processes

MDPI

Article

Petri Net Model Predictive Control Method for Batch Chemical Systems

Zexuan Lin [1,2], Jiazhong Zhou [1,2], Shasha Sun [1,2], Jiliang Luo [1,2]* and Jiabing Zhang [1,2]

[1] College of Information Science and Engineering, Huaqiao University, Xiamen 361021, China; lzx886200@163.com (Z.L.); zhoujiazhong@hqu.edu.cn (J.Z.); ss.sun@hqu.edu.cn (S.S.); jbzhang@hqu.edu.cn (J.Z.)
[2] Fujian Engineering Research Centerof Motor Control and System Optiomal Schedule, Xiamen 361021, China
* Correspondence: jlluo@hqu.edu.cn; Tel.: +86-131-1059-5996

Abstract: In order to address the problem of the real-time scheduling and control of batch chemical systems, this work proposes a model predictive control method based on Petri nets. First, a method is presented to construct a batch chemical system's timed Petri net model. Second, a control structure is designed to augment the Petri net model to control the valves. This results in timed Petri nets that formally represent the process specifications of a batch chemical system. Third, a model predictive control method is developed to schedule and control timed Petri nets, where a proposed heuristic function is utilized to perform the optimization computation. The model parameters are dynamically adjusted using online data, and both scheduling and valve control instructions are calculated in real time. Finally, a series of experiments is carried out in a beer canning plant to verify the proposed method. According to the experimental results, the scheduling and control problem can be solved in real time, where the online computations can be performed in milliseconds, and the resulting scheduling strategies are optimal or near-optimal.

Keywords: timed Petri net; batch chemical system; model predictive control; heuristic function; real-time scheduling

Citation: Lin, Z.; Zhou, J.; Sun, S.; Luo, J.; Zhang, J. Petri Net Model Predictive Control Method for Batch Chemical Systems. *Processes* **2024**, *12*, 620. https://doi.org/10.3390/pr12030620

Academic Editor: Dan Zhang

Received: 13 February 2024
Revised: 10 March 2024
Accepted: 19 March 2024
Published: 21 March 2024

1. Introduction

Under the influence of the Internet of Things and intelligent manufacturing technologies, the resource units and task numbers in batch chemical systems are increasing. The logical relationships among processes in competition for resource units are becoming increasingly complex, leading to a combinatorial explosion of system states. Simultaneously, the system must deal with unforeseen events such as urgent orders, process adjustments, and equipment failures. This necessitates rapid real-time solutions for scheduling strategies and control commands, posing highly complex scheduling and control problems.

Batch chemical systems are driven by events such as the opening or closing of valves and the initiation and completion of operations, making them a typical group of discrete event systems. Petri nets are an excellent tool for discrete event systems [1,2] and are widely used in research on fault diagnosis, supervisory control, and optimization scheduling [3,4].

Batch chemical systems involve complex scheduling strategies and specific valve control commands translated from scheduling strategies [5–14]. Since scheduling and control commands require real-time computation and execution, responding quickly to unforeseen events, such as emergency orders and equipment failures, becomes challenging.

Regarding real-time scheduling, modeling via Petri nets and the use of model predictive control (MPC) are possible [15–17]. However, they fail to address the mismatch between the system model and the physical plant, and the repeated expansion of the reachability tree leads to unnecessary computation.

To overcome these drawbacks, this work presents an MPC method using place-timed Petri nets, which not only addresses the mismatch between the Petri net model

and the actual plant but also fulfills the real-time scheduling requirement. It is summarized as follows.

(1) For batch chemical systems, a systematic method is proposed to obtain its place-timed Petri net model by analyzing the relationship between the process specifications and logical operations.
(2) A heuristic function is designed based on the place-timed Petri net mode to estimate states' time costs, which can be used to smartly adjust the searching direction within a reachability graph to reduce the computation required.
(3) A predictive control algorithm is designed to search for an optimal or near-optimal control action (transition) under the guidance of a heuristic function within each prediction time domain (time step). It has a model correction module that collects data online about operational times and device failures, and we can dynamically modify the Petri net model accordingly.
(4) We conduct experiments on a beer canning plant. The online computations of control actions are completed within milliseconds, and the makespan is close to the optimal one. The results show that the proposed method provides an efficient and practical solution for the real-time scheduling of a batch chemical plant.

Compared to the approaches in [15–17], our approach addresses the issue of the mismatch between the Petri net model and the actual plant. The model calibration is performed in real-time according to abrupt events, such as rush orders and device failures. Furthermore, we retain a portion of the reachability graph of the previous time step, and the expanded part of the graph is saved in each predictive time domain. This enables us to explore more states and thereby enhance the scheduling performance.

The rest of this paper consists of the following sections. Section 1 presents some related works and describes contributions concerning the existing methods. Section 2 reviews basic definitions and notions regarding Petri nets and MPC. Section 3 describes the optimization issue of batch chemical systems. Section 4 proposes the method to model batch chemical systems using place-timed Petri nets. Section 5 presents an MPC method to schedule a batch chemical system based on Petri nets and the proposed heuristic function. Section 6 describes a series of experiments to verify the proposed method. Section 7 provides the conclusions of the work.

2. Related Works

The control and scheduling problems of batch chemical systems have received widespread attention, leading to valuable research efforts. Tittus and Lennartson et al. [5] propose a hierarchical Petri net control method for batch chemical systems that can prevent deadlock states, allocate material transport equipment and reactors reasonably, and execute multiple process flows in parallel. Wang et al. [6,7] introduce a control approach based on hierarchical Petri nets for batch chemical systems, capable of generating control instructions for the production and transportation of multiple materials, thereby increasing the system's stability. Susumu et al. [8] develop a Petri net model for a batch chemical production process and propose a method to design reliable operation instructions. Falkman et al. [9] introduce a process algebra Petri net, which is helpful in accurately describing the complex request relationships between operations and resources. Ghaeli et al. [10] establish a place-timed Petri net model for batch factories and present a branching and bounding algorithm to compute optimal scheduling strategies. Lai et al. [11] model a pipeline network of a batch chemical plant as a Petri net and use integer programming to determine the shortest material transfer paths. With max-plus algebra, Weyerman et al. [12] construct a scheduling optimization model of batch flow shops to find optimal processing sequences. Given a batch chemical system, Lin et al. [13] design a Petri-net-based A* algorithm to obtain the optimal paths within a reachable graph and map the paths to valve operational instructions. Given a place-timed Petri net, Zhou et al. [14] propose a state space approximation method to reduce the computational complexity in searching for optimal or suboptimal scheduling

strategies. These works show promising progress in scheduling batch chemical systems. However, they fail to compute scheduling strategies real time.

Model predictive control (MPC) is an important type of advanced control approach that can utilize system information through well-developed models and real-time process measurements to predict the future trajectories of processes. Cahyono et al. [18] propose a model predictive allocation method that integrates berth and dockside crane models to enhance the efficiency during the berthing process. However, in the rolling optimization part, an exhaustive search algorithm is employed, resulting in time-consuming computation. Lefebvre et al. [15] introduce a Petri-net-based model predictive control method that can be used to calculate the shortest or approximate shortest transition firing sequences to reach target states. Lefebvre et al. [16,17] extend the method in [15] to a timed Petri net such that approximate shortest-duration scheduling strategies can be calculated even if there exist uncontrollable transitions. These works have achieved good results in real-time scheduling using MPC. However, they do not consider abrupt events, which may cause the MPC methods to fail.

3. Preliminaries

An ordinary Petri net is $N = (P, T, F, W)$, where P is a finite non-empty set of places and T is a finite non-empty set of transitions, where $P \cup T \neq \emptyset \wedge P \cap T = \emptyset$. $F \subseteq (P \times T) \cup (T \times P)$ is the set of directed arcs connecting the nodes, i.e., the directed arcs from the place to the transition or from the transition to the place. $W : F \to N^+$ denotes the weights mapped to the directed arcs. $[N]$ denotes the $|P| \times |T|$ integer matrix such that

$$[N](p,t) = \begin{cases} W(t,p), & \text{if } (t,p) \in F \wedge (p,t) \notin F \\ -W(p,pt), & \text{if } (t,p) \notin F \wedge (p,t) \in F \\ W(t,p) - W(p,t), & \text{if } (t,p) \in F \wedge (p,t) \in F \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Given that $x \in P \cup T$ is a node in a Petri net, define the x forward set as ${}^\bullet x = \{y \in P \cup T | (y,x) \in F\}$; define the x backward set as $x^\bullet = \{y \in P \cup T | (x,y) \in F\}$. This representation can be extended to a set of nodes, i.e., given that $X \subseteq P \cup T$, ${}^\bullet X = \cup_{x \in X} {}^\bullet x$ and $X^\bullet = \cup_{x \in X} x^\bullet$.

A marking m is a vector indicating a distribution of tokens among places. Given a Petri net, a transition t is state-enabled at m, if $m(p) \geq W(p,t) \wedge \forall p \in {}^\bullet t$, i.e., $m[t\rangle$. If transition t can fire, the net reaches a new marking m' such that

$$\forall p \in P, m'(p) = m(p) + [N](p,t), \tag{2}$$

and this is denoted as $m[t\rangle m'$. If there exists a transition sequence $\sigma_= e_1 e_2 ... e_n$ and marking $m_1, m2, \ldots, m_n$ such that $m_0[e_1\rangle m_1[e_2\rangle m_2 \ldots m_{n_1}[e_n\rangle m_n$ holds, then the marking m_m is reachable from marking m_0, denoted as $m_0[\sigma\rangle m_n$.

A place-timed Petri net (P-TPN) is defined as $G_t = (N, m_0, d)$, where N is an ordinary Petri net structure, m_0 is the initial marking, and $d : P \to \{0\} \cup R^+$ is the function mapping the set of places to the set R^+ of non-negative real numbers, where $d(p)$ denotes the delay of the place. A place-timed Petri net is used for system modeling, and the delay of the place corresponding to a time-consuming task is not zero. In this paper, the number of tokens in the place is not greater than one.

The state of a place-timed Petri net is $X = (m, w, g)$, where m is the marking of state X, g is the cost function of the system, and $w : P' \to \{0\} \cup R^+$ is a function in which the set P' of places has a token for the set R^+, specifying the waiting time of the token in the place. If there exists a transition sequence σ such that $m_0[\sigma\rangle m_n$, where m_0 and m_n are the markings of X_0 and X_n, respectively, then the state X_n is reachable from marking X_0, denoted as $X_0[\sigma\rangle X_n$.

Given the state $X_K = (m_k, w_k, g_k)$ of a place-timed Petri net at the k-th moment, if the transition e_k is enabled, we update the marking m_{k+1}, the waiting time w_{k+1}, and the cost time g_{k+1} according to Equations (2)–(4).

$$\forall p \in P, w_{k+1} = \begin{cases} \min(w_{k+1} + \max(d(p), w_k), d(p)), & \text{if } m_{k+1}(p) \neq 0 \wedge p \notin e_k^\bullet \\ 0, & \text{if } m_{k+1}(p) = 0 \vee p \in e_k^\bullet \end{cases} \quad (3)$$

$$g_{k+1} = g_k + \lambda_{k+1}, \quad (4)$$

where λ_{k+1} is the time interval between X_k and X_{k+1}, i.e., $\lambda_{k+1} \geq max(d(p) - w_k(p))$.

Dijkstra and A^* algorithms are frequently used as tree and graph searching methods to obtain an optimal solution. The former selects the node with the smallest cost to explore a state graph and can guarantee that its solution is an optimal path. The latter utilizes the sum of the cost and heuristic function to adjust the searching direction and outperforms the former in terms of computational efficiency if its heuristic function is appropriate.

Model predictive control (MPC) is an optimal control method that uses a dynamic model to predict future behavior and solves for the optimal control action at each time step to satisfy constraints and performance metrics [19]. It is commonly used in various dynamic systems, including industrial process control, robot control, etc. MPC dynamically adjusts the control strategy through real-time sensor feedback to match the system behavior more accurately and achieve stability and performance optimization.

4. Problem Description

A batch chemical system carries out a series of operations, each requiring a certain amount of time and a set of resources, such as valves, storage tanks, and filters. Additionally, it may encounter unexpected events such as process adjustments, urgent orders, and equipment failures. Therefore, control instructions for valves need to be calculated in real time to adjust the order of operations, ensuring the system's productive efficiency.

To describe a batch chemical system, let D represent the set of resources and O the set of all production operations. Specifically, D has two subsets V and U. Set $V = \{v_1, \ldots, v_i\}$ is the set of valves, where each valve v_i has two states, i.e., the open state v_{s_i} and the closed state $\bar{v}_{s_i}$. Set $U = \{u_1, \ldots, u_j, f_1, \ldots, f_z\}$ is the set of tanks and filters, where $j, z \in N^+$. $u_1, \ldots, u_j$ are buffer tanks, and $f_1, \ldots, f_z$ are filters. Note that, for each tank or filter, $\bar{u}_{s_j}(\bar{f}_{s_z})$, i.e., the tank is full, and the filter is used. Therefore, the set of resource states is denoted as $D_s = \{v_{s_1}, ..., v_{s_i}, \bar{v}_{s_1}, ..., \bar{v}_{s_i}, u_{s_1}, ..., u_{s_j}, \bar{u}_{s_1}, ..., \bar{u}_{s_j}, f_{s_1}, ..., f_{s_z}, \bar{f}_{s_1}, ..., \bar{f}_{s_z}\}$. For an operation $o \in O$, its required set of resource states is represented as $r(o) \subseteq D_s$, and $\delta(o) : o \to N^+$ is the execution time of o.

As shown in Figure 1, a beer canning plant [20] is considered. The plant performs four tasks: filling, filtering, bottling, and cleaning. u_s is the tank for raw beer; u_1 and u_2 are two buffer tanks. f_1, f_2, and f_3 are three filters. CIP stands for the supply and recovery module for cleaning agents, while the bottling module is responsible for canning. $v_1, \ldots, v_{20}$ are double piston valves. Each one can be switched to either an "open" or "closed" position. When a valve is open, beer flows through the horizontal pipeline under its control. When a valve is closed, beer flows through the vertical pipeline under its control. There are 12 operations, denoted by $o_1, \ldots, o_{12}$, as summarized in Table 1.

A filling operation seeks to transport raw beer from u_s through the pipeline to the buffer tank u_1. There are two types of filling operations, o_1 and o_2. Operation o_1 requires the opening of valves v_2 and v_3; the closing of valves $\bar{v}_1$, $\bar{v}_8$, $\bar{v}_9$, and $\bar{v}_{12}$; and an execution time of five seconds. The set of resource states required by o_1 is $r(o_1) = \{v_{s_2}, v_{s_3}, \bar{v}_{s_1}, \bar{v}_{s_8}, \bar{v}_{s_9}, \bar{v}_{s_{12}}, u_{s_1}\}$, and the execution time of o_2 is $\delta(o_2) = 6$ s.

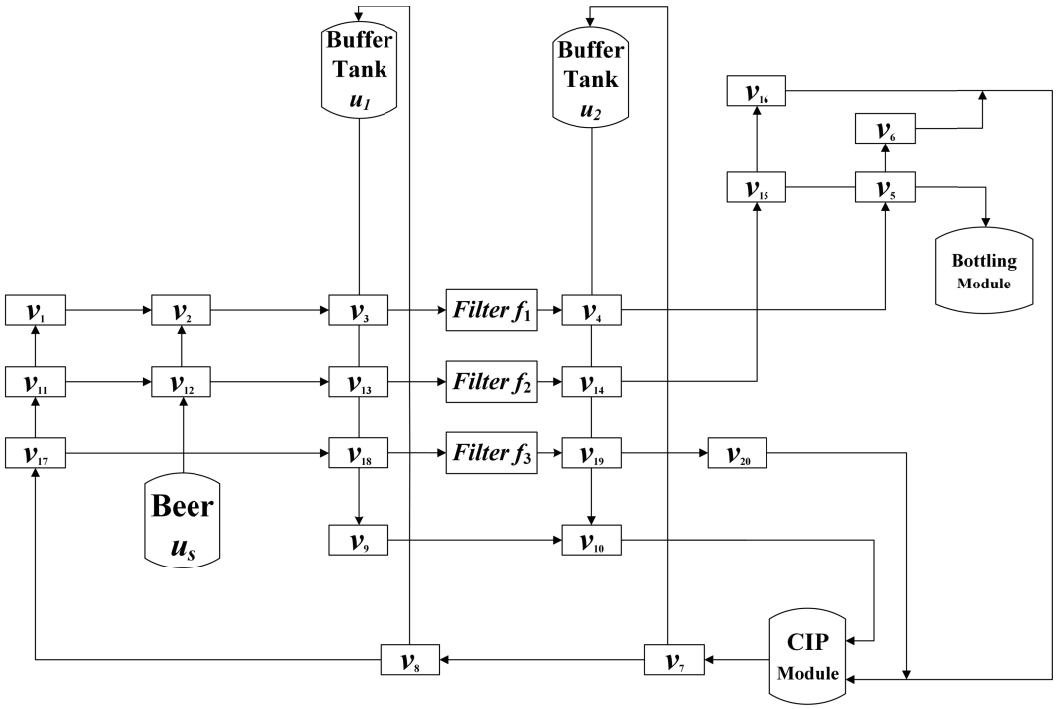

Figure 1. A beer canning plant.

Table 1. Operations of the beer canning plant.

Operations	Open Valve States	Closed Valve States	Tank or Filter States	Time
o_1	v_{s_2}, v_{s_3}	$\bar{v}_{s_1}, \bar{v}_{s_8}, \bar{v}_{s_9}, \bar{v}_{s_{12}}$	u_{s_1}	5 s
o_2	$v_{s_{12}}, v_{s_{13}}$	$\bar{v}_{s_3}, \bar{v}_{s_8}, \bar{v}_{s_9}, \bar{v}_{s_{11}}$	u_{s_1}	6 s
o_3	v_{s_3}, v_{s_4}	$\bar{v}_{s_1}, \bar{v}_{s_2}, \bar{v}_{s_8}, \bar{v}_{s_9}, \bar{v}_{s_7}, \bar{v}_{s_{10}}$	$\bar{u}_{s_1}, u_{s_2}, f_{s_1}$	11 s
o_4	$v_{s_{13}}, v_{s_{14}}$	$\bar{v}_{s_3}, \bar{v}_{s_4}, \bar{v}_{s_8}, \bar{v}_{s_9}, \bar{v}_{s_7}, \bar{v}_{s_{10}}, \bar{v}_{s_{11}}, \bar{v}_{s_{12}}$	$\bar{u}_{s_1}, u_{s_2}, f_{s_2}$	12 s
o_5	$v_{s_{18}}, v_{s_{19}}$	$\bar{v}_{s_3}, \bar{v}_{s_4}, \bar{v}_{s_8}, \bar{v}_{s_9}, \bar{v}_{s_7}, \bar{v}_{s_{10}}, \bar{v}_{s_{13}}, \bar{v}_{s_{14}}, \bar{v}_{s_{17}}$	$\bar{u}_{s_1}, u_{s_2}, f_{s_3}$	13 s
o_6	v_{s_4}, v_{s_5}	$\bar{v}_{s_1}, \bar{v}_{s_6}, \bar{v}_{s_7}, \bar{v}_{s_{10}}$	$\bar{u}_{s_2}$	7 s
o_7	$v_{s_{14}}, v_{s_{15}}$	$\bar{v}_{s_5}, \bar{v}_{s_4}, \bar{v}_{s_7}, \bar{v}_{s_{10}}, \bar{v}_{s_{11}}, \bar{v}_{s_{16}}$	$\bar{u}_{s_2}$	8 s
o_8	v_{s_8}, v_{s_9}	$\bar{v}_{s_3}, \bar{v}_{s_7}, \bar{v}_{s_{10}}, \bar{v}_{s_{13}}, \bar{v}_{s_{18}}$	u_{s_1}	2 min
o_9	$v_{s_7}, v_{s_{10}}$	$\bar{v}_{s_4}, \bar{v}_{s_{14}}, \bar{v}_{s_{19}}$	u_{s_2}	2 min
o_{10}	v_{s_1}, v_{s_6}	$\bar{v}_{s_2}, \bar{v}_{s_3}, \bar{v}_{s_4}, \bar{v}_{s_5}, \bar{v}_{s_7}, \bar{v}_{s_8}, \bar{v}_{s_{11}}, \bar{v}_{s_{17}}$	$\bar{f}_{s_1}$	2 min
o_{11}	$v_{s_{11}}, v_{s_{16}}$	$\bar{v}_{s_7}, \bar{v}_{s_8}, \bar{v}_{s_{12}}, \bar{v}_{s_{13}}, \bar{v}_{s_{14}}, \bar{v}_{s_{15}}, \bar{v}_{s_{17}}$	$\bar{f}_{s_2}$	2 min
o_{12}	$v_{s_{17}}, v_{s_{20}}$	$\bar{v}_{s_7}, \bar{v}_{s_8}, \bar{v}_{s_{18}}, \bar{v}_{s_{19}}$	$\bar{f}_{s_3}$	2 min

A filtering operation serves to transfer liquid from buffer tank u_1, through filters (f_1, f_2, f_3), to buffer tank u_2. There are three types of filtering operations, o_3, o_4, and o_5. Operation o_3 requires the opening of valves v_3 and v_4; the closing of valves $\bar{v}_{s_1}, \bar{v}_{s_2}, \bar{v}_{s_8}, \bar{v}_{s_9}, \bar{v}_{s_7}$, and $\bar{v}_{s_{10}}$; and the execution time 11 s. Hence, $r(o_3) = \{v_{s_3}, v_{s_4}, \bar{v}_{s_1}, \bar{v}_{s_2}, \bar{v}_{s_8}, \bar{v}_{s_7}, \bar{v}_{s_{10}}, \bar{u}_{s_1}, u_{s_2}, f_{s_1}\}$, and $\delta(o_3) = 11$ s. Similarly, $r(o_4) = \{v_{s_{13}}, v_{s_{14}}, \bar{v}_{s_3}, \bar{v}_{s_4}, \bar{v}_{s_7}, \bar{v}_{s_8}, \bar{v}_{s_9}, \bar{v}_{s_{10}}, \bar{v}_{s_{11}}, \bar{v}_{s_{12}}, \bar{u}_{s_1}, u_{s_2}, f_{s_2}\}$, $\delta(o_4) = 12$ s, $r(o_5) = \{v_{s_{18}}, v_{s_{19}}, \bar{v}_{s_3}, \bar{v}_{s_4}, \bar{v}_{s_7}, \bar{v}_{s_8}, \bar{v}_{s_9}, \bar{v}_{s_{10}}, \bar{v}_{s_{13}}, \bar{v}_{s_{14}}, \bar{v}_{s_{17}}, \bar{u}_{s_1}, u_{s_2}, f_{s_3}\}$, and $\delta(o_5) = 13$ s.

A bottling operation transfers liquid from buffer tank u_2 to the bottling module for beer bottling. There are two types of bottling operations, o_6 and o_7. In detail, $r(o_6) = \{v_{s_4}, v_{s_5}, \bar{v}_{s_1}, \bar{v}_{s_6}, \bar{v}_{s_7}, \bar{v}_{s_{10}}, \bar{u}_{s_2}\}$, $\delta(o_6) = 7$ s, $r(o_7) = \{v_{s_{14}}, v_{s_{15}}, \bar{v}_{s_5}, \bar{v}_{s_4}, \bar{v}_{s_7}, \bar{v}_{s_{10}}, \bar{v}_{s_{11}}, \bar{v}_{s_{16}}, \bar{u}_{s_2}\}$, and $\delta(o_7) = 8$ s.

A cleaning operation cleans the resources in U, and there are five cleaning operations:

(1) Operation o_8 (cleaning buffer tank u_1): It is required that the cleaning liquid flows from CIP into buffer tank u_1 for 120 s. The set of required resources is $r(o_8) = \{v_{s_8}, v_{s_9}, \bar{v}_{s_3}, \bar{v}_{s_7}, \bar{v}_{s_7}, \bar{v}_{s_{10}}, \bar{v}_{s_{13}}, \bar{v}_{s_{18}}, u_{s_1}\}$, and the cleaning time is $\delta(o_8) = 120$ s. The buffer tank u_1 must be cleaned after every two uses.

(2) Operation o_9 (cleaning buffer tank u_2): It is required that the cleaning liquid flows from CIP into buffer tank u_2 for 120 s. The set of required resources is

$r(o_9) = \{v_{s_7}, v_{s_{10}}, \bar{v}_{s_4}, \bar{v}_{s_{14}}, \bar{v}_{s_{19}}, \bar{u}_{s_2}\}$, and the cleaning time is $\delta(o_9) = 120$ s. The buffer tank u_2 must be cleaned after every two uses.

(3) Operation o_{10} (cleaning filter f_1): It is required that the cleaning liquid flows from CIP into filter f_1 for 120 s. The set of required resources is $r(o_{10}) = \{v_{s_1}, v_{s_6}, \bar{v}_{s_2}, \bar{v}_{s_3}, \bar{v}_{s_4}, \bar{v}_{s_5}, \bar{v}_{s_7}, \bar{v}_{s_8}, \bar{v}_{s_{11}}, \bar{v}_{s_{17}}, \bar{f}_{s_1}\}$, and the cleaning time is $\delta(o_{10}) = 120$ s. The filter f_1 must be cleaned after every use.

(4) Operation o_{11} (cleaning filter f_2): It is required that the cleaning liquid flows from CIP into filter f_2 for 120 s. The set of required resources is $r(o_{11}) = \{v_{s_{11}}, v_{s_{16}}, \bar{v}_{s_7}, \bar{v}_{s_8}, \bar{v}_{s_{12}}, \bar{v}_{s_{13}}, \bar{v}_{s_{14}}, \bar{v}_{s_{15}}, \bar{v}_{s_{17}}, \bar{f}_{s_2}\}$, and the cleaning time is $\delta(o_{11}) = 120$ s. The filter f_2 must be cleaned after every two uses.

(5) Operation o_{12} (cleaning filter f_3): It is required that the cleaning liquid flows from CIP into filter f_3 for 120 s. The set of required resources is $r(o_{12}) = \{v_{s_{17}}, v_{s_{20}}, \bar{v}_{s_7}, \bar{v}_{s_8}, \bar{v}_{s_{18}}, \bar{v}_{s_{19}}, \bar{f}_{s_3}\}$, and the cleaning time is $\delta(o_{12}) = 120$ s. The filter f_3 must be cleaned after every three uses.

Evidently, an operation cannot be performed simultaneously with another one once there are resource conflicts. For example, o_1 and o_2 cannot be simultaneously performed since o_1 requires the opening of v_3 while operation o_2 requires the closing of v_3. However, some operations can run in parallel. For instance, o_1 and o_6 can be simultaneously executed since they are not in conflict. Therefore, there are both sequential and parallel relations between operations, and different sequences of executed operations may lead to different makespans. In addition, when scheduling a plant, unexpected events need to be considered. For example, an urgent order may suddenly arrive when the plant is running at a given schedule, the duration of an operation may change due to worn devices, or a filter or tank device may be suddenly damaged. Thus, it is essential to quickly modify the original schedule and to respond promptly to abrupt events. The key lies in quickly determining the scheduling strategies and controlling the commands to minimize the processing time.

5. Petri Net Model of a Batch Chemical System

In order to develop a model predictive control method, we need to model a given batch chemical system as a place-timed Petri net. To this end, several notations are introduced.

Definition 1. *A logical operation ρ is a binary tuple (γ, k) that satisfies*

$$\sum_{o \in O_\rho} \gamma(o) = k, \tag{5}$$

where O_ρ is the set of productive operations, $\gamma(o)$ represents the number of times to execute o, and k is a non-negative integer.

In a real-life plant, a logical operation means a specific working procedure on a part. As shown in Figure 1, a filling operation can be realized by o_1 or o_2, and, consequently, it can be represented as the logical operation $\rho_1 = (\gamma_1, 1)$ satisfying $\sum_{o \in O_\rho} \gamma_1(o) = 1$, i.e., $\gamma_1(o_1) + \gamma_1(o_2) = 1$.

Given a batch chemical system J, the batch size, i.e., the number of times to execute the system J, is denoted by $\Omega(J)$.

Definition 2. *An ordered pair of logical operations (ρ_i, ρ_j) is called a process relation, ρ_i is a pre-operation of ρ_j, and ρ_j is a post-operation of ρ_i.*

As shown in Figure 1, ρ_1 is the logical operation of the filling operation, ρ_2 is the logical operation of the filtering operation, and the processing flow is filling and then filtering, so ρ_1 is a pre-operation of ρ_2 and ρ_2 is a post-operation of ρ_1.

Definition 3. *Operations o and o' are in conflict if o requires a valve to open and o' requires a valve to be closed, or they require different states of the same device, such as a tank or filter, i.e.,* $[\exists v \in V, v \in r(o) \wedge \bar{v} \in r(o')] \vee [\exists x \in U, x \in r(o) \wedge x \in r(o')]$.

Definition 4. *A set of operations is in conflict if each operation in it is in conflict with all others.*

Definition 5. *A set of operations is maximally in conflict, denoted by O_{max}, if it is conflict and any superset of it is not in conflict, and the set of maximally conflicting sets is represented as* $\mathbf{O_{max}}$.

Definition 6. *A process graph is a binary tuple (ζ, ψ), where ζ is the set of all logical operations and ψ is the set of all process relations.*

Given a batch chemical system J, its place-timed Petri net model is $G_{ts} = (P^J \cup P^{\bar{J}} \cup P^O \cup P^b \cup P^m, T, F, W, d, m_0)$, where P^J is the finite non-empty set of start places, $P^{\bar{J}}$ is the finite non-empty set of end places, P^O is the finite non-empty set of operation places, P^b is the finite non-empty set of buffer operation places, and P^m is the finite non-empty set of monitor places. It can be obtained by manipulating data on the device resources, process recipes, and so on.

Furthermore, the Petri net modeling algorithm for batch chemical systems plays a crucial role in the model adjustment module of our MPC method, which can be used to reconstruct the Petri net model automatically when abrupt events occur.

For a batch chemical system, Algorithm 1 provides a modeling method to automatically generate the place-timed Petri net from a given group of data about resources such as valves, tanks and filters, batch sizes, operations, and recipes. Step 1 is to initialize the elements of the place-timed Petri net. Steps 2–14 make up a loop where the logical operation variable is the cyclic variable and, for the logical operation of process relations, they design the places, transitions, and arcs. Steps 20–26 make up the loop, where the maximally conflicting set of operations O_{max} is the cyclic variable, and they design a monitor place for each $O_{max} \in \mathbf{O_{max}}$. By means of the monitor places, the number of marked places, which represent the operations in a maximally conflicting set of operations, is, at most, one; consequently, two operations cannot be simultaneously executed if they are in conflict.

By Algorithm 1, we obtain the place-timed Petri net model of the beer canning plant, as shown in Figure 2. Here, we show the procedures to generate the Petri net. Through the production operations in Definition 1 and Table 1, we refine 15 logical operations and store them in ζ. According to Definition 2, we have 12 pairs of process relations and store them in ψ. According to Definition 6, we group ζ and ψ into the process graph (ζ, ψ). According to Definitions 3–5 and Table 1, we obtain nine maximally conflicting sets. As a result, we obtain the inputs of Algorithm 1, which are the set of operations O, the operation time δ, the set of maximally conflicting sets $\mathbf{O_{max}}$, the process graph (ζ, ψ), and batch size $\Omega(J)$. According to Steps 1–14, the operation places, buffer places, transitions, arcs, arc weights, and initial markings of places are designed. Steps 14–17 aim to obtain the batch-head places of jobs and to design transitions connecting these batch starting places with the operation places. Similarly, Steps 17–20 aim to design the batch ending places of jobs. Finally, Steps 20–26 aim to design the monitor places for the maximally conflicting sets such that no operation competes for resources with any other operation.

Algorithm 1: Petri net modeling of a batch chemical system

Input: The batch size $\Omega(J)$, the set of operations O, the process graph (ζ, ψ), the set of maximally conflicting sets $O_{\mathbf{max}}$, and the set of operation times δ

Output: A place-timed Petri net G_{ts}

Initialization $P = \emptyset, T = \emptyset, F = \emptyset, W = \emptyset$;;

for $\rho_i = (\gamma_i, k_i) \in \zeta$ **do**

  for $\rho_j = (\gamma_j, k_j) \in \zeta \wedge \rho_j \neq \rho_i$ **do**

    if $(\rho_i, \rho_j) \in \psi$ **then**

      Design a buffer place $p_{i,j}$ for the process relation (ρ_i, ρ_j), i.e., $P^b \leftarrow P^b \cup \{p_{i,j}\}$;

    for $o_x \in O_{\rho_i}$ **do**

      $P^o \leftarrow P^o \cup \{p_{o_x}\}, T \leftarrow T \cup \{t_{\bar{o}_x}\}, F \leftarrow F \cup \{(p_{o_x}, t_{\bar{o}_x}), (t_{\bar{o}_x}, p_{o_{i,j}})\}, d(p_{o_x}) = \delta(o_x)$ and $m_0(p_{o_x}) = 0$;

    for $o_y \in O_{\rho_j}$ **do**

      $P^o \leftarrow P^o \cup \{p_{o_y}\}, T \leftarrow T \cup \{t_{\bar{o}_y}\}, F \leftarrow F \cup \{(p_{o_{i,j}}, t_{o_y}), (t_{o_y}, p_{o_y})\}, d(p_{o_y}) = \delta(o_y)$ and $m_0(p_{o_y}) = 0$;

    if $\frac{k_i}{k_j} \geq 1$, *and* $\rho_i = (\gamma_i, k_i) \wedge \rho_j = (\gamma_j, k_j)$ **then**

      $W(p_{o_{i,j}}, t_{o_j}) = \frac{k_i}{k_j}, W(p_{o_i}, t_{\bar{o}_i}) = W(t_{\bar{o}_i}, p_{o_{i,j}}) = W(t_{\bar{o}_j}, p_{o_j}) = 1, m_0(p_{o_{i,j}}) = 0$ and $o_x \in O_{\rho_i} \wedge o_y \in O_{\rho_j}$;;

    else

      $m_0(p_{o_{i,j}}) = W(t_{\bar{o}_i}, p_{o_{i,j}}) = \frac{k_j}{k_i}, W(p_{o_i}, t_{\bar{o}_i}) = W(p_{o_{i,j}}, t_{o_j}) = W(t_{o_j}, p_{o_j}) = 1, m_0(p_{o_{i,j}}) = 0$ and $o_x \in O_{\rho_i} \wedge o_y \in O_{\rho_j}$;

$P^J \leftarrow P^J \cup \{p_J\}, d(p_J) = 0$ and $m_0(p_J) = \Omega(J)$;;

for $o_j \in O_{\rho_s}$ *where* ρ_s *is the first logical operation from* J **do**

  $T \leftarrow T \cup \{t_{o_j}\}$ and $F \leftarrow F \cup \{(p_J, t_{o_j}), (t_{o_j}, p_{o_j})\}$;

$P^{\bar{J}} \leftarrow P^{\bar{J}} \cup \{p_{\bar{J}}\}, d(p_{\bar{J}}) = 0, m_0(p_{\bar{J}}) = 0$;;

for $\rho_e \in \zeta$ *where* ρ_e *is the last logical operation from* J **do**

  $T \leftarrow T \cup \{t_{\bar{o}_e}\}, F \leftarrow F \cup \{(p_{o_e}, t_{\bar{o}_e}), (t_{\bar{o}_e}, p_{\bar{J}})\}$;

for $O_{max} \in \mathbf{O_{max}}$ **do**

  Design the monitor places p_c with $m_0(p_c) = 1$, i.e., $P^m \leftarrow P^m \cup \{p_c\}$;

  for $o_i, o_j \in O_{max} \wedge o_j \neq o_i$ **do**

    if $(\rho_x, \rho_y) \in \psi, o_i \in O_{\rho_x}$ *and* $o_j \in O_{\rho_y}$ **then**

      The arcs (p_c, t_{o_i}) and $(t_{\bar{o}_j}, p_c)$ are designed;

    else

      The arcs $(p_c, t_{o_i}), (t_{\bar{o}_i}, p_c), (p_c, t_{o_j})$, and $(t_{\bar{o}_j}, p_c)$ are designed;

return A place-timed Petri net G_{ts}**;**

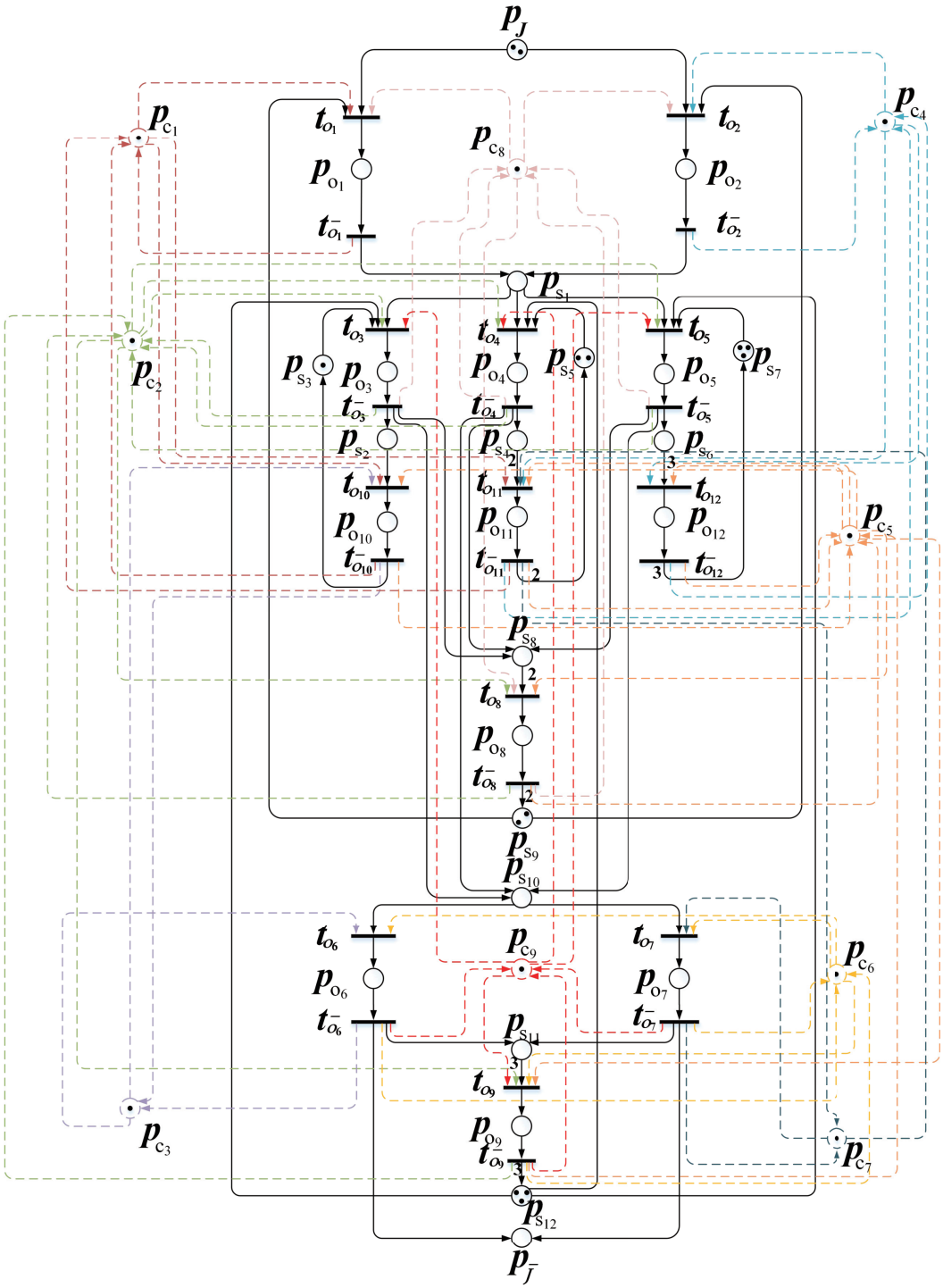

Figure 2. The place-timed Petri net model of the beer canning plant.

6. Petri-Net-Based Predictive Control for Batch Chemical Systems

On the basis of the place-timed Petri net, a model predictive control method is presented to schedule and control a batch chemical plant in real time. It consists of three parts: the prediction model, rolling optimization, and feedback correction.

The model prediction is responsible for determining an appropriate event (transition) to be executed at each current time. This involves evaluating all candidate transitions by means of the evolving trajectories within the given prediction time domain. To do so, we

need to construct a heuristic function to evaluate the execution time from any state to a goal one.

In a Petri net, an elementary path is an ordered sequence of nodes, i.e., $\pi = x_1 x_2 ... x_K$, where x_{k+1} is an output node of x_k for $k = 1, ..., K-1$. The beginning and terminal nodes of π are denoted by ${}^\bullet\pi$ and $\pi^\bullet$, respectively. The delay cost of an elementary path π is $\tau(\pi) = \sum_{p \in P_\pi} d(p)$, where P_π denotes the places occurring in π and $d(p)$ is the delay of place p.

Definition 7. *For an operation place $p \in P^o$, an elementary path is called its downstream path if ${}^\bullet\pi = p$ and $\pi^\bullet = p_{\bar{J}}$, where $p_{\bar{J}}$ is a place representing the complete logical operation of a system, and the set of its downstream paths is denoted by Π_p.*

Since the completion of jobs is described as the flow of tokens into certain places, we can define the time cost that tokens require by means of the downstream path. Consequently, we obtain a method to evaluate a Petri net's states and search for them within or near the transition trajectories that require minimal time.

The path cost function is defined as follows:

$$h_k(m_k) = \max_{p \in P^o} (\min_{\pi \in \Pi_p} \tau(\pi) \cdot m_k(p)), \tag{6}$$

where m_k is the marking in a state X_k.

According to Equation (6), the path cost can be used to estimate the time taken to complete all jobs and, consequently, to provide an approach to the optimization problem in the proposed Petri-net-based MPC.

A place-timed Petri net plays the role of the prediction model and is used to explore a sub-reachability graph, where a current state $X_k = (m_k, w_k, g_k)$ is taken as the root node, within a given prediction time domain H, where $H \to N^+$. By evaluating the leaf nodes of the sub-reachability graph with $g_k(m_k) + h_k(m_k)$, the leaf node with the minimal value is chosen since it is most possible in the optimal trajectory. By backtracking the sub-reachability graph from the leaf node, we can find the transition e_{k+1} that can be fired at the current state X_k and consequently take it as the control action at the current state.

Sensors that detect the aging of the equipment can lead to variations in operating process times, and devices occasionally break down in practice. The mismatch between an original Petri net model and a real plant is inevitable and leads to a decrease in the MPC's reliability. To tackle this issue, we introduce a feedback correction mechanism in the proposed Petri-net-based MPC method.

Definition 8. *Given the k-th firing transition e_k, its workspace is the set of operations whose ending events are modeled by e_k in the Petri net model, which is denoted by $O(e_k)$.*

Definition 9. *Given the k-th firing transition e_k, the feedback time α_k is the function mapping from its workspace to the real number set, where $\forall o \in O(e_k)$, $\alpha_k(o)$ is the difference in the actual ending and beginning times of o.*

As a plant runs, events are executed one by one and are modeled by transitions in its Petri net model. Once a transition fires, we can calculate the real processing times of the operations that have been completed. According to Definitions 8 and 9, the real processing times are represented by α_k and are fed back into the MPC such that the delays of the corresponding places can be corrected.

In a real plant, there are devices that are prone to breakdown. If a device is damaged suddenly, the operations relying on it cannot be performed. Hence, we need to re-model the plant.

Definition 10. *At the k-th moment, the fault-event* $\eta_k : D \to \{0,1\}$ *is the function mapping from a resource set to* $True(1)$ *or* $False(0)$*, and* $\forall \varpi \in D$*, if* ϖ *breaks down,* $\eta_k(\varpi) = 1$*; otherwise,* $\eta_k(\varpi) = 0$*.*

According to Definition 10, we can identify the damaged devices by the fault event, determine the operations that cannot be performed, and correspondingly revise the process graph (ζ, ψ). Then, we can correct the Petri net model via Algorithm 1.

For a state X_k, its leaf nodes are obtained by firing k transitions from the initial state, and the leaf nodes are stored in the leaf node set Φ_k.

In order to calculate the scheduling strategy and control instructions, we proposed the Petri-net-based MPC method for a batch chemical system, which is summarized in Algorithm 2.

Algorithm 2: Petri-net-based MPC for batch chemical plants

Input: The place-timed Petri net G_{ts}, the initial state X_0, the prediction time domain H, the target X_g, the fault event η_k, the feedback time α_k, and urgent order event ε_k, the leaf node set Φ_k
Output: The control action (transition) e_k
Parameter initialization:$X_k = X_0$, $\Phi_k = \emptyset$;
while *state* X_k *is equal to target state* X_g **do**
  Calculate the control action (transition) e_k by the function **Rolling Optimization**(X_k, H, Φ_k), i.e., $e_k \leftarrow$ **Rolling Optimization**(X_k, H, Φ_k);
  Calculate next state X_{k+1} after firing transition e_k according to Equation (2)–(4);
  if $(\alpha_k \neq d) \vee (\eta_k = 1) \vee (\varepsilon_k \neq 0)$ **then**
    Correct the delay of d for the Petri net model G_{ts} by the function **Place Delay Model Correction**(α_k, d, e_k), i.e., $d \leftarrow$ **Place Delay Model Correction**(α_k, d, e_k);
    Correct the state X_{k+1} and the Petri net model G_{ts} by the function **Fault Event Model Correction**(η_k, X_{k+1}), i.e., $X_{k+1}, G_{ts} \leftarrow$ **Fault Event Model Correction**(η_k, X_{k+1});
    Correct the state X_{k+1} and goal state X_g by the function **Urgent Order Model Correction**$(X_{k+1}, X_g, \varepsilon_k)$, i.e., $X_{k+1}, X_g \leftarrow$ **Urgent Order Model Correction**$(X_{k+1}, X_g, \varepsilon_k)$;
Function: Rolling Optimization(X_k, H, Φ_k) **return** the control action (transition) e_k;
if $\Phi_k = \emptyset$ **then**
  Expand the sub-reachability graph whose root and depth are the state X_k and the prediction domain H and place the leaf nodes into Φ_k;
else
  Expand the sub-reachability graph whose root and depth are the state X_k and the prediction domain one and place the leaf nodes into Φ_k;
Select the leaf state X_s in the reachability graph with the minimum value of $g_s(m_s) + h_s(m_s)$, where $h_s(m_s)$ is calculated from Equation (6);
Backtrack the state X_s to the root state to obtain the optimal transition e_k;
return the control action (transition) e_k;
Function: Place Delay Model Correction(α_k, d, e_k) **return** d;
if *operation real time* α_k *in the operation set* $O(e_k)$ *is not equal to* d **then**
  for $o \in O(e_k)$ **do**
    $d(p_o) = \alpha_k(o)$;
return d;
Function: Fault Event Model Correction(η_k, X_{k+1}) **return** place-timed Petri net G_{ts} and state X_{k+1};
if $\eta_k = 1$ **then**
  Based on the faulty resources in η_k, the parameters of X_{k+1} are corrected, the process graph (ζ, ψ) is updated, and the place-timed Petri net model G_{ts} is regenerated by Algorithm 1;
return the state X_{k+1} and the place-timed Petri net model G_{ts};
Function: Urgent Order Model Correction$(X_{k+1}, X_g, \varepsilon_k)$ **return** X_{k+1}, X_g;
if $\varepsilon_k \neq 0$ **then**
  $m_{k+1}(p_J) = m_g(p_{\bar{J}}) = \varepsilon_k$, where $m_{k+1} \in X_{k+1}$ and $m_g \in X_g$;
return X_{k+1}, X_g;

Algorithm 2 consists of four functions: rolling optimization, place delay model correction, fault event model correction, and rush order model correction. The first one is to calculate a control action in each time step, and the others are responsible for correcting the Petri net model according to the feedback data.

Step 1 is to initialize the elements. Steps 2–9 make up a loop, where the state X_k variable is the cyclic variable. In every iteration, the rolling optimization function, place delay model correction, fault event model correction, and urgent order model correction are respectively called according to the state feedback. The loop stops, and the algorithm exits until the target state is reached. In Steps 9–16, the rolling optimization function seeks to generate the reachability graph with the limited depth defined by the prediction domain H, to evaluate each leaf node by the heuristic function, and to identify the most promising transition e_k. After firing e_k, the next marking m_{k+1} is obtained and can be used to determine all marked operation places. We can in turn obtain all the operations being carried out in m_{k+1} and the valve instructions according to the resource requirement function. In Steps 17–21, the place delay model correction function updates the delays of places if the operation time in the operation set α_k is not equal to d. In Steps 22–25, the fault event model correction function computes the new process graph (ζ, ψ) and regenerates the place-timed Petri net model by using Algorithm 1. In Steps 26–29, the urgent order model correction updates the markings of the batch-head places when an urgent order suddenly arrives.

Algorithm 2 combines the model predictive control method and the heuristic searching algorithm. It is able to flexibly adjust the depth of the reachability graph via the prediction time domain and quickly correct the system model according to the feedback.

7. Experiments

In order to verify the Petri-net-based MPC method, a series of numerical experiments are carried out on the batch chemical plant, as shown in Figure 1. We develop the Petri-net-based MPC Visual Studio 2017 simulation program in C++, and it is run on a computer with an i7-11700K 3.6 GHZ CPU and 32.0 GB RAM.

To evaluate the performance of the rolling optimization function in the proposed MPC algorithm, we conduct six experiments without an abrupt event, and the results are summarized in Table 2.

Table 2. Experiments on the Petri-net-based MPC algorithm without an abrupt event.

Experiments		Dijkstra	Petri-Net-Based MPC			
			Predication Domain H = 10		Predication Domain H = 30	
No.	Batch size	Makespan (min)	Makespan (min)	Search time (ms)	Makespan (min)	Search time (ms)
1	10	19.96	20.23	4.9	19.96	9.1
2	20	44.30	44.60	18.6	44.35	25.6
3	30	69.60	70.10	24.1	69.70	44.1
4	40	96.63	97.21	53.2	96.86	56.8
5	50	121.75	122.25	81.8	121.90	84.6
6	60	147.10	147.45	88.9	147.10	89.4

As shown in Table 2, the Dijkstra algorithm is used to obtain the optimal solutions, and two Petri-net-based MPC algorithms are applied, where the prediction times are, respectively, 10 and 30. In the Petri-net-based MPC experiments, the mean search times per step are no more than 90 ms. Hence, we can conclude that the MPC method can schedule the plant in real time.

The average makespan in the six experiments using the MPC method is within 0.5% of the average makespan of the Dijkstra method. Hence, our MPC method can obtain a near-optimal solution to the scheduling issue of the plant.

Furthermore, the makespans of the MPC with the predication time of 30 are closer to the optimal ones than those with the predication time of 10. This means that we can approach the optimal solutions by increasing the prediction time domain if we obtain enough computational resources.

To evaluate the MPC methods with different prediction time domains, we draw Gantt graphs of Experiment 1, as shown in Figure 3. It is evident that there are major differences between the scheduling schemes. This implies that it may be worth paying greater computational costs.

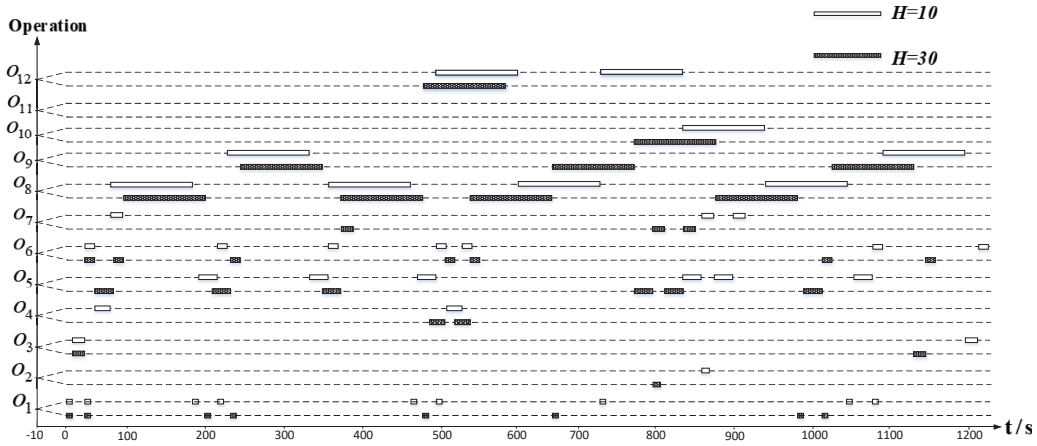

Figure 3. Gantt graphs of the MPC results in Experiment 1.

To evaluate our MPC method with model correction, we conducted three experiments, which adopted the process adjustment event, urgent order event, and equipment damage event, respectively. The results are summarized in Table 3.

Table 3. Experiments on the Petri-net-based MPC with abrupt events.

Experiments		No Model Correction Predication Domain H = 10	Model Correction Predication Domain H = 10
No.	Batch size	Makespan (min)	Makespan (min)
7	10	20.23	19.17
8	10	N/A	23.65
9	10	N/A	25.43

In Experiment 7, there are five abrupt events, in which the durations of o_8, o_9, o_{10}, o_{11}, and o_{12} change to 90 s, 100 s, 110 s, and 130 s, respectively. Once such an abrupt event occurs, the place delay model correction function is called to correct the delays of the places, and the rolling optimization function computes control actions with the new Petri net model. The executive process of the plant is shown in the Gantt graph, as shown in Figure 4. The makespan without model correction is 20.23 min, while that with model correction is 19.17 min. This means that the model correction function is useful to improve the scheduling solution.

In Experiment 8, two urgent orders occur at 154 s and 172 s, respectively. The MPC algorithm is able to immediately adjust the schedule so that all orders are completed in 23.65 min.

In Experiment 9, the filters f_1 and f_2 are damaged at 465 s and 849 s, respectively. The MPC algorithm is able to regenerate the Petri net model immediately and recompute the scheduling strategy. As a result, the makespan is 25.43 min, which reflects the strong robustness of our MPC method.

From the above-mentioned results, our MPC method is able to compute the schedule scheme in real time and is robust enough to adapt to abrupt events. Further, its solution is very close to the optimal one.

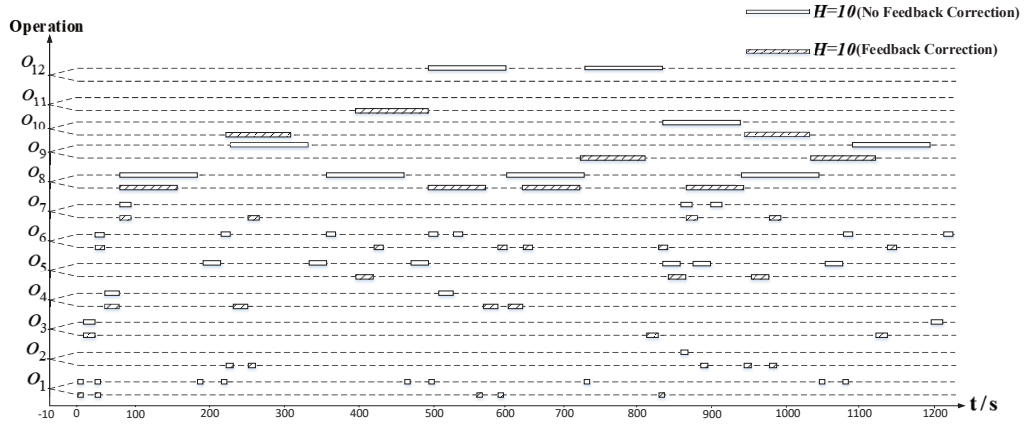

Figure 4. Gantt graphs of the MPC results in Experiment 7.

8. Conclusions

The Petri-net-based MPC method is presented to schedule a batch chemical plant in real time. It consists of two algorithms. The first one aims to design the Petri net model automatically for a given batch chemical system. The second is made up of the rolling optimization function and three model correction functions. The proposed method allows the system to continuously optimize resource allocation while performing tasks, thereby achieving an efficient and intelligent production process.

The heuristic function and the predictive time domain are two important issues that affect the performance of the proposed method. However, the fixed prediction time domain limits the search area in the state space and this leads to a sacrifice in the quality of the scheduling strategy. In the future, we aim to improve the heuristic function via deep learning and to explore a method to dynamically adjust the prediction domain time such that the makespan can be shortened.

Author Contributions: Z.L. is responsible for preparing, creating, and presenting published works, and clearly writing the initial draft; J.Z. (Jiazhong Zhou) is responsible for conducting a research and investigation process; S.S. is responsible for programming, software development, designing computer programs, implementation of the computer code and supporting algorithms, testing of existing code components; J.L. is responsible for preparation, creation of the published work, specifically critical review, revision; J.Z. (Jiabing Zhang) is responsible for management and coordination responsibility for the research activity planning and execution. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant 61973130, the Fujian Provincial Science and Technology Development Project Led by the Central Government under Grant 2022L2012, and the Natural Science Foundation of Fujian Province under Grant 2021J0129.

Data Availability Statement: Data supporting the reported results can be found in the paper.

Conflicts of Interest: The authors declare that they have no conflicts of interest related to this work.

References

1. Murata, T. Petri nets: Properties, analysis and applications. *Proc. IEEE* **1989**, *77*, 541–580. [CrossRef]
2. Alessandro, G.; Manuel, S. Petri nets and Automatic Control: A historical perspective. *Annu. Rev. Control* **2018**, *45*, 223–239.
3. Tianlong, G.; Bahri, P.A. A survey of Petri net applications in batch processes. *Comput. Ind.* **2002**, *47*, 99–111.
4. Manuel, S. On the history of Discrete Event Systems. *Annu. Rev. Control* **2018**, *45*, 213–222.
5. Tittus, M.; Lennartson, B. Hierarchical supervisory control for batch processes. *IEEE Trans. Control. Syst. Technol.* **1999**, *7*, 542–554. [CrossRef]
6. Wang, Y.; Chou, H.; Chang, C. Generation of batch operating procedures for multiple material-transfer tasks with Petri nets. *Comput. Chem. Eng.* **2005**, *29*, 1822–1836. [CrossRef]

7. Wang Y.; Chang, C. A hierarchical approach to construct Petri nets for modeling the fault propagation mechanisms in sequential operations. *Comput. Chem. Eng.* **2003**, *27*, 259–280. [CrossRef]
8. Susumu, H.; Tomoyuki, Y.; Takashi, I.; Katsuaki, O. Synthesis of Operating Procedures and Procedural Controllers for Batch Processes Based on Petri Nets. *J. Chin. Inst. Chem. Eng.* **2004**, *35*, 363–369.
9. Petter, F.; Bengt, L.; Michael, T. Specification of a batch plant using process algebra and petri nets. *Control. Eng. Pract.* **2009**, *17*, 1004–1015.
10. Mahsa, G.; Bahri, P.A.; Lee, P.; Gu, T. Petri-net based formulation and algorithm for short-term scheduling of batch plants. *Comput. Chem. Eng.* **2005**, *29*, 249–259.
11. Lai, J.; Chou, H.; Chang, C. Petri-net based integer programs for synthesizing optimal material-transfer procedures in pipeline networks. *J. Chin. Inst. Eng.* **2006**, *29*, 337–346. [CrossRef]
12. Samuel, W.W.; Anurag, R.; Sean, W. Model approximation for batch flow shop scheduling with fixed batch sizes. *Discret. Event Dyn. Syst.-Theory Appl.* **2015**, *25*, 487–529.
13. Lin, W.; Luo, J.; Zhou, J.; Huang, Y.; Zhou, M. Scheduling and control of batch chemical processes with timed Petri nets. In Proceedings of the IEEE International Conference on Automation Science and Engineering, Fort Worth, TX, USA, 21–25 August 2016.
14. Zhou, J.; Lefebvre, D.; Zhiwu, L. A Clustering Approach to Approximate the Timed Reachability Graph for a Class of Time Petri Nets. *IEEE Trans. Autom. Control* **2022**, *67*, 3693–3698. [CrossRef]
15. Lefebvre, D.; Leclercq, E. Control Design for Trajectory Tracking With Untimed Petri Nets. *IEEE Trans. Autom. Control* **2015**, *60*, 1921–1926. [CrossRef]
16. Lefebvre, D. Near-Optimal Scheduling for Petri Net Models With Forbidden Markings. *IEEE Trans. Autom. Control* **2018**, *63*, 2550–2557. [CrossRef]
17. Lefebvre, D. Dynamical Scheduling and Robust Control in Uncertain Environments with Petri Nets for DESs. *Processes* **2017**, *5*, 54. [CrossRef]
18. Cahyono, R.T.; Jacob, F.E.; Bayu, J. Discrete-Event Systems Modeling and the Model Predictive Allocation Algorithm for Integrated Berth and Quay Crane Allocation. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 1321–1331. [CrossRef]
19. Camacho, E.F. Model Predictive Control and Hybrid Systems. In *Model Predictive Control*; Springer: London, UK, 2007; pp. 289–310.
20. Yeh, M.L.; Chang, C.T. An automata based method for online synthesis of emergency response procedures in batch processes. *Comput. Chem. Eng.* **2012**, *38*, 151–170. [CrossRef]