IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 13, 2020, accepted August 21, 2020, date of publication September 1, 2020, date of current version September 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3020805

Modeling and Scheduling Methods for Batch Production Systems Based on Petri Nets and Heuristic Search

JIAZHONG ZHOU[1], JILIANG LUO[2,3], (Senior Member, IEEE), DIMITRI LEFEBVRE[4], (Senior Member, IEEE), AND ZHIWU LI[1,5], (Fellow, IEEE)

[1]School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China
[2]College of Information Science and Engineering, Huaqiao University, Xiamen 361021, China
[3]Fujian Engineering Research Center of Motor Control and System Optimal Schedule, Xiamen 361021, China
[4]GREAH, University of Le Havre Normandie, 76600 Le Havre, France
[5]Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau

Corresponding author: Jiliang Luo (jlluo@hqu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1700104, and in part by the National Natural Science Foundation of China under Grant 61573158, Grant 61873442, and Grant 61973130.

ABSTRACT An approach is proposed for modeling and scheduling batch production systems based on Petri nets. First, a group of jobs to be carried out by a batch production system is modeled by a transition-timed Petri net according to logical relations between operations and time requirements on operations. Second, conflict relations between operations are obtained by operation-resource diagrams that describe how operations compete for resources, and are formally expressed by linear constraints. Third, monitor places are designed to enforce linear constraints on the transition-timed Petri net, and a plant net is consequently built and made free of conflicts between operations, which can well emulate a real batch production system. Fourth, an optimal schedule problem, where the optimization performance index is to minimize a makespan, is formalized based on a plant net. Finally, a new variant of filtered beam search method is proposed to solve the problem based on an ad hoc function and a dynamical timed extended reachability graph of a plant net. A typical chemical production plant illustrates the theoretic results.

INDEX TERMS Petri net, modeling, optimal schedule, timed extended reachability graph, batch production.

I. INTRODUCTION

In a batch production process, operations require a large number of resources, such as valves and reactors. These resources are often shared by different operations such that some operations cannot be executed at the same time. However, certain operations that do not compete for resources can be performed simultaneously. Therefore, there are various possible sequences of operations to accomplish a given group of jobs, and they have different makespans. It is a valuable issue to express the relations between resources and operations in order to model a batch production process and to compute an optimal schedule.

A batch production system can be viewed as a discrete event system (DES) since its evolution is driven by various events such as starting and ending an operation, opening and closing a valve, and switching on and off a heater. Petri nets are a graphical and mathematical tool that provides a uniform framework for modeling, analysis, and control of DESs [1]. Tittus and Akesson [2] design Petri net models for plant resources and production recipes, where recipes consist of elementary tasks, such as carrying, mixing, adding and splitting of material. Furthermore, general Petri net building blocks are used to represent elementary tasks, and, in turn, to formally describe a recipe. Falkman *et al.* [3] propose a process algebra Petri net that is utilized to represent desired specifications of a batch process. Ferrarini and Piroddi [4] employ a hierarchical approach to model complex fluid transportation operations, where a higher level is for allocating resources and synchronizing operations, and a lower one is for executing operations. Wu *et al.* [5], [6] design a hybrid colored Petri net for modeling a crude-oil refining process such that a scheduling problem can be formulated in the framework of the control theory of hybrid systems. The above

The associate editor coordinating the review of this manuscript and approving it for publication was Ayaz Ahmad.

mentioned methods fail to formally determine and represent conflict relations among operations by Petri net structures since it is a challenging issue due to the large number of devices and their complicated relations defined by recipes. However, to do so is valuable since a well developed Petri net model makes it possible to address the control and scheduling problems via the Petri net theory.

Furthermore, the processing times of operations need to be taken into account if a scheduling issue is addressed. Timed Petri nets (TPNs) that encode time specifications are used for modeling batch production systems [7]. Tsinarakis and Tsourveloudis [8] present a transition-timed Petri net (T-TPN) for the modeling, analysis, synthesis and performance evaluation of batch production systems. Ghaeli *et al.* [9] model batch plants with place-timed Petri nets (P-TPNs), where time constants associated with places represent durations of operations. In general, the size of a T-TPN model is smaller than a P-TPN one for a batch production system. Baruwa *et al.* [10] apply timed colored Petri nets (TCPNs) to model flexible manufacturing systems, where each operation has a certain number of preconditions, an estimated duration, and a set of postconditions. Yang *et al.* [11] extend a resource-oriented Petri net to model a cluster tool in wafer fabrication. They define colors of transitions, and associate both transitions and places with time. In addition, under uncertain environment, TPNs can be extended to associate transitions with stochastic transition firing times [12].

For batch production systems, a schedule issue can be represented by a mixed integer linear programing (MILP) [9]. However, the number of constraints and variables is large and MILP may become untractable. Since TPNs have powerful graphical and algebraic representations and asynchronous and parallel capabilities, TPNs can model systematically behaviors of batch production systems. Furthermore, firing rules of the enabled transitions of TPNs play the roles of complex time constraints in MILP. Pioneer works try to tackle schedule issues by using timed extensions of reachability graphs of T-TPNs, as the state class graph [13], [14], timed aggregate graphs [15] and modified state class graphs [16]. Based on a relaxed mixed semantics model [17], a state class method is presented for the verification and analysis of temporal properties. More recently, control and schedule issues are considered with timed reachability graphs [18] and timed extended reachability graphs (TERGs) at the earliest firing policy [19].

An alternative research direction is to develop informed graph search algorithms, such as Dijkstra, A* and beam search ones, with untimed reachability graphs of Petri nets. Lee and Dicesare [20] design an A* algorithm to heuristically search for an optimal branch in a reachability graph. Xiong and Zhou [21] propose two hybrid scheduling methods that integrates a heuristic best-first strategy and a controlled backtracking one. Mejia and Odrey [22] present an improved heuristic search, where an aggressive node pruning strategy and an improved evaluation function are used. Li *et al.* [23] design an admissible heuristic function based on available periods. Huang *et al.* [24] adopt admissible and non-admissible heuristic functions to improve the searching efficiency. Xing *et al.* [25] propose a scheduling method integrating a deadlock control policy and a genetic algorithm to obtain a strategy. Luo *et al.* [26] redefine an existing deadlock prevention policy and embed it into a scheduling method. Zhang *et al.* [27] model an assembly process by TPNs, and dispatch tasks with a dynamic programming algorithm, where a depth-first search and a heuristic policy are utilized to select a most promising path. In [28], a filtered beam search (FBS) algorithm is presented where an evaluation function is used to pre-valuate and filter out nodes that seem not promising. Mejia and Nino [29], [30] introduce a filtering mechanism to limit the number of nodes at each expanding step such that the complexities in space and time are reduced significantly. These methods need to evaluate a cost function. Since time information is absent in reachability graphs, it is difficult to design and improve heuristic functions.

The main contents and contributions of this work are summarized as follows.

1) A method is presented to model batch production systems by T-TPNs, where productive processes and conflicts between operations are represented by Petri net structures. Compared with [2]–[6], conflict relations among operations are accurately described by Petri net structures, and this is helpful to formulate and solve scheduling problems of batch production systems. In detail, an operation-resource diagram is defined for valves and tanks to formally describe the relationship between operations and resources, and is used to correctly derive conflict sets which are sets of operations that cannot be performed in the same time. In turn, linear constraints are used to express conflict sets, and are enforced on the T-TPN model by designing their monitor places. As a result, a plant Petri net is derived to accurately describe and to imitate dynamical behaviors of batch plants.
2) A dynamical timed extended reachability graph (D-TERG) is introduced to preserve the timed properties of T-TPNs on specific regions. In contrast to other existing timed graphs [15], [16], [19], the D-TERG encodes only the part of the timed language that is necessary for a particular schedule issue. The D-TERG is a dynamical graph that grows up according to a sequence of states that are successively explored.
3) Based on the D-TERG, a variant of the FBS is proposed with an improved heuristic cost function. This function uses the characteristics of the states of the D-TERG and refines the estimation of the remaining time required to complete the schedule compared to the usual cost functions that search the shortest paths from the marked places to a reference one. In particular it takes into account the residual times of enabled transitions and extra waiting times of non enabled transitions due to conflicts about resources.

The main challenge of the proposed approach is certainly to combine both the D-TERG design with the new variant of the FBS. On the one hand, the proposed heuristic cost function is based on the D-TERG obtained at a certain point. On the other hand, the iterative construction of the D-TERG is driven by the FBS that prunes non-promising branches. This new scheduling strategy is successfully applied to the T-TPN models of the batch production system.

The remaining parts of this paper are organized as follows. In Section II, the concepts and notations of Petri nets are reviewed. In Section III, a scheduling problem of batch production systems is formulated. In Section IV, a method is given for modeling a batch production system. Section V shows how to solve a scheduling problem with D-TERG and a new variant of FBS algorithm. In Section VI, examples are presented to illustrate the proposed methods. Section VII concludes this paper.

II. PRELIMINARY

A. PETRI NETS

A Petri net structure is $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$, where P is a set of m places and T is a set of n transitions with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$; $\boldsymbol{Pre} := P \times T \to \mathbb{N}$, and $\boldsymbol{Post} := P \times T \to \mathbb{N}$ are the pre-incidence and post-incidence functions that specify the arcs, where $\mathbb{N}$ is the set of nonnegative integers. Given a place $p_i \in P$ and a transition $t_j \in T$, the element of matrix $\boldsymbol{Pre}$ at row i and column j is denoted by $\boldsymbol{Pre}(p_i, t_j)$ or $\boldsymbol{Pre}(i, j)$. The same holds for $\boldsymbol{Post}$. A marking is a function $\boldsymbol{m} : P \to \mathbb{N}$ that assigns to each place of a Petri net structure a non-negative integer number of tokens, represented by black dots. The number of tokens in $p \in P$ at $\boldsymbol{m}$ is denoted by $\boldsymbol{m}(p)$.

A Petri net is $(N, \boldsymbol{m}_0)$ with a Petri net structure N and an initial marking $\boldsymbol{m}_0$. Given a marking $\boldsymbol{m}$, $P(\boldsymbol{m}) \subseteq P$ denotes the set of places with a non-zero number of tokens at $\boldsymbol{m}$, i.e., the support of $\boldsymbol{m}$. A transition t is enabled at marking $\boldsymbol{m}$ if $\boldsymbol{m} \geq \boldsymbol{Pre}(:, t)$, which is denoted by $\boldsymbol{m}[t\rangle$. The enabled degree of transition t at $\boldsymbol{m}$ is defined as $n_t(\boldsymbol{m}) = \min\{\lfloor \boldsymbol{m}(p)/\boldsymbol{Pre}(p, t) \rfloor, \boldsymbol{m}(p) \in {}^\bullet t\}$, where ${}^\bullet t$ stands for the set of input places of t: ${}^\bullet t = \{p \in P | \boldsymbol{Pre}(p, t) > 0\}$, and $\lfloor x \rfloor$ stands for the largest integer smaller than or equal to real number x. The set of all enabled transitions at $\boldsymbol{m}$ is denoted as $\boldsymbol{m}^\bullet$. The fact that a Petri net reaches $\boldsymbol{m}'$ from $\boldsymbol{m}$ via the firing of t is denoted by $\boldsymbol{m}[t\rangle\boldsymbol{m}'$. A firing sequence σ is defined as $\sigma = t_{\sigma_1} t_{\sigma_2} \ldots t_{\sigma_h}$ where $\sigma_1, \ldots, \sigma_h$ are the indexes of the transitions, the length of σ is $h = |\sigma|$, and $\sigma = \varepsilon$ stands for an empty sequence. A marking $\boldsymbol{m}$ is reachable from an initial marking $\boldsymbol{m}_0$ if there exists a firing sequence σ such that $\boldsymbol{m}_0[\sigma\rangle\boldsymbol{m}$, and σ is enabled at $\boldsymbol{m}_0$. $R(\boldsymbol{m}_0)$ is the set of all markings reachable from $\boldsymbol{m}_0$, and, for simplicity, we denote this set by R when no ambiguity exists. A Petri net is bounded if and only if (iff) there exists a natural integer k such that the number of tokens in each place does not exceed k for any reachable marking. The considered Petri nets in this paper are assumed to be bounded. A consequence of boundedness is that R is of finite cardinality.

B. TRANSITION-TIMED PETRI NETS

A transition-timed Petri net (T-TPN) is defined as $(N, \boldsymbol{m}_0, D)$, where N is a Petri net structure, $\boldsymbol{m}_0$ is the initial marking, and $D : T \to \mathbb{R}^+$ ($\mathbb{R}^+$ is the set of nonnegative real numbers) is the function that associates each transition $t \in T$ with a duration $D(t)$, which is the minimal delay that t should take since it is enabled. If $D(t) = 0$, t can fire immediately once it is enabled. The time semantics of TPN depends on $D(t)$, $t \in T$, and the server, memory and choice policies [13]. In this paper, an infinite server policy, and enabling memory policy are adopted. With the enabling memory policy, at the entrance in a marking, the residual durations associated with still enabled transitions are decremented and the residual durations associated with disabled transitions are forgotten. The choice policy is a preselection performed by an external agent (for example, a scheduler): in case of concurrency or conflict, the selection of transitions that will fire next is decided by the scheduler. These specifications are discussed in Section V.

A timed firing sequence is written as $\sigma = (t_{j_1}, \tau_1)(t_{j_2}, \tau_2) \cdots (t_{j_h}, \tau_h)$, where $j_1, j_2, \ldots, j_h$ are the indexes of the transitions, $\tau_1, \tau_2, \ldots, \tau_h$ are the firing instants, and $0 \leq \tau_1 \leq \tau_2 \leq \cdots \leq \tau_h$. The timed trajectory associated with σ and starting at $\boldsymbol{m}_0$ is defined by

$$(\sigma, \boldsymbol{m}_0) = \boldsymbol{m}(0)[(t_{j_1}, \tau_1)\rangle\boldsymbol{m}(1)[(t_{j_2}, \tau_2)\rangle \cdots \boldsymbol{m}(h-1)[(t_{j_h}, \tau_h)\rangle\boldsymbol{m}(h), \quad (1)$$

such that $\boldsymbol{m}(0) = \boldsymbol{m}_0$. We call $\boldsymbol{m}(h)$ the final marking of $(\sigma, \boldsymbol{m}_0)$.

A path $ph = x_1\, x_2 \ldots x_K$ in a T-TPN structure is defined as an orderly and oriented succession of K nodes with $x_k \in T \cup P$ for $x_1, x_K \in P$ and $x_{k+1} \in x_k^\bullet$ for $k = 1, \ldots, K-1$. The duration of ph is defined as:

$$d(ph) = \sum_{t \in ph} D(t) \quad (2)$$

For any places $p, p' \in P$, let us define $PH(p, p')$ as the set of paths from the place p to the place p' and $ph^*(p, p') \in PH(p, p')$ as the path of shortest duration within $PH(p, p')$.

In order to define the states of a T-TPN system, it is necessary to define iteratively the residual times $\delta(t)$ of each transition t enabled at marking $\boldsymbol{m}(k)$, $k = 0, \ldots, h$ of a given trajectory of Eq. (1).

- For $k = 0$, a set of residual times associated to $\boldsymbol{m}(0) = \boldsymbol{m}_0$ is defined as a multiset $\Delta(\boldsymbol{m}_0)$ on an ordinary set $\{(t, D(t)) \in (T \times \mathbb{R}^+), t \in (\boldsymbol{m}_0)^\bullet\}$, where the number of occurrences of the element $(t, D(t))$ in $\Delta(\boldsymbol{m}_0)$ is $n_t(\boldsymbol{m}_0)$, i.e., for each transition t enabled at $\boldsymbol{m}_0$ with degree $n_t(\boldsymbol{m}_0)$, the residual time $D(t)$ is repeated $n_t(\boldsymbol{m}_0)$ times in $\Delta(\boldsymbol{m}_0)$.
- For $k > 0$, a set of residual times associated to $\boldsymbol{m}(k)$, conditioned by a timed trajectory $(\sigma, \boldsymbol{m}_0)$, is defined as a multiset $\Delta(\boldsymbol{m}(k))$ on an ordinary set $\{(t, \delta) \in (T \times \mathbb{R}^+), t \in (\boldsymbol{m}(k))^\bullet\}$, where the number of occurrences of the element (t, δ) in $\Delta(\boldsymbol{m}(k))$ is $\boldsymbol{Num}(\sigma, \boldsymbol{m}_0)$ that

$\boldsymbol{Num} : (T \times \mathbb{R}^+)^* \times \mathbb{N}^m \to \mathbb{N}^+$ is a function that gives the number of occurrences of the element (t, δ) in $\Delta(\boldsymbol{m}(k))$ when the timed trajectory $(\sigma, \boldsymbol{m}_0)$ is executed; δ is the minimal residual time before firing t and computed as $\delta = \max\left(0, \delta' - (\tau_k - \tau_{k-1})\right)$ (with $\tau_0 = 0$) if (1) $(t, \delta') \in \Delta(\boldsymbol{m}(k-1))$ and (2) t is not disabled by the firing of t_{j_k}; otherwise $\delta = D(t)$. It is noted that the number of occurrences of the element (t, δ) in $\Delta(\boldsymbol{m}(k))$ is at most $n_t(\boldsymbol{m}(k))$ when t is enabled $n_t(\boldsymbol{m}(k))$ times at $\boldsymbol{m}(k)$ (due to the infinite server policy).

Observe that some interactions exist between the infinite server policy and the enabling memory policy. At the entrance in $\boldsymbol{m}(k)$, the enabling degree of some transitions t may decrease to a non-zero value. In that case the $n_t(\boldsymbol{m}(k-1)) - n_t(\boldsymbol{m}(k))$ latest minimal residual times δ are used to update $\Delta(\boldsymbol{m}(k))$.

A timed trajectory $(\sigma, \boldsymbol{m}_0)$ of the form (1) is feasible if (1) t_{j_k} is enabled at $\boldsymbol{m}(k-1)$, $k = 1, \ldots, h$; (2) the minimal residual times in $\Delta(\boldsymbol{m}(k))$, $k = 1, \ldots, h$ are elapsed. Consequently, the set of states $\boldsymbol{S}(\boldsymbol{m}_0)$ of a given T-TPN system $(N, \boldsymbol{m}_0, D)$ is defined as $\boldsymbol{S}(\boldsymbol{m}_0) = \{(\boldsymbol{m}, \Delta)$ such that there exists a feasible timed trajectory $(\sigma, \boldsymbol{m}_0)$ of Eq. (1) with final marking $\boldsymbol{m}$ and $\Delta(\boldsymbol{m}) = \Delta\}$. For simplicity, we denote this set as $\boldsymbol{S}$ when no ambiguity exists.

In addition to the time semantics previously defined, we consider in this paper a subclass of TPN where the firing times are computed with the earliest firing policy (EFP). When a transition is preselected for the next firing, it will fire as soon as its residual time has elapsed and its firing cannot be deferred. More formally, a feasible timed trajectory $(\sigma, \boldsymbol{m}_0)$ of the form (1) is an EFP-trajectory if each transition t_{j_k}, $k = 1, \ldots, h$, of the trajectory satisfies $(t_{j_k}, \tau_k - \tau_{k-1}) \in \Delta(\boldsymbol{m}(k-1))$ and for all $(t_{j_k}, \delta) \in \Delta(\boldsymbol{m}(k-1))$, $\delta \geq \tau_k - \tau_{k-1}$. Note that the set of feasible timed trajectories is the timed language of the TPN system and that the subset of EFP-trajectories can be viewed as the sublanguage of the TPN under EFP.

C. EXTENDED TIMED REACHABILITY GRAPH

In this section we give some basic notions about the extended timed reachability graph (TERG) for T-TPN systems that behave under EFP. The TERG has been defined and studied in [38] for T-TPN systems that satisfy the following assumptions.

- $\boldsymbol{A1}$: the TPN system is bounded, i.e., there exists a positive constant k such that, for all $\boldsymbol{m} \in R(\boldsymbol{m}_0)$ and for all $p \in P$, $\boldsymbol{m}(p) \leq k$;
- $\boldsymbol{A2}$: the minimal firing time $D(t)$ of any transition $t \in T$ is a multiple of a common time stamp.

In particular in [38], the TERG is proved to represent the timed language of a T-TPN system that behaves under EFP, and each state of the TERG coincides to a state of the T-TPN under EFP. Let $(N, \boldsymbol{m}_0, D)$ be a TPN system. The TERG of $(N, \boldsymbol{m}_0, D)$ is defined as a four-tuple $(\boldsymbol{S_E}, \Omega_E, B_E, S_0)$, where

- $\boldsymbol{S_E}$ is a finite set of N_E states, and each state $S \in \boldsymbol{S_E}$ is of the form $S = (\boldsymbol{m}(S), \Delta(S))$,
- $\Omega_E \in (T \cup \{\varepsilon\})^{N_E \times N_E}$ is the labeled adjacency matrix of the graph: for all $S, S' \in \boldsymbol{S_E}(\boldsymbol{m}_0)$, $\Omega_E(S, S') = t$ if there exists $(t, \delta) \in \Delta(S)$, otherwise $\Omega_E(S, S') = \varepsilon$.
- $B_E \in (\mathbb{R}^+)^{N_E \times N_E}$ is the earliest firing time matrix: for all $S, S' \in \boldsymbol{S_E}(\boldsymbol{m}_0)$, $B_E(S, S') = \delta$ if (1) there exists $(t, \delta) \in \Delta(S)$ with $\Omega_E(S, S') = t$ and (2) for all $(t, \delta') \in \Delta(S)$, $\delta \leq \delta'$ (i.e. t fires at earliest); otherwise $B_E(S, S') = \infty$.
- $S_0 = (\boldsymbol{m}_0, \Delta(\boldsymbol{m}_0))$ is the initial state.

A path $ph = S_1\, S_2 \ldots S_K$ in TERG $(\boldsymbol{S_E}, \Omega_E, B_E, S_0)$ is defined as an orderly and oriented succession of K states with $S_k \in \boldsymbol{S_E}$ for $k = 1, \ldots, K$ such that $\Omega_E(S_k, S_{k+1}) \neq \varepsilon$ for $k = 1, \ldots, K-1$. The duration of ph is defined as:

$$d(ph) = \sum_{k=1}^{K-1} B_E(S_k, S_{k+1}) \tag{3}$$

For any states $S, S' \in \boldsymbol{S_E}$, let us define $PH_E(S, S')$ as the set of paths from the state S to the state S' and $ph^*(S, S') \in PH_E(S, S')$ as the path of shortest duration within $PH_E(S, S')$.

III. PROBLEM DESCRIPTION

In a batch production system, each type of product corresponds to a job that consists of a series of operations. The execution of an operation requires a set of resources, such as valves and tanks, and takes a certain time. This indicates that multiple operations may compete for the same resource. The executive logic of operations depends not only on orders defined by jobs but also on conflicts due to operations competing for limited resource. Thus, it is complicated to describe processes of a batch production system.

Given a considered batch production system, $O = \{o_1, o_2, \ldots\}$ is the set of operations, and $\Omega = V \cup U$ is the set of resources, where $V = \{v_1, v_2, \ldots\}$ denotes the set of valves, $U = \{u_1, u_2, \ldots\}$ denotes the set of containing units such as supply or storage tanks and reactors. The set of all resource states is $\Omega_s = \{v_{s1}, \bar{v}_{s1}, v_{s2}, \bar{v}_{s2}, \ldots, u_{s1}, \bar{u}_{s1}, u_{s2}, \bar{u}_{s2}, \ldots\}$, where $v_{s1}, v_{s2}, \ldots$ are open states of valves, $\bar{v}_{s1}, \bar{v}_{s2}, \ldots$ are close states of valves, $u_{s1}, u_{s2}, \ldots$ are empty states of containing units, and $\bar{u}_{s1}, \bar{u}_{s2}, \ldots$ are full states of them. For an operation o, $r(o) : o \to \Omega_s$ is the set of resources that o requires, and $d(o) : o \to \mathbb{R}^+$ is the processing time of o.

Definition 1: Given a considered batch production system, a job $J_j = o_{j.1} o_{j.2} o_{j.3} \ldots o_{j.I}, j, I \in \mathbb{N}^+$, is a series of operations $o_{j.1} o_{j.2} o_{j.3} \ldots$ that need to be executed according to the order induced by the operation indexes.

Let $\rho(J_j) : J_j \to \mathbb{N}^+$ be the number of times that J_j is to be taken. Given a job $J_j = o_{j.1} o_{j.2} o_{j.3} \ldots o_{j.I}$, the operation $o_{j.i+1}$ is the subsequent operation of $o_{j.i}$ and $1 \leq i \leq I-1$, and any sequence of successive operations in J_j is a sub-job of J_j. For example, $o_{j.3}$ is the subsequent operation of $o_{j.2}$, and $o_{j.1} o_{j.2} o_{j.3}$, $o_{j.1} o_{j.2}$ and $o_{j.2} o_{j.3}$ are sub-jobs of J_j.

Some operations cannot be concurrently executed since they compete for the same resource. For a same job, there exist different processing sequences that cost different time.

A typical scheduling problem is to search for a processing sequence that is carried out in a minimal time. In order to illustrate the proposed concepts and notations, a typical chemical plant is adopted as a running example.

Example 1: A typical chemical plant is shown in Fig. 1. Seven processing units, which are three supply tanks: u_1, u_2 and u_3, two reactors u_4 and u_5, and two storage tanks u_6 and u_7, are connected by a pipeline system including 13 valves $v_1, v_2, \ldots, v_{13}$.

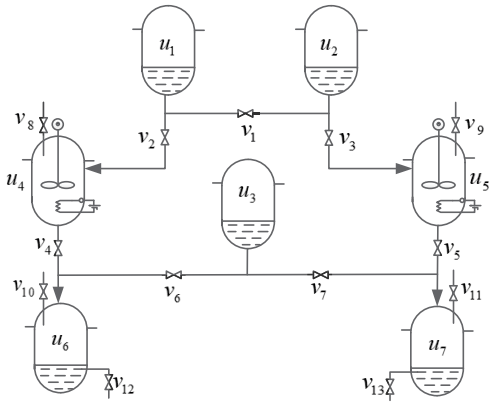

FIGURE 1. A batch production system.

There are two jobs $J_1 = o_{1.1}o_{1.2}o_{1.3}o_{1.4}o_{1.5}$ and $J_2 = o_{2.1}o_{2.2}o_{2.3}o_{2.4}o_{2.5}$ for manufacturing two types of products. In Table 1, the resource units and processing times that are required by the operations are summarized. An operation $o_{1.1}$ is to transport fluid from u_1 and u_2 to u_5 in 20 minutes. An operation $o_{1.2}$ is a chemical reaction in u_5 in 30 minutes. An operation $o_{1.3}$ is to transport material from u_5 and u_3 to u_7 in 30 minutes. An operation $o_{1.4}$ is a chemical reaction in u_7 in 40 minutes. An operation $o_{1.5}$ is to transport the first product in u_7 to another tank in 40 minutes. An operation $o_{2.1}$ is to transport material from u_1 to u_4 in 30 minutes. An operation $o_{2.2}$ is a chemical reaction in u_4 in 40 minutes. An operation $o_{2.3}$ is to transport material from u_4 and u_3 to u_6 in 40 minutes. An operation $o_{2.4}$ is a chemical reaction in u_6 in 50 minutes. An operation $o_{2.5}$ is to transport the second product in u_6 to another tank in 60 minutes.

TABLE 1. Two jobs of the batch production system shown in Fig. 1.

Type of jobs	$r(o_{j.i})$	$d(o_{j.i})$ (min.)
J_1, $\rho(J_1)=2$	$r(o_{1.1}) = \{v_{s1}, v_{s3}, \bar{v}_{s2}, \bar{v}_{s5}, \bar{u}_{s1}, \bar{u}_{s2}, u_{s5}\}$	$d(o_{1.1}) = 20$
	$r(o_{1.2}) = \{v_{s9}, \bar{v}_{s3}, \bar{v}_{s5}, \bar{u}_{s5}\}$	$d(o_{1.2}) = 30$
	$r(o_{1.3}) = \{v_{s5}, v_{s7}, \bar{v}_{s3}, \bar{v}_{s6}, \bar{v}_{s13}, \bar{u}_{s3}, \bar{u}_{s5}, u_{s7}\}$	$d(o_{1.3}) = 30$
	$r(o_{1.4}) = \{v_{s11}, \bar{v}_{s5}, \bar{v}_{s7}, \bar{v}_{s13}, \bar{u}_{s7}\}$	$d(o_{1.4}) = 40$
	$r(o_{1.5}) = \{v_{s13}, \bar{v}_{s5}, \bar{v}_{s7}, \bar{u}_{s7}\}$	$d(o_{1.5}) = 40$
J_2, $\rho(J_2)=2$	$r(o_{2.1}) = \{v_{s2}, \bar{v}_{s1}, \bar{v}_{s4}, \bar{u}_{s1}, u_{s4}\}$	$d(o_{2.1}) = 30$
	$r(o_{2.2}) = \{v_{s8}, \bar{v}_{s2}, \bar{v}_{s4}, \bar{u}_{s4}\}$	$d(o_{2.2}) = 40$
	$r(o_{2.3}) = \{v_{s6}, v_{s4}, \bar{v}_{s2}, \bar{v}_{s7}, \bar{v}_{s12}, \bar{u}_{s3}, \bar{u}_{s4}, u_{s6}\}$	$d(o_{2.3}) = 40$
	$r(o_{2.4}) = \{v_{s10}, \bar{v}_{s4}, \bar{v}_{s6}, \bar{v}_{s12}, \bar{u}_{s6}\}$	$d(o_{2.4}) = 50$
	$r(o_{2.5}) = \{v_{s12}, \bar{v}_{s4}, \bar{v}_{s6}, \bar{u}_{s6}\}$	$d(o_{2.5}) = 60$

As in Example 1, $r(o_{1.1}) = \{v_{s1}, v_{s3}, \bar{v}_{s2}, \bar{v}_{s5}, \bar{u}_{s1}, \bar{u}_{s2}, u_{s5}\}$, and $d(o_{1.1}) = 20$ minutes, which means that $o_{1.1}$ requires valves v_1 and v_3 to be opened, valves v_2 and v_5 to be closed, and tanks u_1 and u_2 are filled by fluid from tank u_5 for 20 minutes. For this production system, we assume that J_1 and J_2 are executed twice, thus $\rho(J_1) = \rho(J_2) = 2$. Obviously, some operations cannot be executed simultaneously, such as $o_{1.1}$ (with $r(o_{1.1}) = \{v_{s1}, v_{s3}, \bar{v}_{s2}, \bar{v}_{s5}, \bar{u}_{s1}, \bar{u}_{s2}, u_{s5}\}$) and $o_{1.2}$ (with $r(o_{1.2}) = \{v_{s9}, \bar{v}_{s3}, \bar{v}_{s5}, u_{s5}\}$), which are in conflict since $o_{1.2}$ requires v_3 to be opened and $o_{1.2}$ requires v_3 to be closed. Operations that do not compete for resources can be executed simultaneously. For example, $o_{1.1}$ can be executed with $o_{2.2}$ simultaneously. Different sequences have different processing time, and the key of this scheduling problem is to find an appropriate one to minimize the processing time.

IV. MODELING METHOD FOR BATCH PRODUCTION SYSTEMS

A. JOB MODELING

In essence, a job is composed of a series of operations in a certain order. For the scheduling of a batch production system, Algorithm 1 is presented to model a job.

Algorithm 1 Modeling a Job by Petri Nets

Input: A job $J_j := o_{j.1}o_{j.2}o_{j.3}\ldots o_{j.I}$, the processing duration $d(o_{i,j})$ to perform operation $o_{i,j}$, and the number of times $\rho(J_j)$ of the job J_j manufactured;

Output: The Petri net model $(P_{J_j}, T_{J_j}, \boldsymbol{Pre}_{J_j}, \boldsymbol{Post}_{J_j}, \boldsymbol{m}_{0,J_j}, D_{J_j})$ of the job J_j;

1: The starting place $p_{j.0}$ is designed, and its initial marking equals times by which J_j is to be manufactured, i.e., $P_{J_j} = \{p_{j.0}\}$ and $\boldsymbol{m}_{0,J_i}(p_{j.0}) = \rho(J_j)$;
2: **for** all $1 \leq i \leq I$ **do**
3: A transition $t_{j.i}$ is designed to model the operation $o_{j.i}$, i.e., $T_{J_j} = T_{J_j} \cup \{t_{j.i}\}$, $D_{J_j}(t_{j.i}) = d(o_{j.i})$;
4: A place $p_{j.i}$ is designed, which is the buffer place of the operation $o_{j.i}$, i.e., $P_{J_j} = P_{J_j} \cup \{p_{j.i}\}$;
5: The arcs $(p_{j.i-1}, t_{j.i})$ and $(t_{j.i}, p_{j.i})$ are designed;
6: **end for**

Algorithm 1 is presented to design a Petri net model of a job J_j including the ordering relationship, the processing time of operations, and the initial marking.

In Algorithm 1, Step 1 is to design the starting place for a job, and to mark it with $\rho(J_j)$; Steps 2–6 are to design the place, transition and arc sets of operations, and to determine the minimal duration $D(t_{i,j})$ of each transitions.

Example 2: By Algorithm 1, the Petri net model of $J_1 := o_{1.1}o_{1.2}o_{1.3}o_{1.4}o_{1.5}$ in Example 1 is designed as shown in Fig. 2. Place $p_{1.0}$ is the starting place, and $\boldsymbol{m}_{0,J_1}(p_{1.0}) = \rho(J_1) = 2$. There are five places $p_{1.1}$, $p_{1.2}$, $p_{1.3}$, $p_{1.4}$ and $p_{1.5}$, which are the buffer places of $o_{1.1}$, $o_{1.2}$, $o_{1.3}$, $o_{1.4}$ and $o_{1.5}$, respectively. There are five transitions $t_{1.1}$, $t_{1.2}$, $t_{1.3}$, $t_{1.4}$ and $t_{1.5}$, that represent the execution of $o_{1.1}$, $o_{1.2}$, $o_{1.3}$, $o_{1.4}$ and $o_{1.5}$, respectively. $D_{J_1}(t_{1.1}) = d(o_{1.1}) = 20$ min.,

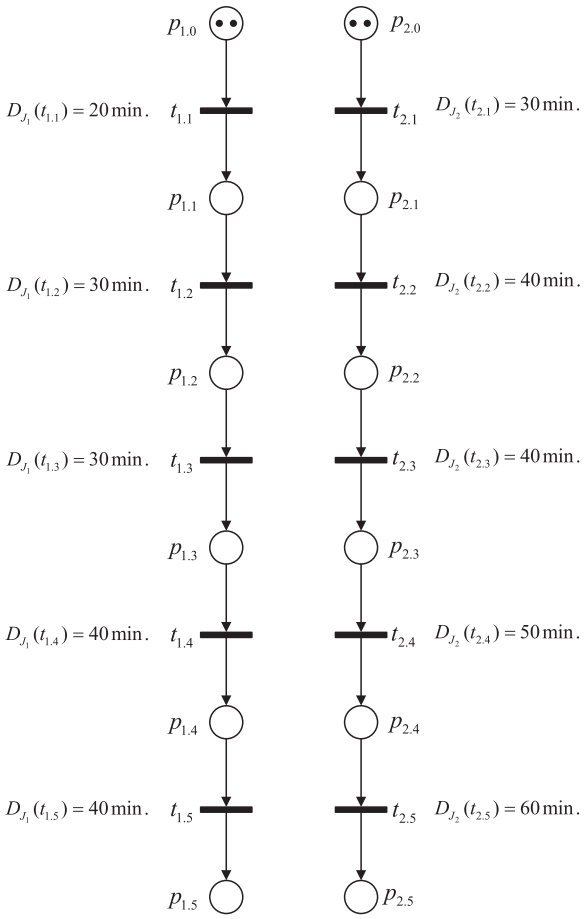

FIGURE 2. Petri net model of job J_1 and J_2 of the batch production system in Fig. 1.

$D_{J_1}(t_{1.2}) = d(o_{1.2}) = 30$ min., $D_{J_1}(t_{1.3}) = d(o_{1.3}) = 30$ min., $D_{J_1}(t_{1.4}) = d(o_{1.4}) = 40$ min., $D_{J_1}(t_{1.5}) = d(o_{1.5}) = 40$ min.

Similarly, the Petri net model of J_2 is designed based on Algorithm 1.

B. MODELING CONFLICTS BETWEEN OPERATIONS

The competition of resources among different operations results in conflicts. To accurately express conflicts is important for finding an optimal sequence to execute operations. In this section, the specific definition and model of conflicts are designed.

Definition 2: Two operations o and o' are conflict if and only if (iff) there exists a resource v (or u), such that $v_s \in r(o)$ and $\bar{v}_s \in r(o')$ (or $u_s \in r(o)$ and $\bar{u}_s \in r(o')$).

Definition 3: A set of operations is maximally conflict, denoted by $O_{\max}$, iff each operation in it is conflict with any other one in it, and there exists an operation in it that is not conflict with any operations not in it.

According to Definitions 2 and 3, any two operations of $O_{\max} = \{o_1, o_2, \ldots, o_x\}$ are conflict and cannot be executed simultaneously. One operation corresponds to one transition according to Algorithm 1. Any two transitions of $\{t_1, t_2, \ldots, t_x\}$ cannot fire simultaneously, where $t_1, t_2, \ldots, t_x$ are transitions representing the action of $o_1, o_2, \ldots, o_x$, respectively. Since $p_1, p_2, \ldots, p_x$ are the only output places of $t_1, t_2, \ldots, t_x$, respectively, a Petri net model of the batch production system should satisfy $\sum_{i=1}^{x} \boldsymbol{m}(p_i) \leq 1$. Since the relationship between operations and resources is complicated, it is difficult to directly find all maximal conflict operation sets. Then, an operation-resource diagram is defined to represent the relationship between resources and operations and to further obtain $O_{\max}$.

Definition 4: Given a considered batch production system, its operation-resource diagram $G_{OR} = (N_O, N_R, E_{OR})$ is an oriented graph, where N_O and N_R are the sets of nodes that represent operations and resources, respectively and E_{OR} is the set of monodirectional edges that represent the relationship between operations and resources (valves and tanks).

The procedure for designing an operation-resource diagram is presented in Algorithm 2.

Algorithm 2 Design of an Operation-Resource Diagram

Input: Types of jobs J_j, the set of resources Ω, the set of resources states Ω_s, and the set of resources states $r(o_{i,j})$ of operation $o_{i,j}$;

Output: An operation-resource diagram $G_{OR} = (N_O, N_R, E_{OR})$, where N_O and N_R are the sets of triangle nodes and rectangular nodes, respectively and E_{OR} is the set of monodirectional edges;

$N_O = N_R = \emptyset$, $E_{OR} = \emptyset$;
for all jobs J_i **do**
  for all operations $o_{i,j}$ of J_i **do**
    A triangle node n is designed, i.e., $N_O = \{n\} \cup N_O$;
  end for
end for
for all resources of Ω **do**
  A rectangular node n' is designed, i.e., $N_R = \{n'\} \cup N_R$;
end for
for all jobs J_i **do**
  for all operations $o_{i,j}$ of J_i **do**
    for all resources states of Ω_s **do**
      if $v_s \in r(o_{i,j})$ (or $u_s \in r(o_{i,j})$) **then**
        A monodirectional edge $e = (v, o_{i,j})$ (or $e = (u, o_{i,j})$) from the rectangular node of v (or u) to the triangle node of the operation $o_{i,j}$ is added, i.e., $E_{OR} = \{e\} \cup E_{OR}$;
      else
        if $\bar{v}_s \in r(o_{i,j})$ (or $\bar{u}_s \in r(o_{i,j})$ **then**
          A monodirectional edge $e' = (o_{i,j}, v)$ (or $e' = (o_{i,j}, u)$) from the triangle node of $o_{i,j}$ to the rectangular node of v (or u) is added, i.e., $E_{OR} = \{e'\} \cup E_{OR}$;
        end if
      end if
    end for
  end for
end for

In Algorithm 2, Steps 2–6 are to derive the set N_O of nodes for all operations, denoted by triangles; Steps 7–9 are to obtain the set N_R of nodes for all resources, denoted by rectangles; and Steps 11–23 are to draw monodirectional edges among these triangle and rectangular nodes.

Example 3: Let us design the operation-resource diagram for the batch production system in Example 1, as shown in Fig. 3. According to Steps 2–6 of Algorithm 2, ten triangle nodes are designed, which correspond to operations $o_{1.1}$, $o_{1.2}$, $o_{1.3}$, $o_{1.4}$, $o_{1.5}$, $o_{2.1}$, $o_{2.2}$, $o_{2.3}$, $o_{2.4}$ and $o_{2.5}$, respectively. Twenty rectangular nodes are designed by Steps 7–9 for all resources in $\Omega = \{v_1, v_2, \ldots, v_{13}, u_1, u_2, \ldots, u_7\}$.

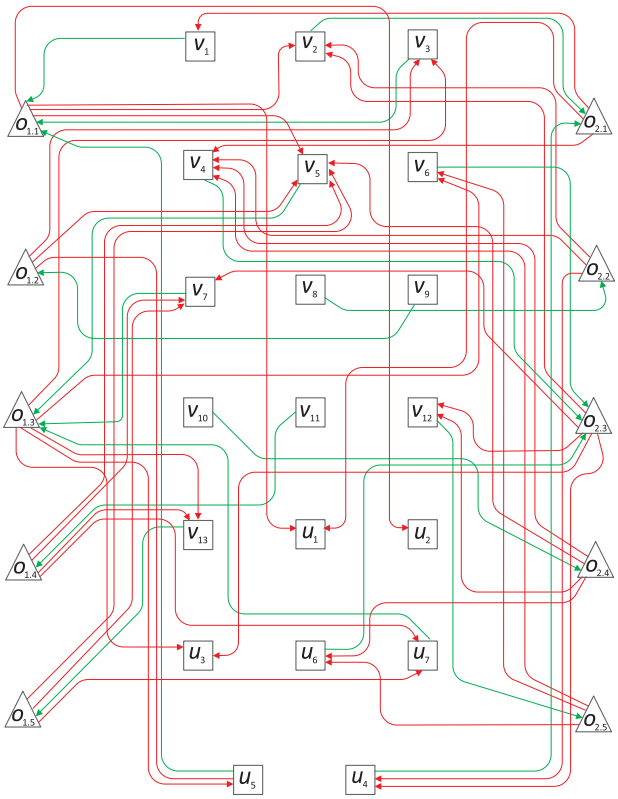

FIGURE 3. Operation-resource diagram of the batch production system shown in Fig. 1.

An operation-resource diagram correctly shows the relationship between operations and resources. The conflicts between different operations come from their competition for resources with different states. For representing and obtaining conflicts among operations, an operation-conflict graph is given in Definition 5.

Definition 5: Given a considered batch production system, an operation-conflict graph $G_{OC} = \langle N_{OC}, E_{OC} \rangle$ is an undirected graph, where N_{OC} is a set of triangle nodes that represent operations, and E_{OC} is a set of edges that connect nodes of conflict operations, i.e, nodes of operation o and o' are connected by one edge if o and o' are conflict.

The procedure for designing an operation-conflict graph is presented in Algorithm 3.

Algorithm 3 Design of an Operation-Conflict Graph

Input: An operation-resource diagram $G_{OR} = (N_O, N_R, E_{OR})$;

Output: An operation-conflict graph $G_{OC} = \langle N_{OC}, E_{OC} \rangle$;

1: $N_{OC} = N_O$ of N_{OR}, $E_{OC} = \emptyset$;
2: **for** all rectangular nodes N_O of G_{OR} **do**
3: **if** $\{(o, v) \in E_{OR} \wedge (v, o') \in E_{OR}\}$ **or** $\{(o, u) \in E_{OR} \wedge (u, o') \in E_{OR}\}$, where o and o' are two different operations **then**
4: **if** $|[o, o']| < 1$, where $|[o, o']|$ is the number of undirected edges $[o, o']$ of G_{OC} between the node of operation o and o' **then**
5: An undirected edge $[o, o']$ between the triangle node of operation o and o' is designed;
6: $E_{OC} = \{[o, o']\} \cup E_{OC}$
7: **end if**
8: **end if**
9: **end for**

In Algorithm 3, Step 1 is to design the nodes, and Steps 2–9 are to determine undirected edges among these nodes.

Example 4: For the operation-resource diagram in Fig. 3, the operation-conflict graph is obtained by Algorithm 3, as shown in Fig. 4.

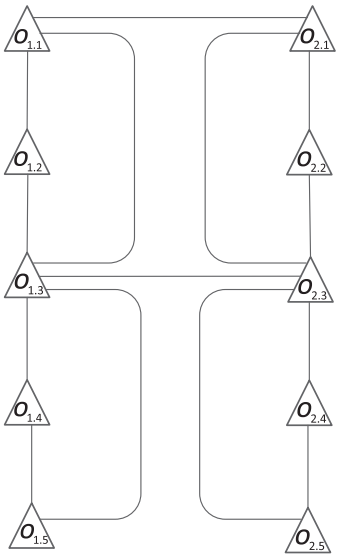

FIGURE 4. Operation-conflict graph of the operation-resource diagram in Fig. 3.

An operation-conflict diagram clearly represents the relationship between operations. Actually, a maximal conflict operation set $O_{\max}$ corresponds to a subgraph of G_{OC}. The corresponding definition and theorem are given as follows.

Definition 6: A clique of an operation-conflict graph G_{OC} is a subgraph of G_{OC}, denoted by Cl, if any two nodes of it are connected by one edge.

Definition 7: A clique of an operation-conflict graph G_{OC} is called a maximal clique of G_{OC}, denoted by $Cl_{\max}$, if there does not exist another clique Cl such that $Cl_{\max} \subset Cl$.

Theorem 1: Given a batch production system, there is a maximal conflict operation set $O_{\max} = \{o_1, o_2, \ldots, o_x\}$ iff there exists a maximal clique $Cl_{\max}$ of G_{OC} such that the nodes of $Cl_{\max}$ represent the operations of $O_{\max}$.

Proof: We start by proving the if-part and assume that there exists a maximal clique $Cl_{\max}$ consisting of x nodes $o_1, o_2, \ldots, o_x$. We also assume that $\{o_1, o_2, \ldots, o_x\}$ is not a maximal conflict operation set. Based on Definitions 2–7, it is obvious that any two nodes of $Cl_{\max}$ correspond to conflict operations. That is, any two operations of $\{o_1, o_2, \ldots, o_x\}$ are conflict. Since we assume that $\{o_1, o_2, \ldots, o_x\}$ is not a maximal conflict operation set, there is another operation $o' \notin \{o_1, o_2, \ldots, o_x\}$ that conflicts with any one operation of $\{o_1, o_2, \ldots, o_x\}$. Based on Definitions 5 and 6, nodes of $o', o_1, o_2, \ldots, o_x$ in G_{OC} form a clique Cl and $Cl_{\max} \subset Cl$. It is obvious that $Cl_{\max} \subset Cl$ is in conflict with the definition of $Cl_{\max}$. We conclude that the assumption is false. The set $\{o_1, o_2, \ldots, o_x\}$ is a maximal conflict operation set.

Then, we prove the only-if part. $O_{\max} = \{o_1, o_2, \ldots, o_x\}$ is a maximal conflict operation set. We assume that nodes of $o_1, o_2, \ldots, o_x$ cannot form a maximal clique. Based on Definitions 2–7, any two operations of $\{o_1, o_2, \ldots, o_x\}$ are conflict and any two nodes of $o_1, o_2, \ldots, o_x$ are connected in G_{OC}. Nodes of $o_1, o_2, \ldots, o_x$ in G_{OC} form a clique. In addition, nodes of $o_1, o_2, \ldots, o_x$ cannot form a maximal clique. That is, there exists another clique Cl such that $Cl_{\max} \subset Cl$. There exists one node in Cl such that it corresponds an operation $o' \notin (o_1, o_2, \ldots, o_x)$ that conflicts with any one operation of $o_1, o_2, \ldots, o_x$. Consequently, nodes of $o_1, o_2, \ldots, o_x$ form a maximal clique of G_{OC}. ▲

Theorem 1 converts the computation of $O_{\max}$ into a classic problem in the graph theory. There exist several algorithms to find maximal clique in the graph theory. In this paper, Bron-Kerbosch Algorithm [31] is used. In Algorithm 4, monitor places are designed based on maximal cliques of G_{OC} in order to prevent a Petri net model from entering any conflict state.

In Algorithm 4, by Steps 2–5, the Petri net model for all jobs of the batch production system is designed; by Steps 6–7, the operation-resource diagram G_{OR} and the operation-conflict graph G_{OC} are drawn; by Steps 8–9, all maximal cliques and all maximal conflict operation sets $O_{\max}$ are obtained; by Steps 10–21, monitor places are designed.

Example 5: Consider the batch production system in Fig. 1, Petri net models of J_1 and J_2 are modeled by Algorithm 1 and Steps 2–5 of Algorithm 4, where

1) there are two starting places $p_{1.0}$ and $p_{2.0}$, and $\boldsymbol{m}_{0,J_1}(p_{1.0}) = \rho(J_1) = 2$, $\boldsymbol{m}_{0,J_2}(p_{2.0}) = \rho(J_2) = 2$,

2) there are ten places $p_{1.1}$, $p_{1.2}$, $p_{1.3}$, $p_{1.4}$, $p_{1.5}$, $p_{2.1}$, $p_{2.2}$, $p_{2.3}$, $p_{2.4}$ and $p_{2.5}$, which are the buffer places of operations $o_{1.1}$, $o_{1.2}$, $o_{1.3}$, $o_{1.4}$, $o_{1.5}$, $o_{2.1}$, $o_{2.2}$, $o_{2.3}$, $o_{2.4}$ and $o_{2.5}$, respectively,

Algorithm 4 Modeling of a Batch Production System

Input: Types of jobs J_j, the set of resources Ω, the set of resources states Ω_s, the set of resources states $r(o_{i,j})$ of operation $o_{i,j}$, the processing duration $d(o_{i,j})$ to perform operation $o_{i,j}$, and the number of times $\rho(J_j)$ of the job J_j manufactured;

Output: A batch production system Petri net model $(P, T, \boldsymbol{Pre}, \boldsymbol{Post}, \boldsymbol{m}_0, D)$;

1: $P = \emptyset$, $T = \emptyset$;
2: **for** all job J_j **do**
3: The job Petri net model $(P_{J_j}, T_{J_j}, \boldsymbol{Pre}_{J_j}, \boldsymbol{Post}_{J_j}, \boldsymbol{m}_{0,J_j}, D_{J_j})$ is designed by Algorithm 1;
4: $P = P \cup P_{J_j}$, $T = T \cup T_{J_j}$;
5: **end for**
6: The operation-resource diagram $G_{OR} = (N_O, N_R, E_{OR})$ is designed by Algorithm 2;
7: The operation-conflict graph $G_{OC} = \langle N_{OC}, E_{OC} \rangle$ is designed by Algorithm 3;
8: Find all maximal cliques of G_{OC} based on Bron-Kerbosch Algorithm.
9: Find all maximal conflict operation sets $O_{\max}$, where one maximal conflict operation set corresponds to one maximal clique of G_{OC};
10: **for** all maximal conflict operation sets $O_{\max}$ **do**
11: The linear constraints of places corresponding to operations in $O_{\max}$ are got based on Theorem 1, and the monitor place p_c and corresponding arcs are designed based on those linear constraints [32]–[37]:
12: A monitor place p_c is designed, and $\boldsymbol{m}_0(p_c) = 1$, i.e., $P = \{p_c\} \cup P$;
13: **if** all operations $o_{j.i}$ of $O_{\max}$ belong to a sub-job **then**
14: Let $o_{j.i}$ and $o_{j.i'}$ be the first and last operations of this sub-job, respectively;
15: The arcs $(p_c, t_{j.i})$ and $(t_{j.i'}, p_c)$ are designed;
16: **else**
17: **for** all operations $o_{j.i}$ of $O_{\max}$ **do**
18: The arcs $(p_c, t_{j.i})$ and $(t_{j.i}, p_c)$ are designed;
19: **end for**
20: **end if**
21: **end for**

3) there are ten transitions $t_{1.1}$, $t_{1.2}$, $t_{1.3}$, $t_{1.4}$, $t_{1.5}$, $t_{2.1}$, $t_{2.2}$, $t_{2.3}$, $t_{2.4}$ and $t_{2.5}$ that represent the execution of $o_{1.1}$, $o_{1.2}$, $o_{1.3}$, $o_{1.4}$, $o_{1.5}$, $o_{2.1}$, $o_{2.2}$, $o_{2.3}$, $o_{2.4}$ and $o_{2.5}$, respectively.

The operation-resource diagram is designed by Algorithm 2. The operation-conflict graph G_{OC} is designed by Algorithm 3. Based on the operation-conflict graph in Fig. 4, and Steps 8–9 of Algorithm 4, all maximal conflict operation sets are computed, which are $\{o_{1.1}, o_{1.2}, o_{1.3}\}$, $\{o_{1.3}, o_{1.4}, o_{1.5}\}$, $\{o_{2.1}, o_{2.2}, o_{2.3}\}$, $\{o_{2.3}, o_{2.4}, o_{2.5}\}$, $\{o_{1.1}, o_{2.1}\}$, $\{o_{1.3}, o_{2.3}\}$.

Taking $\{o_{1.1}, o_{1.2}, o_{1.3}\}$ as an example, its linear constraint is formulated as: $\boldsymbol{m}(p_{1.1}) + \boldsymbol{m}(p_{1.2}) + \boldsymbol{m}(p_{1.3}) \leq 1$, and its monitor place p_{c1} is designed. $o_{1.1}$ and $o_{1.3}$ are the first and

last operations of this sub-job, respectively. Arcs $(p_{c1}, t_{1.1})$ and $(t_{1.3}, p_{c1})$ are designed by Steps 12–20. All monitor places and corresponding arcs are designed by Steps 12–20, which are $p_{c1}, p_{c2}, \ldots, p_{c6}$.

Finally, the plant Petri net of the batch production system is obtained, as shown in Fig. 5.

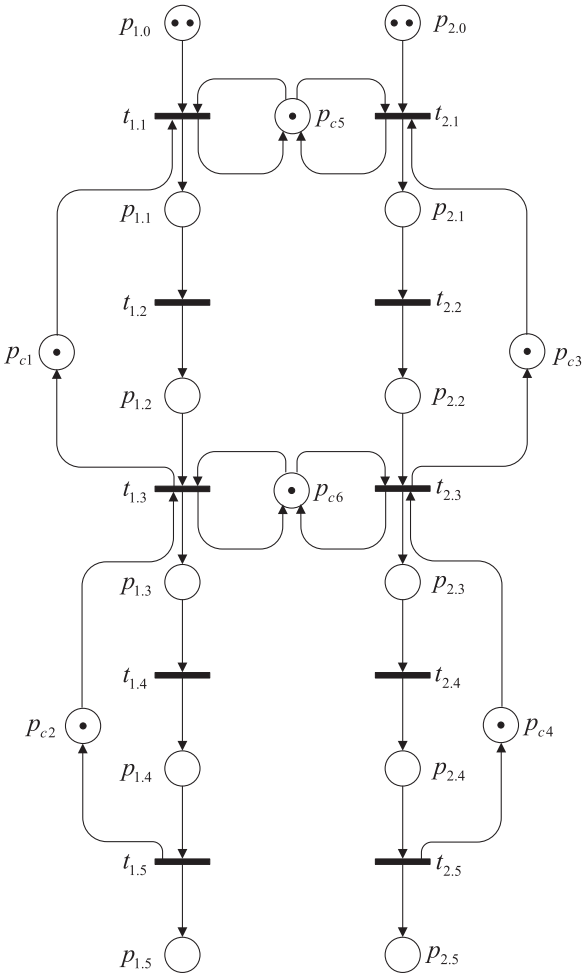

FIGURE 5. Plant Petri net model of the batch production system.

V. A NEW VARIANT OF FILTERED BEAM SEARCH BASED ON D-TERG

A. OPTIMAL SCHEDULE

Scheduling problems for systems modeled by a given T-TPN can be solved thanks to the TERG by reformulating the problem as the usual problem of searching for the shortest path in a weighted graph. For this purpose, let us first observe that, in the proposed T-TPN model, the reference marking $\boldsymbol{m}_{ref}$ is a deadlock (if it is not the case, one can add a counting place p_{ref} to the model to deadlock this marking). Then, introduce $S_{ref} = (\boldsymbol{m}_{ref}, \emptyset)$ as the unique reference state. The optimal solution of the scheduling problem is obtained by searching the shortest path in duration from the initial state S_0 to the reference state S_{ref}, and it can be formalized as the solution of (4):

$$ph^*(S_0, S_{ref}) = \underset{ph \in PH_E(S_0, S_{ref})}{\arg\min} \{d(ph)\}. \quad (4)$$

The determination of the optimal path $ph^*(S_0, S_{ref})$ for any state $S \in \boldsymbol{S_E}$ can be obtained with global optimization algorithms [39], for example the well-known Dijkstra algorithm. Observe that such a method requires the computation of the complete TERG. One important problem with such a computation is that the size of the graph may become rapidly large (the increase in size is a critical limitation for all variants of state class graphs that are based on time). Intuitively, the price to pay, when $\boldsymbol{S_E}$ is used instead of R, depends on the number of different earliest firing times that may occur for a given marking. In some particular cases, the state space becomes infinite even if the untimed net has very few reachable markings. Consequently, to obtain an applicable and tractable approach it is necessary to ensure that the number of explored states remains finite and as small as possible.

Note first that the infinite expansion of $\boldsymbol{S_E}$ can be avoided by formulating all earliest firing times as multiples of a given period dt. More precisely, each transition t is associated with an earliest firing time $D(t) = \alpha(t) \cdot dt$ and $\alpha(t) \in \mathbb{N}^+$ ($\mathbb{N}^+$ is the set of all positive integers) then $\boldsymbol{S_E}$ is of finite cardinality N_E that satisfies $N < N_E < N \cdot (\alpha_m + 1)^{k \cdot q}$ with $\alpha_m = \max\{\alpha(t), t \in T\}$ [38]. In that case, the number of states is bounded even if it may be large. Note that this simplification is reasonable from a practical point of view because any real number can be approximated by a rational number with a given precision (i.e., the multiple of a given period dt that depends on the precision). Then, the computation complexity to design the complete graph of the TERG can be measured by the ratio N_E/N that equals at least 1.

In order to make the method more tractable, we propose to expand only the part of the TERG that will be required to design the expected schedule by pruning the non promising branches of the graph. We refer to this subgraph as a dynamical TERG (D-TERG).

B. DYNAMICAL TERG

Let us consider a TERG $(\boldsymbol{S_E}, \Omega_E, B_E, S_0)$ and a sequence of K states $\Sigma = S(1) \ldots S(K)$, $S(k) \in \boldsymbol{S_E}$ with $S(1) = S_0$ and $S(k) = \boldsymbol{\{S(1), \ldots, S(k-1)\}^\bullet}$ (i.e. each state $S(k)$ has a predecessor within $\{S(1), \ldots, S(k-1)\}$). The D-TERG (Σ) obtained for $(\boldsymbol{S_E}, \Omega_E, B_E, S_0)$ and Σ is defined as $(\boldsymbol{S_\Sigma}, \Omega_\Sigma, B_\Sigma, S_0)$ with

- $\boldsymbol{S_\Sigma} \subseteq \boldsymbol{S_E}$ is a subset of N_Σ states of the TERG $(\boldsymbol{S_E}, \Omega_E, B_E, S_0)$. Observe that $N_\Sigma \leq N_E, N_\Sigma \geq K$ and that $\boldsymbol{S_\Sigma}$ is composed of the states $S(k)$, $k = 1, \ldots, K$ and their successors,
- $\Omega_\Sigma \in (T \cup \{\varepsilon\})^{N_\Sigma \times N_\Sigma}$ is the labeled adjacency matrix of the graph computed in a similar way as Ω_E,
- $B_\Sigma \in (\mathbb{R}^+)^{N_\Sigma \times N_\Sigma}$ is the earliest firing time matrix of the graph computed in a similar way as B_E,
- S_0 is the initial state.

The following remarks hold

- as far as each state $S(k), k = 1, \ldots, K$ has a predecessor within $\{S(1), \ldots, S(k-1)\}$, there exists at least one path from S_0 to $S(k)$. Consequently, the D-TERG can be viewed as a subgraph of the TERG,
- the D-TERG is completly defined by the sequence Σ.

C. FILTERED BEAM SEARCH METHOD FOR T-TPN

The filtered beam search (FBS) is an aggressive and efficient pruning method based on original beam search (BS) [30]. The different variants of BS algorithms expand only a predefined number of nodes (the beam width β_g) at each iteration according to a cost function that will be discussed in the next section. The FBS is an improvement of the basic BS that combines the global filter β_g with a local filter β_l that allows the expansion of up to β_l nodes from any parent node [30]. In the FBS method, the selection of nodes to expand is based on a cost function f composed of the actual cost g from the initial node to the current node, and an estimate h of the remaining cost to the reference node. The search uses a list containing at most β_g nodes ranked with the cost function f. The search stops when one or several solutions are found with a certain cost g^* and when all candidates in the list satisfy $g > g^*$. The search principle of FBS as explained in [30], [42] is illustrated in Fig. 6 for parameters $\beta_g = 3$ and $\beta_l = 2$, and the explored nodes are colored in grey. At step 1, the list of nodes contains only the initial state 1. At step 2, the list is limited by β_l and only 2 nodes are selected: 3 and 5. At step 3, the list of nodes is limited by β_g and 3 nodes are selected: 7, 8 and 10. Finally at step 4, the list of nodes is limited by both parameters β_l and β_g and is computed as 12, 13 and 16. The search continues up to the verification of the stopping criteria.

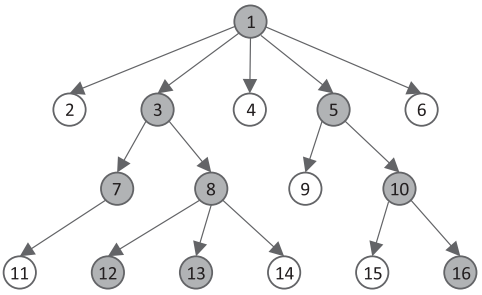

FIGURE 6. Filtered beam search principl.

The core of the beam search approaches is the definition of the heuristic function h that is used to select at each step the next node to be expanded. To be efficient, this function should have two important properties:

- in order to ensure that the algorithm converges to a solution, h should never overestimate the true cost to the reference,
- in order to avoid to remove promising candidates, the difference between the true cost and its estimation computed by h should be as small as possible.

In [40], [41], the heuristic function h is formally defined for each intermediate marking $\boldsymbol{m}$ of a given trajectory $(\sigma, \boldsymbol{m}_0)$ by dividing $(\sigma, \boldsymbol{m}_0)$ in two parts: the already computed trajectory $(\sigma_1, \boldsymbol{m}_0)$ from $\boldsymbol{m}_0$ to $\boldsymbol{m}$ (i.e. $\boldsymbol{m}_0[\sigma_1\rangle\boldsymbol{m}$) and the residual trajectory $(\sigma_2, \boldsymbol{m})$ with unknown sequence σ_2 from $\boldsymbol{m}$ to $\boldsymbol{m}_{ref}$ (i.e. $\boldsymbol{m}[\sigma_2\rangle\boldsymbol{m}_{ref}$). The cost function may be rewritten as (5):

$$f(\sigma_1, \boldsymbol{m}_0, \boldsymbol{m}_{ref}) = g(\sigma_1, \boldsymbol{m}_0) + h(\boldsymbol{m}, \boldsymbol{m}_{ref}) \tag{5}$$

The function $g(\sigma_1, \boldsymbol{m}_0)$ gives the cost, i.e., the duration in our case, of the already computed trajectory $(\sigma_1, \boldsymbol{m}_0)$ and the function $h(\boldsymbol{m}, \boldsymbol{m}_{ref})$ approximates the residual duration of $(\sigma_2, \boldsymbol{m})$.

For the part of the trajectory $(\sigma_1, \boldsymbol{m}_0)$ already computed, the method in [40] computes the duration $g(\sigma_1, \boldsymbol{m}_0)$ of $(\sigma_1, \boldsymbol{m}_0)$ by transforming any untimed trajectory into a timed one. This algorithm uses the chronological firing order of the transitions in σ_1 and the earliest firing policy to update, at each new marking $\boldsymbol{m}$, the residual durations of the transitions enabled at $\boldsymbol{m}$. These transitions and their residual durations are stored in a calendar that is updated at each intermediate marking of the trajectory. Observe that the transformation previously mentioned and the use of the calendar are required because the usual method is based on untimed trajectories and adds a posteriori timing information. In this paper, on the contrary, we derive directly $g(\sigma_1, \boldsymbol{m}_0)$ from the D-TERG.

For the unknown part of the trajectory, several estimations $h(\boldsymbol{m}, \boldsymbol{m}_{ref})$ of the residual duration from the current marking $\boldsymbol{m}$ to the reference $\boldsymbol{m}_{ref}$ have been studied [30], [42]. In particular, Luo *et al.* in [41] propose a heuristic function based on the search of resource and operational places. Lefebvre proposes an estimation based on residual firing count vector [43]. The previous methods are based on the following schema:

- search for the paths that exist in the T-TPN structure from all places that have at least one token at marking $\boldsymbol{m}$,
- compute the durations of these paths with Eq. (2),
- estimate the residual duration to the reference marking with Eq. (6):

$$h(\boldsymbol{m}, \boldsymbol{m}_{ref}) = \max_{p \in P(\boldsymbol{m})} \{ \min_{ph \in PH(p, p_{ref})} \{d(ph) - D(t_1)\}\} \tag{6}$$

where t_1 is the first transition in ph. In this paper, we refine the heuristic function h based on the D-TERG.

D. FILTERED BEAM SEARCH BASED ON D-TERG

Based on D-TERG(Σ) as previously defined, the cost function defined in Eq. (5) is reformulated for any state $S \in \boldsymbol{S}_\Sigma$ as:

$$f(S_0, S, S_{ref}, \Sigma) = g(S_0, S, \Sigma) + h(S, S_{ref}) \tag{7}$$

The function $g(S_0, S, \Sigma)$ gives the minimal duration of the paths from S_0 to S, and the function $h(S, S_{ref})$ estimates the residual duration from state S to state S_{ref}. Note that $g(S_0, S, \Sigma)$ depends on Σ whereas $h(S, S_{ref})$ does not.

Let us first detail the computation of $g(S_0, S, \Sigma)$. According to the definition of D-TERG(Σ), there at least exists a path

from S_0 to S. Consequently, $g(S_0, S, \Sigma)$ results from Eq. (3) and is computed directly by Eq. (8).

$$g(S_0, S, \Sigma) = \min_{ph \in PH_\Sigma(S_0,S)} \{ \sum_{\substack{S(k)\in ph,\\ S(k)\neq S}} B_\Sigma(S(k), S(k+1))\}, \quad (8)$$

where $PH_\Sigma(S, S')$ is the set of paths from state S_0 to those S in D-TERG(Σ). Observe that all paths that exist in TERG from S_0 to S are not in general reported in D-TERG(Σ) (as far as $N_\Sigma < N_E$). For this reason, depending on Σ, the final results obtained with the combined use of the D-TERG and the FBS is not, in general, the optimal schedule. But, compared with the method in [40], the use of D-TERG has two advantages:

- once D-TERG(Σ) is computed, $g(S_0, S, \Sigma)$ becomes easy to compute by Eq. (8) because timing information is included in D-TERG(Σ), and there is no need to use the transformation of untimed firing sequence into timed sequence as proposed in [40],
- more important, all timing information available at state S is also explicitly encoded in $\Delta(S)$ and will be used to compute $h(S, S_{ref})$.

Let us compute $h(S, S_{ref})$. Observe that Eq. (6) can be reformulated in D-TERG as $h(S, S_{ref}) = h(\boldsymbol{m}(S), \boldsymbol{m}_{ref})$. Two corrections are proposed to refine the estimation h.

- if the first transition $t_1 \in ph(p, p_{ref})$ is enabled at S, the residual time $\delta(S, t_1)$ of t_1 is the earliest firing time of t_1 in state S. Consequently this time should be added to the estimation $h(S, S_{ref})$,
- if the first transition $t_1 \in ph(p, p_{ref})$ is not enabled at S (this could occur if the resources needed to fire t_1 are currently used for another operation), an extra waiting time $WT(S, t_1)$ before enabling t_1 should be considered. $WT(S, t_1)$ depends on the enabled transitions t in conflict with t_1:

$$WT(S, t_1) = \max_{\substack{t\in \boldsymbol{m}(S)^\bullet, (\delta,t)\in\Delta(S)\\ {}^\bullet t\cap {}^\bullet t_1\neq\emptyset, t\neq t_1}} \{\delta\}. \quad (9)$$

Consequently $h(S, S_{ref})$ is refined as:

$$h(S, S_{ref}) = \max_{p\in P(\boldsymbol{m}(S))} \{ \min_{ph\in PH(p,p_{ref})} \{d(ph) - D(t_1) + \delta(S, t_1) + WT(S, t_1)\}\}. \quad (10)$$

where t_1 is the first transition in ph. The heuristic function $h(S, S_{ref})$ never overestimate the true duration to the reference.

VI. APPLICATION TO A CASE STUDY

In this section, we aim to schedule the operations of the batch production system in order to minimize the total duration $C_{\max}$ of the operations according to the T-TPN model of the system, the D-TERG and the new variant of FBS introduced in the previous section.

Dijkstra algorithm, FBS algorithm [40], [41], and a variant of FBS based on D-TERG in this paper are applied to the case study. For these methods, solutions and performance are discussed with respect to the initial number of products to be processed $\boldsymbol{m}_0(p_{1.0}) = \boldsymbol{m}_0(p_{2.0}) = \rho(J_1) = \rho(J_2) = k$. Table 2 sums up the results obtained for Dijkstra algorithm based on the computation of the TERG. For this purpose, a period $g^* = 10$ minutes has been considered. In this table we report the number of markings in the usual reachability set $R(\boldsymbol{m}_0)$ (i.e., the size of $R(\boldsymbol{m}_0)$ denoted by N), the number of states in TERG (i.e., the size of $\boldsymbol{S_E}$ denoted by N_E), the average number of states in TERG for one batch (i.e., N_E/k), the size (as a number of firings) of the control sequence that is found, the corresponding makespan $C_{\max}$ and the computation time required to obtain the solution with an Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz-2.80 GHz. In particular for $k = 1$, the solution σ_1 is found using Dijkstra algorithm in TERG:

$$\sigma_1 = (t_6, 30)(t_1, 50)(t_7, 70)(t_2, 80)(t_8, 110) (t_3, 140)(t_9, 160)(t_4, 180)(t_{10}, 220)(t_5, 220)$$

One can notice that Dijkstra algorithm fails rapidly when k increases due to the numerical complexity (we limit the search for TERG with a maximal size of 15,000 states). On the contrary, when a solution is found, it is of minimal duration.

Table 3 sums up the result obtained for local optimization based on the use of FBS with parameters $\beta_g = 20$ and $\beta_l = 20$. In this table, the number of markings expanded by FBS is also reported, denoted by N'. In particular for $k = 1$, two solutions σ_2 and σ_3 are found:

$$\sigma_2 = (t_1, 20)(t_2, 50)(t_6, 50)(t_3, 80)(t_7, 90) (t_4, 120)(t_8, 130)(t_5, 160)(t_9, 180)(t_{10}, 220)$$
$$\sigma_3 = (t_6, 30)(t_7, 70)(t_1, 70)(t_8, 110)(t_2, 110) (t_3, 140)(t_9, 160)(t_4, 180)(t_{10}, 220)(t_5, 220)$$

Table 4 sums up the result obtained by the proposed variant of FBS based on D-TERG with parameters $\beta_g = 20$ and $\beta_l = 20$. The number of states in D-TERG (i.e., the size of $\boldsymbol{S_\Sigma}$), denoted by N_Σ, is reported in Table 4. In particular for $k = 1$, the solution σ_4 is found using the proposed variant

TABLE 2. Results of the global optimization with Dijkstra algorithm and TER.

k	1	2	3
N	36	441	3839
N_E	86	4689	> 15000
N_E/k	86	2344.5	>5000
Size of the solution	20	40	Not found
$C_{\max}$	220 min.	370 min.	Not found
Computation time	< 1 sec.	120 sec.	Not found

TABLE 3. Results of FBS for $\beta_g = \beta_l = 20$.

k	1	2	3	5	10	20
$C_{\max}$	220 min.	380 min.	520 min.	980 min.	1920 min.	4290 min.
$C_{\max}/k$	220 min.	190 min.	173 min.	196 min.	192 min.	214 min.
N'	36	214	413	789	1722	3595
N'/k	36	107	137.7	157.8	172.2	179.75
Computation time	< 1 sec.	2 sec.	4 sec.	10 sec.	21 sec.	102 sec.

TABLE 4. Results of a variant of FBS based on D-TERG for $\beta_g = \beta_l = 20$.

k	1	2	3	5	10	20
$C_{\max}$	220 min.	370 min.	540 min.	880 min.	1670 min.	3310 min.
$C_{\max}/k$	220 min.	185 min.	180 min.	176 min.	167 min.	165.5 min.
N_Σ	86	297	476	851	1759	3456
N_Σ/k	86	148.5	158.7	170.2	175.9	172.8
Computation time	<1 sec.	2 sec.	5 sec.	12 sec.	26 sec.	107 sec.

of FBS:

$$\sigma_4 = (t_6, 30)(t_1, 50)(t_7, 70)(t_2, 80)(t_8, 110)$$
$$(t_3, 140)(t_9, 160)(t_4, 180)(t_5, 220)(t_{10}, 220)$$

The size of D-TERG in Table 4 is 476 that is less than the size of TERG in Table 2 when $k = 3$. The size of D-TERG is 3456 when $k = 20$, it is still much less than 15000. One can notice that FBS and the variant of FBS based on D-TERG find always a solution when k increases but the optimality cannot be ensured. It is obvious that $C_{\max}$ reflects the performance of solutions obtained by different algorithms, and that N_E, N' and N_Σ reflect the complexity of the different algorithms. However, these indicators strongly depends on k. In order to obtain performance and complexity indicators only depending on the Petri net structure and on the specific algorithm, but not on k, we further compute $C_{\max}/k$, N'/k and N_Σ/k in Tables 3 and 4. Observe that values of $C_{\max}/k$, N'/k and N_Σ/k converge to asymptotic values as k increases. It is better to use $C_{\max}/k$, N'/k and N_Σ/k as the performance and complexity indicators, respectively. From Tables 3 and 4, we can find that the proposed variant of FBS has better performance indicators (i.e., $C_{\max}/k$) than classical FBS. At the same time, classical FBS and the new variant of FBS have better complexity indicators (i.e., N'/k and N_Σ/k) than Dijkstra algorithm. Table 5 provides the duration of the solution found with respect to k as the required computational time to compute the solution for larger values of k.

TABLE 5. Results of a variant of FBS based on D-TERG for $\beta_g = \beta_l = 5$.

k	1	5	20	50	100	200
$C_{\max}$	220 min.	910 min.	3390 min.	8290 min.	16500 min.	32790 min.
$C_{\max}/k$	220 min.	182 min.	169.5 min.	168.5 min.	165 min.	163.95 min.
N_Σ	43	231	920	2300	4817	9200
N_Σ/k	43	46.1	46	46	48.2	46
Computation time	<1 sec.	2 sec.	13 sec.	57 sec.	145 sec.	466 sec.

Note that the performance of the variant of FBS based on D-TERG also depends on the parameters β_g and β_l. By decreasing β_g and β_l, the size of D-TERG also decreases as the computation time, but the makespan increases. In particular, using a small number of global or local beams can lead to degraded performances: one can notice from Tables 4 and 5 that the makespan obtained for $k = 5$ is 880 min. when $\beta_g = \beta_l = 20$ in Table 4 and increases to 910 min. when β_g and β_l decrease to 5 in Table 5. This is a typical characteristic of the beam search method: The performance is obviously better when numerous candidates are expanded (i.e. large values of β_g and β_l) but rapidity is penalized, whereas performance is weaker when only few candidates are expanded (i.e. small values of β_g and β_l) but rapidity is improved.

To further illustrate the application and complexity of our method, we apply it to a more complex batch process with more jobs. This example is inspired by the model proposed in [9]. The model in [9] is first transformed and expanded into a T-TPN model shown in Fig. 7 with $T = \{t_1, t_2, \ldots, t_{15}\}$, $P = \{p_1, p_2, \ldots, p_{23}\}$, $\boldsymbol{m}_0 = (1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,1,1)^T$. The T-TPN models a batch production system with five jobs J_1, J_2, J_3, J_4, and J_5. Transitions t_1 to t_3, t_4 to t_6, t_7 to t_9, t_{10} to t_{12}, and t_{13} to t_{15} define J_1, J_2, J_3, J_4, and J_5, respectively. Minimal durations of transitions are shown in Table 6.

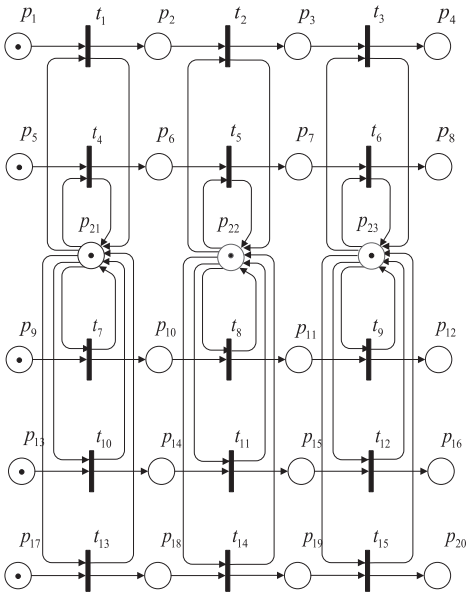

FIGURE 7. Petri net model of a variant of the batch production system in [9].

TABLE 6. Minimal durations of transitions of Fig. 7.

t_j	t_1, t_6, t_7, t_{11}	t_2	t_3	t_4, t_{13}	t_5	t_8, t_{14}	t_9	t_{10}	t_{12}, t_{15}
$D(t_j)$	3.5	4.3	8.7	4	5.5	7.5	6	12	8

Table 7 sums up the results obtained for Dijkstra algorithm, FBS algorithm, and the new variant of FBS when r ($r = 2, 3, 4, 5$) jobs are considered. In this table we report $C_{\max}$, $C_{\max}/N_{Pn}$, N_E, N', N_Σ, N_{Pn} and so on, where N_{Pn} denotes the sum of numbers of places, transitions and arcs of the Petri net for the first r jobs.

Similar to k in Tables 2, 3, and 4, r affects $C_{\max}$, N_{Pn}, N_E, N' and N_Σ, in Table 7. Normalized indicators $C_{\max}/N_{Pn}$, N_E/N_{Pn}, N'/N_{Pn} and N_Σ/N_{Pn} tend to asymptotic values and may be used as the performance and complexity indicators, respectively. The distribution of these indicators for different algorithms is shown in Fig. 8. We can find that the distribution of the performance and complexity indicators of the new variant of FBS tend to smaller values, and it has better

TABLE 7. Results of different algorithms for the first *r* jobs of Fig. 7.

Dijkstra algorithm				
r	2	3	4	5
N_{Pn}	41	60	79	98
$C_{\max}$	20	26	34	Not found
$C_{\max}/N_{Pn}$	0.49	0.43	0.43	Not found
N_E	20	158	2484	>15000
N_E/N_{Pn}	0.5	2.6	31.4	Not found
FBS algorithm				
r	2	3	4	5
N_{Pn}	41	60	79	98
$C_{\max}$	20	26	53.2	61.5
$C_{\max}/N_{Pn}$	0.49	0.43	0.67	0.63
N'	16	64	169	242
N'/N_{Pn}	0.4	1.1	2.1	2.5
The new variant of FBS				
r	2	3	4	5
N_{Pn}	41	60	79	98
$C_{\max}$	20	26	34	42
$C_{\max}/N_{Pn}$	0.49	0.43	0.43	0.43
N_Σ	20	109	186	249
N_Σ/N_{Pn}	0.5	1.8	2.4	2.5

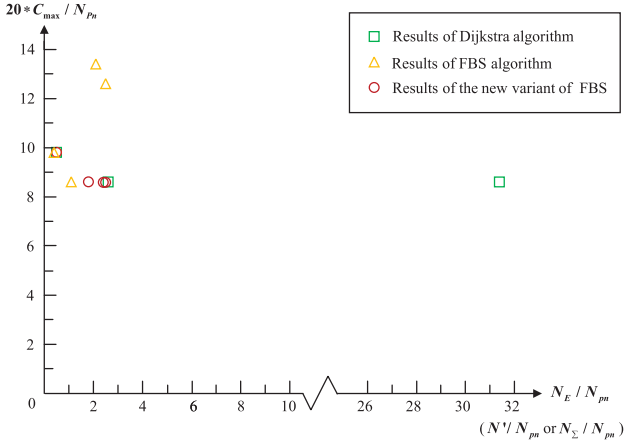

FIGURE 8. The distribution of performance and complexity indicators in Table 7.

performance and complexity than the other two approaches. This verifies that the new variant of FBS is suitable for solving the scheduling problem of complex batch production systems.

VII. CONCLUSION

In this paper, modeling and scheduling methods for batch production systems are proposed. The approach is based on T-TPNs and a new variant of FBS methods. A method to model a batch production system with multiple resource units as T-TPN with conflicts resulting from the use of resource units is proposed. Then a D-TERG is presented according to a sequence of states that are successively explored for the schedule issue. Based on T-TPNs and D-TERG, a variant of FBS is designed with an improved heuristic cost function to obtain optimal or suboptimal solutions. The application of this method to the batch production systems illustrates the proposed scheduling approach.

In the next work, fault diagnosis such as equipment failure and deadlock control will be considered for the scheduling issue of production systems and multi-agent systems [44]–[48]. In addition, maximal time constraints will be also introduced to consider chemical reactions that should respect maximal processing times.

REFERENCES

[1] R. Zurawski and M. Zhou, "Petri nets and industrial applications: A tutorial," *IEEE Trans. Ind. Electron.*, vol. 41, no. 6, pp. 567–583, Dec. 1994.
[2] M. Tittus and K. Åkesson, "Petri net models in batch control," *Math. Comput. Model. Dyn. Syst.*, vol. 5, no. 2, pp. 113–132, Jun. 1999.
[3] P. Falkman, B. Lennartson, and M. Tittus, "Specification of a batch plant using process algebra and Petri nets," *Control Eng. Pract.*, vol. 17, no. 9, pp. 1004–1015, Sep. 2009.
[4] L. Ferrarini and L. Piroddi, "Modeling and control of fluid transportation operations in production plants with Petri nets," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 5, pp. 1090–1098, Sep. 2008.
[5] N. Wu, M. Zhou, and Z. Li, "Short-term scheduling of crude-oil operations: Enhancement of crude-oil operations scheduling using a Petri net-based control-theoretic approach," *IEEE Robot. Autom. Mag.*, vol. 22, no. 2, pp. 64–76, Jun. 2015.
[6] N. Wu, L. Bai, and M. Zhou, "An efficient scheduling method for crude oil operations in refinery with crude oil type mixing requirements," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 3, pp. 413–426, Mar. 2016.
[7] C. Cassandras, "Discrete event systems: Modeling and performances analysis," *IEEE Trans. Autom. Control*, vol. 14, nos. 1–2, pp. 57–78, 1993.
[8] G. J. Tsinarakis, N. C. Tsourveloudis, and K. P. Valavanis, "Modular Petri net based modeling, analysis, synthesis and performance evaluation of random topology dedicated production systems," *J. Intell. Manuf.*, vol. 16, no. 1, pp. 67–92, Feb. 2005.
[9] M. Ghaeli, P. A. Bahri, P. Lee, and T. Gu, "Petri-net based formulation and algorithm for short-term scheduling of batch plants," *Comput. Chem. Eng.*, vol. 29, no. 2, pp. 249–259, Jan. 2005.
[10] O. T. Baruwa, M. A. Piera, and A. Guasch, "Deadlock-free scheduling method for flexible manufacturing systems based on timed colored Petri nets and anytime heuristic search," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 5, pp. 831–846, May 2015.
[11] F. Yang, N. Wu, Y. Qiao, and R. Su, "Polynomial approach to optimal one-wafer cyclic scheduling of treelike hybrid multi-cluster tools via Petri nets," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 1, pp. 270–280, Jan. 2018.
[12] J. Zhou, J. Wang, and J. Wang, "A simulation engine for stochastic timed Petri nets and application to emergency healthcare systems," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 4, pp. 969–980, Jul. 2019.
[13] P. Merlin and D. Faber, "Recoverability of communication protocols—Implications of a theoretical study," *IEEE Trans. Commun.*, vol. 24, no. 9, pp. 1036–1043, Sep. 1976.
[14] B. Berthomieu and M. Diaz, "Modeling and verification of time dependent systems using time Petri nets," *IEEE Trans. Softw. Eng.*, vol. 17, no. 3, pp. 259–273, Mar. 1991.
[15] K. Klai, N. Aber, and L. Petrucci, "A new approach to abstract reachability state space of time Petri nets," in *Proc. 20th Int. Symp. Temporal Represent. Reasoning*, 2013, pp. 117–124.
[16] F. Basile, M. P. Cabasino, and C. Seatzu, "State estimation and fault diagnosis of labeled time Petri net systems with unobservable transitions," *IEEE Trans. Autom. Control*, vol. 60, no. 4, pp. 997–1009, Apr. 2015.
[17] L. Pan, B. Yang, J. Jiang, and M. Zhou, "A time Petri net with relaxed mixed semantics for schedulability analysis of flexible manufacturing systems," *IEEE Access*, vol. 8, pp. 46480–46492, 2020.
[18] P. Heidari and H. Boucheneb, "Controller synthesis of time Petri nets using stopwatch," *J. Eng.*, vol. 2013, Apr. 2013, Art. no. 970487.
[19] D. Lefebvre and C. Daoui, "Control design for bounded partially controlled TPNs using timed extended reachability graphs and MDP," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 6, pp. 2273–2283, Jun. 2020.
[20] D. Yong Lee and F. DiCesare, "Scheduling flexible manufacturing systems using Petri nets and heuristic search," *IEEE Trans. Robot. Autom.*, vol. 10, no. 2, pp. 123–132, Apr. 1994.
[21] H. Henry Xiong and M. Zhou, "Scheduling of semiconductor test facility via Petri nets and hybrid heuristic search," *IEEE Trans. Semicond. Manuf.*, vol. 11, no. 3, pp. 384–393, Aug. 1998.
[22] G. Mejía and N. G. Odrey, "An approach using Petri nets and improved heuristic search for manufacturing system scheduling," *J. Manuf. Syst.*, vol. 24, no. 2, pp. 79–92, Jan. 2005.
[23] C. Li, W. Wu, Y. Feng, and G. Rong, "Scheduling FMS problems with heuristic search function and transition-timed Petri nets," *J. Intell. Manuf.*, vol. 26, no. 5, pp. 933–944, Oct. 2015.
[24] B. Huang, R. Jiang, and G. Zhang, "Search strategy for scheduling flexible manufacturing systems simultaneously using admissible heuristic functions and nonadmissible heuristic functions," *Comput. Ind. Eng.*, vol. 71, pp. 21–26, May 2014.

[25] K. Xing, L. Han, M. Zhou, and F. Wang, "Deadlock-free genetic scheduling algorithm for automated manufacturing systems based on deadlock control policy," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 3, pp. 603–615, Jun. 2012.
[26] J. Luo, Z. Liu, M. Zhou, and K. Xing, "Deadlock-free scheduling of flexible assembly systems based on Petri nets and local search," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Sep. 10, 2018, doi: 10.1109/TSMC.2018.2855685.
[27] W. Zhang, T. Freiheit, and H. Yang, "Dynamic scheduling in flexible assembly system based on timed Petri nets model," *Robot. Comput.-Integr. Manuf.*, vol. 21, no. 6, pp. 550–558, Dec. 2005.
[28] P. Ow and T. Morton, "Filtered beam search in scheduling," *Int. J. Prod. Res.*, vol. 26, no. 1, pp. 35–62, 1988.
[29] G. Mejía, K. Niño, C. Montoya, M. A. Sánchez, J. Palacios, and L. Amodeo, "A Petri net-based framework for realistic project management and scheduling: An application in animation and videogames," *Comput. Oper. Res.*, vol. 66, pp. 190–198, Feb. 2016.
[30] G. Mejía and K. Niño, "A new hybrid filtered beam search algorithm for deadlock-free scheduling of flexible manufacturing systems using Petri nets," *Comput. Ind. Eng.*, vol. 108, pp. 165–176, Jun. 2017.
[31] H. C. Johnston, "Cliques of a graph-variations on the Bron–Kerbosch algorithm," *Int. J. Comput. Inf. Sci.*, vol. 5, no. 3, pp. 209–238, Sep. 1976.
[32] A. Giua, F. DiCesare, and M. Silva, "Generalized mutual exclusion constraints on nets with uncontrollable transitions," in *Proc. IEEE Int. Conf. SMC*, Chicago, IL, USA, Oct. 1992, pp. 974–979.
[33] K. Yamalidou, J. Moody, M. Lemmon, and P. Antsaklis, "Feedback control of Petri nets based on place invariants," *Automatica*, vol. 32, no. 1, pp. 15–28, Jan. 1996.
[34] J. Luo, H. Ni, W. Wu, S. Wang, and M. Zhou, "Simultaneous reduction of Petri nets and linear constraints for efficient supervisor synthesis," *IEEE Trans. Autom. Control*, vol. 60, no. 1, pp. 88–103, Jan. 2015.
[35] J. Luo and M. Zhou, "Petri-net controller synthesis for partially controllable and observable discrete event systems," *IEEE Trans. Autom. Control*, vol. 62, no. 3, pp. 1301–1313, Jun. 2017.
[36] J. Luo, Q. Zhang, X. Chen, and M. Zhou, "Modeling and race detection of ladder diagrams via ordinary Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 7, pp. 1166–1176, Jul. 2018.
[37] J. Luo, W. Wu, M. Zhou, H. Shao, K. Nonami, and H. Su, "Structural controller for logical expression of linear constraints on Petri nets," *IEEE Trans. Autom. Control*, vol. 65, no. 1, pp. 397–403, Jan. 2020.
[38] D. Lefebvre, "Approximated timed reachability graphs for the robust control of discrete event systems," *Discrete Event Dyn. Syst.*, vol. 29, no. 1, pp. 31–56, Mar. 2019.
[39] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2001.
[40] D. Lefebvre, "Dynamical scheduling and robust control in uncertain environments with Petri nets for DESs," *Processes*, vol. 5, no. 4, p. 54, Oct. 2017, doi: 10.3390/pr5040054.
[41] J. Luo, K. Xing, M. Zhou, X. Li, and X. Wang, "Deadlock-free scheduling of automated manufacturing systems using Petri nets and hybrid heuristic search," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 3, pp. 530–541, Mar. 2015.
[42] D. Lefebvre and G. Mejia, "Robust scheduling in uncertain environment with Petri nets and beam search," in *Proc. IEEE INCOM*, Bergamo, Italy, 2018, pp. 1077–1082.
[43] D. Lefebvre, "Approaching minimal time control sequences for timed Petri nets," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 1215–1221, Apr. 2016.
[44] L. Yang, Z. Yu, M. A. El-Meligy, A. M. El-Sherbeeny, and N. Wu, "On multiplexity-aware influence spread in social networks," *IEEE Access*, vol. 8, pp. 106705–106713, 2020, doi: 10.1109/ACCESS.2020.2999312.
[45] X. Li, Z. Yu, Z. Li, and N, Wu, "Group consensus via pinning control for a class of heterogeneous multi-agent systems with input constraints," *Inf. Sci.*, vol. 542, pp. 247–262, 2021, doi: 10.1016/j.ins.2020.05.085.
[46] S. Zhou, Z. Yu, E. S. A. Nasr, H. A. Mahmoud, E. M. Awwad, and N. Wu, "Homomorphic encryption of supervisory control systems using automata," *IEEE Access*, vol. 8, pp. 147185–147198, 2020, doi: 10.1109/ACCESS.2020.3014217.
[47] Q. Chen, L. Yin, N. Wu, M. El-Meligy, M. Sharaf, and Z. Li, "Diagnosability of vector discrete-event systems using predicates," *IEEE Access*, vol. 7, pp. 147143–147155, 2019.
[48] D. Sun, Y. Chen, M. Chen, M. El-Meligy, M. Sharaf, N. Wu, and Z. Li, "On algebraic identification of critical states for deadlock control in automated manufacturing systems modeled with Petri nets," *IEEE Access*, vol. 7, pp. 121332–121349, 2019.

JIAZHONG ZHOU received the M.S. degree from the Institute of Information Science and Engineering, Huaqiao University, Xiamen, China, in 2016. He is currently pursuing the Ph.D. degree in control theory and engineering with Xidian University, Xi'an, China. His research interests include Petri net theory and applications, the scheduling of discrete event systems, and the space abstraction of time Petri nets.

JILIANG LUO (Senior Member, IEEE) received the B.E. and M.S. degrees in thermal power engineering from Northeast Dianli University, Jilin, China, in 2000 and 2003, respectively, and the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2006. He joined Huaqiao University, Xiamen, China, in 2006. He was a Visiting Researcher with Chiba University, from 2009 to 2010, and also with the New Jersey Institute of Technology, from 2014 to 2015. He is currently a Professor of control science and engineering and the Director of the Fujian Engineering Research Center of Motor Control and System Optimal Schedule. His research interests include the supervisory control of discrete event systems, Petri nets, programmable logic controllers, and intelligent manufacturing systems and robots. He was a recipient of the Asian Journal of Control's Outstanding Reviewer in 2011 and the Ho-Pan-Qing-Qi Award in 2012.

DIMITRI LEFEBVRE (Senior Member, IEEE) received the Ph.D. degree in automatic control and computer science from the University of Sciences and Technologies, Lille, France, in 1994, and the HDR degree from the University of Franche-Comté, Belfort, France, in 2000. Since 2001, he has been a Professor with the Institute of Technology and Faculty of Sciences, University of Le Havre Normandie, France. He is with the Research Group on Electrical Engineering and Automatic Control (GREAH), where he was the head of the group from 2007 to 2012. His current research interests include Petri nets, learning processes, adaptive control, fault detection, and diagnosis and their applications to electrical engineering.

ZHIWU LI (Fellow, IEEE) received the B.S. degree in mechanical engineering, the M.S. degree in automatic control, and the Ph.D. degree in manufacturing engineering from Xidian University, Xi'an, China, in 1989, 1992, and 1995, respectively. He joined Xidian University, in 1992. He is currently with the Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau. He has published two monographs with M. Zhou in Springer, in 2009, and Y. Chen in CRC Press, in 2013. His current research interests include Petri net theory and applications, the supervisory control of discrete event systems, workflow modeling and analysis, system reconfiguration, game theory, and data and process mining. He is listed in *Who's Who in the World* (27th Edition, Marquis, 2010). He is the Founding Chair of the Xi'an Chapter of the IEEE Systems, Man, and Cybernetics Society. He serves as a frequent Reviewer for over 80 international journals, including *Automatica*, and a number of IEEE TRANSACTIONS and many international conferences.