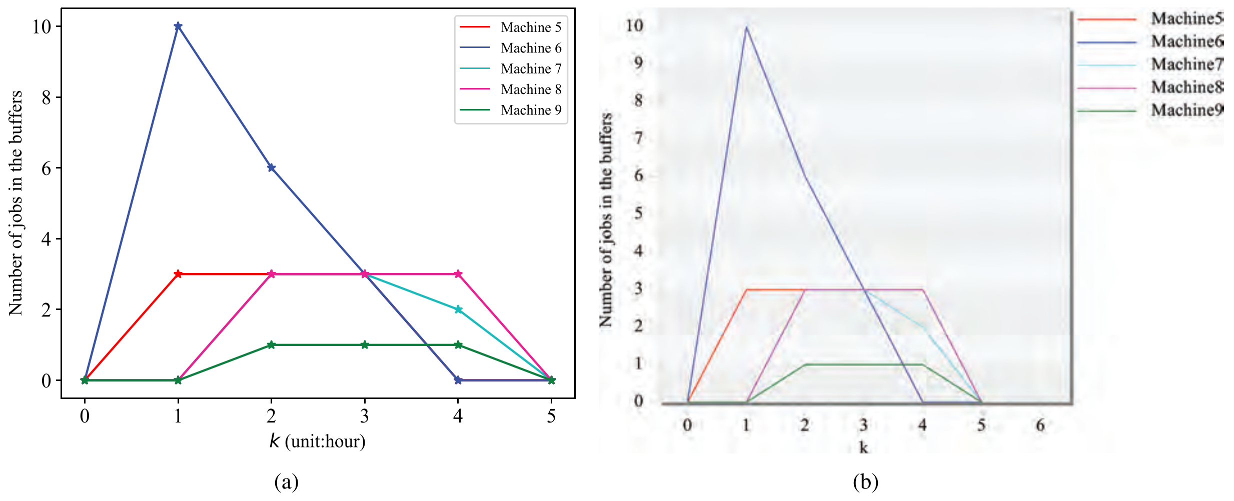


Figure 11. Curves of the number of jobs in the buffer over time by the MPC method and Plant Simulation for newly arrived job's scenario 3. (a) MPC. (b) Plant Simulation.

line graph of the number of jobs in the buffer obtained through the simulation in Plant Simulation is depicted in Figure 10(b). By comparing the two graphs, we can observe that they are consistent, validating the correctness of the proposed method.

The simulation validation in Plant Simulation is performed in the case of new job arrivals. Under the layout configuration of scenario 3, let's assume that there are initially 16 jobs to be processed, and at time instant 2, 4 new jobs arrive. The line graph of the number of jobs in the buffer based on the MPC numerical solution results is shown in Figure 11(a). The simulated line graph of the number of jobs in the buffer is depicted in Figure 11(b). By comparing the two graphs, we can observe that they are consistent, validating the correctness of the proposed method.

6. Conclusion and future research

In this paper, the dynamic task assignment problem of hybrid flow shop for machines in parallel with different speeds is studied, the MLD model of production system state is established, and the dynamic task assignment method based on MPC is proposed. This method decomposes the global planning problem into a smaller local planning model and transforms offline scheduling into an online scheduling mode, which can deal with the uncertain situation in the production process. Extensive numerical experiments demonstrate that this method outperforms traditional global optimization and rule-based allocation methods in terms of processing completion time and completion rate. Furthermore, the proposed method is validated by modeling and simulating the considered hybrid flow shop case using the production process simulation software, Plant Simulation. The numerical results obtained from the proposed method align with the simulation software results, confirming the correctness of the proposed method.

Future research will build upon the model proposed in the paper and consider the relationship between variable processing rates and energy consumption. It will focus on studying the bi-objective optimization problem of completion time and energy consumption and develop efficient optimization algorithms to solve it.

Acknowledgments

This research is supported in part by the National Natural Science Foundation of China under Grant 62,173,311 and 72,201,252, and in part by College Youth Backbone Teacher Project of Henan Province under Grant 2021GGJS001.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the National Natural Science Foundation of China [62173311 and 72201252].

Notes on contributors

Jianbin Xin (Member, IEEE) received the B.Sc. degree in electrical engineering from Xidian University, China, in 2007, the M.Sc. degree in control science and engineering from Xi'an Jiaotong University, China, in 2010, and the Ph.D. degree from the Department of Maritime and Transport Technology, Delft University of Technology, The Netherlands, in 2015.Currently, he is an Associate Professor with the Department of Automation, Zhengzhou University, China. His research interests include planning and control of smart logistics systems and cooperative robots.

Sixuan Li received a bachelor's degree in Automation from Zhengzhou University in 2021, where she is currently pursuing a master's degree at the School of Electrical and and Information Engineering. Her research interests include task allocation and vehicle allocation for manufacturing.

Yanjie Zhou received Ph.D. degree from the Department of Industrial Engineering at Pusan National University in 2020 and received B.S. Degree and M.S. Degree in Computer Science and Computer Applied Technology from Zhengzhou University in 2012 and 2015, respectively. He is currently an associate professor with the School of Management at Zhengzhou University. He focuses on solving real-world optimization problems by using artificial intelligence techniques.

Andrea D'Ariano received the B.S. and M.S. degrees in computer science, automation, and management engineering from Roma Tre University and the Ph.D. degree from the Department of Transport and Planning, Delft University of Technology, in April 2008, under the supervision of Prof. I. A. Hansen.Currently, he is a Full Professor at the Department of Engineering, Roma Tre University. His research interests include the study of scheduling problems with application to public transportation and logistics. He is the Associate Editor of well-known international journals, such as Transportation Research—B: Methodological, Transportation Research—C: Emerging Technologies, and Transportation Research—E: Logistics and Transportation Review, and conferences, such as IEEE ITSC.

ORCID

Yanjie Zhou http://orcid.org/0000-0003-2222-9140

References

[1] Neufeld JS, Schulz S, Buscher U. A systematic review of multi-objective hybrid flow shop scheduling[J]. Eur J Oper Res. 2023;309(1):1–23.

[2] Shao W, Shao Z, Pi D. Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem[J]. Knowledge-Based Syst. 2020;194:105527.

[3] Lin JT, Chen CM. Simulation optimization approach for hybrid flow shop scheduling problem in semiconductor back-end manufacturing[J]. Simul Modell Pract Theory. 2015;51:100–114.

[4] Wang CN, Hsu HP, Fu HP, et al. Scheduling flexible flow shop in labeling companies to minimize the makespan [J]. Comput Syst Sci Eng. 2022;40(1):17–36.

[5] Johnson SM. Optimal twoand threestage production schedules with setup times included[J]. Nav Res Logist Q. 1954;1(1):61–68.

[6] Kyparisis GJ, Koulamas C. A note on makespan minimization in two-stage flexible flow shops with uniform machines[J]. Eur J Oper Res. 2006;175(2):1321–1327.

[7] Paternina-Arboleda CD, Montoya-Torres JR, Acero-Dominguez MJ, et al. Scheduling jobs on ak-stage flexible flow-shop[J]. Ann Oper Res. 2008;164(1):29–40.

[8] Ribas I, Leisten R, Framian JM. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective [J]. Comput Oper Res. 2010;37(8):1439–1454.

[9] Lin SW, Cheng CY, Pourhejazy P, et al. New benchmark algorithm for hybrid flowshop scheduling with identical machines[J]. Expert Syst Appl. 2021;183:115422.

[10] Ab Rashid MFF, Nik Mutasim MA. Cost-based hybrid flow shop scheduling with uniform machine optimization using an improved tiki-taka algorithm[J]. J Ind Prod Eng. 2024;41(2):182–199.

[11] Kyparisis GJ, Koulamas C. Flexible flow shop scheduling with uniform parallel machines[J]. Eur J Oper Res. 2006;168(3):985–997.

[12] Kazemi H, Mahdavi Mazdeh M, Rostami M, et al. The integrated production-distribution scheduling in parallel machine environment by using improved genetic algorithms[J]. J Ind Prod Eng. 2021;38 (3):157–170.

[13] Garavito-Hernndez EA, Pea-Tibaduiza E, Perez-Figueredo LE, et al. A meta-heuristic based on the Imperialist Competitive Algorithm (ICA) for solving

Hybrid Flow Shop (HFS) scheduling problem with unrelated parallel machines[J]. J Ind Prod Eng. 2019;36(6):362–370.

[14] Meng L, Zhang C, Shao X, et al. More MILP models for hybrid flow shop scheduling problem and its extended problems[J]. Int J P Res. 2020;58(13):3905–3930.

[15] Ruiz R, Vzquez-Rodrguez JA. The hybrid flow shop scheduling problem[J]. Eur J Oper Res. 2010;205(1):1–18.

[16] Moursli O, Pochet Y. A branch-and-bound algorithm for the hybrid flowshop[J]. Int J Prod Econ. 2000;64(1–3):113–125.

[17] Han W, Deng Q, Gong G, et al. Multi-objective evolutionary algorithms with heuristic decoding for hybrid flow shop scheduling problem with worker constraint [J]. Expert Syst Appl. 2021;168:114282.

[18] SHAO Z, SHAO W, PI D. Effective constructive heuristic and iterated greedy algorithm for distributed mixed blocking permutation flow-shop scheduling problem [J]. Knowledge-Based Syst. 2021;221:106959.

[19] Wang S, Wang X, Chu F, et al. An energy-efficient two-stage hybrid flow shop scheduling problem in a glass production[J]. Int J P Res. 2020;58(8):2283–2314.

[20] Zheng J, Wang L, Wang J. A cooperative coevolution algorithm for multi-objective fuzzy distributed hybrid flow shop[J]. Knowledge-Based Syst. 2020;194:105536.

[21] Zhao F, He X, Wang L. A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flow-shop problem[J]. IEEE Trans Cybern. 2020;51(11):5291–5303.

[22] Zhang C, Tan J, Peng K, et al. A discrete whale swarm algorithm for hybrid flow-shop scheduling problem with limited buffers[J]. Rob Comput Integr Manuf. 2021;68:102081.

[23] HAN Y, LI J, Sang H, et al. Discrete evolutionary multi-objective optimization for energy-efficient blocking flow shop scheduling with setup time [J]. Appl Soft Comput. 2020;93:106343.

[24] Lei C, Zhao N, Ye S, et al. Memetic algorithm for solving flexible flow-shop scheduling problems with dynamic transport waiting times[J]. Comput Ind Eng. 2020;139:105984.

[25] Babayan A, He* D. Solving the n-job 3-stage flexible flowshop scheduling problem using an agent-based approach[J]. Int J P Res. 2004;42(4):777–799.

[26] Han W, Guo F, Su X. A reinforcement learning method for a hybrid flow-shop scheduling problem[J]. Algorithms. 2019;12(11):222.

[27] LUO J, Fujimura S, El Baz D, et al. GPU based parallel genetic algorithm for solving an energy efficient dynamic flexible flow shop scheduling problem [J]. J Parallel Distrib Comput. 2019;133:244–257.

[28] Tliba K, Diallo TML, Penas O, et al. Digital twin-driven dynamic scheduling of a hybrid flow shop[J]. J Intell Manuf. 2023;34(5):2281–2306.

[29] Shi L, Guo G, Song X. Multi-agent based dynamic scheduling optimisation of the sustainable hybrid flow shop in a ubiquitous environment[J]. Int J P Res. 2021;59(2):576–597.

[30] Williams HP. Model building in mathematical programming[M]. Chichester, England: John Wiley & Sons; 2013.

[31] Bemporad A, Morari M. Control of systems integrating logic, dynamics, and constraints[J]. Automatica. 1999;35(3):407–427.

[32] Darby ML, Nikolaou M. MPC: current practice and challenges[J]. Control Eng Pract. 2012;20(4):328–342.

[33] Kianfar K, Fatemi Ghomi SMT, Oroojlooy Jadid A. Study of stochastic sequence-dependent flexible flow shop via developing a dispatching rule and a hybrid GA [J]. Eng Appl Artif Intell. 2012;25(3):494–506.

[34] Hart WE, Laird CD, Watson JP, et al. Pyomo-optimization modeling in python[M]. Berlin: Springer; 2017.

[35] Bangsow S. Tecnomatix plant simulation[M]. Cham, Switzerland: Springer International Publishing; 2020.

Appendix 1

$$A=\begin{bmatrix} 0 & 0 & \cdots & & 0 & & & 0 \\ \vdots & \vdots & & & & & & \vdots \\ 0 & 0 & \cdots & & & & & 0 \\ 0 & \cdots & \cdots & & 1 & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & & 0 & 1 & \cdots & 0 \\ \vdots & & & & & & & \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix} \quad B_1=\begin{bmatrix} -U_1(k) & 0 & \cdots & 0 & 0 & & & 0 \\ \vdots & \cdots & & & & & & \vdots \\ 0 & 0 & \cdots & -U_{|S_1|}(k) & & \cdots & & 0 \\ 0 & \cdots & \cdots & & 0 & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & & 0 & 0 & \cdots & 0 \\ \vdots & & & & & & & \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 \end{bmatrix}$$

$$B_2=\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & & & 0 \\ \vdots & \cdots & & & & & & \vdots \\ 0 & 0 & \cdots & 1 & & \cdots & & 0 \\ 0 & \cdots & \cdots & & 0 & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & & 0 & 0 & \cdots & 0 \\ \vdots & & & & & & & \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 \end{bmatrix}$$

$$B_3=\begin{bmatrix} 0 & \cdots & \cdots & & 0 & 0 & \cdots & 0 & 0 \\ \vdots & & & & & & & & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 0 \\ 1 & 0 & \cdots & 0 & \cdots & -1 & 0 & \cdots & 0 \\ 0 & \cdots & 1 & \cdots & 0 & \cdots & -1 & \cdots & 0 \\ \vdots & \vdots & & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & \cdots & 0 & \cdots & -1 \end{bmatrix}$$

$$E_1=\begin{bmatrix} -M & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ \vdots & & & & & & & \vdots \\ 0 & 0 & \cdots & \cdots & & & \cdots & -M \\ M & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & M \\ -M_b & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & -M_b \\ m_b & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & m_b \\ -m_b & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & -m_b \\ M_b & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & M_b \\ U_1(k) & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & U_{|S|}(k) \\ -U_1(k) & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & -U_{|S|}(k) \\ 0 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & 0 \end{bmatrix} \quad E_2=\begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ \vdots & & & & & & & \vdots \\ 0 & 0 & \cdots & \cdots & & & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & 0 \\ 1 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & 1 \\ -1 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & -1 \\ 1 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & 1 \\ -1 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & -1 \\ -1 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & -1 \\ 1 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & 1 \\ 0 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots & \vdots \\ 0 & 0 & \cdots & & \cdots & & \cdots & 0 \end{bmatrix}$$

$$E_3 = \begin{bmatrix}
-1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\
\vdots & & & & & & & \vdots \\
0 & 0 & \cdots & \cdots & & & \cdots & -1 \\
1 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\
\vdots & \vdots & & & & & \vdots & \vdots \\
0 & 0 & \cdots & & \cdots & & \cdots & 1 \\
0 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\
\vdots & \vdots & & & & & \vdots & \vdots \\
0 & 0 & \cdots & & \cdots & & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\
\vdots & \vdots & & & & & \vdots & \vdots \\
0 & 0 & \cdots & & \cdots & & \cdots & 0 \\
1 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\
\vdots & \vdots & & & & & \vdots & \vdots \\
0 & 0 & \cdots & & \cdots & & \cdots & 1 \\
-1 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\
\vdots & \vdots & & & & & \vdots & \vdots \\
0 & 0 & \cdots & & \cdots & & \cdots & -1 \\
-1 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\
\vdots & \vdots & & & & & \vdots & \vdots \\
0 & 0 & \cdots & & \cdots & & \cdots & -1 \\
1 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\
\vdots & \vdots & & & & & \vdots & \vdots \\
0 & 0 & \cdots & & \cdots & & \cdots & 1 \\
0 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\
\vdots & \vdots & & & & & \vdots & \vdots \\
0 & 0 & \cdots & & \cdots & & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & & \cdots & 0 \\
\vdots & \vdots & & & & & \vdots & \vdots \\
0 & 0 & \cdots & & \cdots & & \cdots & 0
\end{bmatrix}
\quad
E_4 = \begin{bmatrix}
0 & 0 & \cdots & & 0 & 0 & & \cdots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \cdots & & & \cdots & & \cdots & 0 \\
0 & 0 & \cdots & & 0 & 0 & & \cdots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \cdots & & & \cdots & & \cdots & 0 \\
0 & 0 & \cdots & & 0 & 0 & & \cdots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \cdots & & & \cdots & & \cdots & 0 \\
0 & 0 & \cdots & & 0 & 0 & & \cdots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \cdots & & & \cdots & & \cdots & 0 \\
0 & 0 & \cdots & & 0 & 0 & & \cdots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \cdots & & & \cdots & & \cdots & 0 \\
0 & 0 & \cdots & & 0 & 0 & & \cdots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \cdots & & & \cdots & & \cdots & 0 \\
0 & 0 & \cdots & 0 & 1 & \cdots & 0 & \cdots & 0 \\
\vdots & & & & & & & & \vdots \\
0 & 0 & \cdots & & \cdots & & & \cdots & 1 \\
0 & 0 & \cdots & 0 & -1 & \cdots & 0 & \cdots & 0 \\
\vdots & & & & & & & & \vdots \\
0 & 0 & \cdots & & \cdots & & & \cdots & -1 \\
-1 & -1 & \cdots & 0 & 1 & \cdots & 0 & \cdots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \cdots & -1 & 0 & \cdots & 1 & \cdots & 0 \\
1 & 1 & \cdots & 0 & -1 & \cdots & 0 & \cdots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \cdots & 1 & 0 & \cdots & -1 & \cdots & 0
\end{bmatrix}$$

$$E_5 = \begin{bmatrix}
U_1(k)-\varepsilon & 0 & \ldots & & 0 & 0 & & \ldots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \ldots & & & \ldots & & \ldots & U_{|S|}(k)-\varepsilon \\
M-U_1(k) & 0 & \ldots & & 0 & 0 & & \ldots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \ldots & & & \ldots & & \ldots & M-U_{|S|}(k) \\
0 & 0 & \ldots & & 0 & 0 & & \ldots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \ldots & & & \ldots & & \ldots & 0 \\
0 & 0 & \ldots & & 0 & 0 & & \ldots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \ldots & & & \ldots & & \ldots & 0 \\
-m_b & 0 & \ldots & & 0 & 0 & & \ldots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \ldots & & & \ldots & & \ldots & -m_b \\
M_b & 0 & \ldots & & 0 & 0 & & \ldots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \ldots & & & \ldots & & \ldots & M_b \\
0 & 0 & \ldots & 0 & 0 & \ldots & 0 & \ldots & 0 \\
\vdots & & & & & & & & \vdots \\
0 & 0 & \ldots & & \ldots & & & \ldots & 0 \\
0 & 0 & \ldots & 0 & 0 & \ldots & 0 & \ldots & 0 \\
\vdots & & & & & & & & \vdots \\
0 & 0 & \ldots & & \ldots & & & \ldots & 0 \\
0 & 0 & \ldots & 0 & 0 & \ldots & 0 & \ldots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \ldots & 0 & 0 & \ldots & 0 & \ldots & 0 \\
0 & 0 & \ldots & 0 & 0 & \ldots & 0 & \ldots & 0 \\
\vdots & \vdots & & & & & & \vdots & \vdots \\
0 & 0 & \ldots & 0 & 0 & \ldots & 0 & \ldots & 0 \\
 & & & & & & & &
\end{bmatrix}$$