The scheduling for batch processes based on clustering approximated timed reachability graphs[☆]

Jiazhong Zhou[a], Dimitri Lefebvre[b,*], Zhiwu Li[c,a]

[a]*School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China*
[b]*GREAH, Normandie University, Le Havre 76600, France*
[c]*Institute of Systems Engineering, Macau University of Science and Technology, Macau SAR, China*

Abstract

To solve some scheduling problems of batch processes based on timed Petri net models, timed extended reachability graphs (TERGs) and approximated TERGs can be used. Such graphs abstract temporal specifications and represent parts of timed languages. By exploring the feasible trajectories in a TERG, optimal schedules can be obtained with respect to the makespans of batch processes that are modeled by timed Petri nets. Nevertheless, the rapid growth of the amount of states in a TERG makes the approach intractable for large systems. In this paper, we propose a clustering approximated TERG that is suitable for large sized batch processes with near-optimal schedules. First, we present a systematic approach to model a batch process with a timed Petri net. Then, information vertices of the clustering approximated TERG for the timed Petri net model are presented to handle time constraints, and a precise metric on the basis of the time constraints of information vertices is designed to avoid excessive approximations. A clustering algorithm is put forward to cluster adjacent information vertices for obtaining a clustering approximated TERG with a reduced size. Finally, an example on scheduling problems for an archetypal

[☆]This work was supported in part by the National Key R&D Program of China under Grant 2018YFB1700104, and the National Natural Science Foundation of China under Grant 61873442.

[*]Corresponding author

Email addresses: `jzzhou@stu.xidian.edu.cn` (Jiazhong Zhou), `dimitri.lefebvre@univ-lehavre.fr` (Dimitri Lefebvre), `zhwli@xidian.edu.cn` (Zhiwu Li)

chemical production plant demonstrates the efficiency of the proposed approach.

Keywords: Batch process, Timed Petri net, Reachability graph, Clustering algorithm, Scheduling.

1. Introduction

A batch chemical system executes various operations by adopting multiple resources, such as storage tanks, piping, valves and so on, according to the recipes of products being ordered. Since several operations may use the same resources, conflicts occur between these operations. As a result, given the processing tasks, different processing sequences lead to different processing times. A typical scheduling problem of batch processes is to allocate resources rationally and design appropriate operation sequences such that the batch plant produces the given products in the shortest time. Batch chemical systems are typical dynamical discrete event systems that can be described accurately by Petri nets [1, 2, 3]. For solving the scheduling problem of batch processes, besides the formal representation at the logical level, temporal constraints and specifications should also be considered. Thanks to their modular design and graphical representation, timed Petri nets [4, 5, 6] are adopted in this article to describe these systems, rather than other discrete event formalisms [7].

For the scheduling problem of batch processes modeled by Petri nets, one promising study direction is to combine graph search algorithms with the execution of Petri nets to compute schedules that lead to the optimal makespan. Ghaeli et al. [8] use the A* algorithm for that purpose. The markings of the reachability graph are heuristically generated and checked to obtain an optimal schedule. Mejia et al. [9] improve heuristic algorithms that prune the search space by controlling the search depth and developing heuristic functions to prevent marking explosion. Li et al. [10] modify a dynamic programming algorithm to select the most promising states evaluated by a heuristic function for further exploration. Mejia et al. [11] present a filtered beam search algorithm, in which a filtering mechanism and an evaluation function are introduced to pre-valuate and limit the expansion of nodes in the reachability graph. Wu et al. [12] model a crude-oil refining process with a hybrid Petri net, and divide the scheduling problem into subproblems with continuous variables solved by linear programming techniques and those with discrete variables solved by heuristic algorithms.

Since the reachability graphs obtained with the above methods do not encode time information, the design of heuristic functions and rules for exploring the nodes of the reachability graph often depends on the specific systems. For practical applications, it becomes difficult to find efficient heuristic functions. An alternative research direction is to combine the state space abstraction of timed Petri nets and a global searching algorithm. These works mainly focus on the improvement of the state space abstraction techniques considering extended time information.

Berthomieu et al. [13] represent finite state spaces with state class graphs for time Petri nets, where the state classes are defined by combining markings with time constraints of enabled transitions at these markings. Inspired by the state class graph, many extended graphs [14, 15, 16, 17] that preserve reachability, branching properties, and linear properties of time Petri nets are presented for model checking or state estimation. In addition, Gardey et al. [18] describe the state space for time Petri nets on the basis of the timed automaton region graph where regions are encoded by bound matrices. Lime et al. [19] construct a time automaton that uses less clock variables to characterize the dynamic evolution of time Petri nets.

However, since the specific firing rule of the transitions is absent in the above methods, these approaches are not appropriate for solving scheduling issues. Moreover, although the timed Petri net system has small scale, these methods may generate graphs with huge state spaces as a consequence of the duplicated time specifications. In [20, 21], a timed extended reachability graph (TERG) is proposed to directly characterize the specific firing times of transitions and the time constraints of enabled transitions under the earliest firing policy. To further reduce the state space, a clustering approximated TERG (ClusTERG) has been designed in [22] for time Petri nets. Then, a global searching algorithm explores the ClusTERG to obtain scheduling solutions. This paper continues this study and focuses on scheduling applications of the ClusTERG.

The main contributions are as follows: (i) we formalize the scheduling problem of batch processes with timed Petri net structures; (ii) we combine the TERG with a hierarchical clustering algorithm to develop a new time graph with a reduced size, namely ClusTERG, for which the sum of the approximation errors is far less than other approximations of TERG proposed in [20]. Compared with [22], we adapt and simplify the construction of the ClusTERG to timed Petri nets. This approach is relevant for a particular case of batch processes, where maximal time constraints are not considered.

This paper is structured as follows. In the following section, batch processes are introduced. Section III reviews the concepts and notations of timed Petri nets and TERG, and designs the timed Petri net model of batch processes. Section IV combines the hierarchical clustering algorithm with the design of the TERG to develop the ClusTERG. Section V presents a case study and discusses the obtained results. Conclusions of this paper are presented in Section VI.

2. Batch processes

Basic notions about batch processes are introduced in this section. In batch processes, each kind of product corresponds to a recipe containing the information required to define the manufacturing requirements [23, 24]. Let us define $\mathbb{N}$ as the set of non negative integer numbers, $\mathbb{R}^+$ as the set of non negative real numbers, and $\mathbb{N}^+$ as the set of strictly positive integers.

Definition 1. *Given a batch process, a recipe $r_i := o_{i,1}\ o_{i,2} \ldots o_{i,J}$ is a sequence of operations that need to be executed according to the order of r_i, where $i, J \in \mathbb{N}^+$.* $\square$

To execute an operation, a batch production system needs several resources, such as tanks and valves, for a certain time. We denote the amount of executions of r_i by $\rho(r_i) \in \mathbb{N}^+$. The set of executed operations of r_i is $O(r_i) =$

$\{o_{i,1}, \ldots, o_{i,J}\}$. The set of resources in a system is $\mathscr{R} = \{v_1, \ldots, v_x, u_1, \ldots, u_y\}$, where $x, y \in \mathbb{N}^+$, $v_1, \ldots, v_x$ denote valves, and $u_1, \ldots, u_y$ denote containers including supply tanks, storage tanks, and reactors. Each resource has two states. We denote the set of all resource states by $\mathscr{R}^s = \{v_1^s, \ldots, v_x^s, \bar{v}_1^s, \ldots, \bar{v}_x^s, u_1^s, \ldots, u_y^s, \bar{u}_1^s, \ldots, \bar{u}_y^s\}$, where $v_1^s, \ldots, v_x^s$ are opened valves, $\bar{v}_1^s, \ldots, \bar{v}_x^s$ are closed valves, $u_1^s, \ldots, u_y^s$ are empty units, and $\bar{u}_1^s, \ldots, \bar{u}_y^s$ are full units. For an operation o, the set of resources required to complete o is denoted by $R(o) \subseteq \mathscr{R}$, the minimal processing time of o is denoted by $d(o) \in \mathbb{R}^+$, and the set of resource states required to complete o is denoted by $R^s(o) \subseteq \mathscr{R}^s$.

For a considered batch system, maximal time constraints are generally not considered [8]. For example, in a closed chemical reactor of a batch system, if a reaction depends on one material, the reaction will stop once the material is totally consumed, and in such a case there is no maximal time constraint [25]. We concentrate on finding sequences of operations that complete the recipes in a minimal time, and the timed Petri net without maximal time constraints is adopted to describe the batch system in this paper. Since the limited resources may be competed by several operations, the design of the processing sequence depends not only on orders formulated by the recipe but also on conflicts as a consequence of the competition for the same resources during executing the operations. Thus, it is complicated to describe recipes and limited resources of a batch process.

Definition 2. *A pair of operations o and o' is said to be conflicting if there exists $v_i \in R(o) \cap R(o')$ with $i \in \mathbb{N}^+$ such that $\bar{v}_i^s \in R^s(o)$ & $v_i^s \in R^s(o')$ holds or there exists $u_j \in R(o) \cap R(o')$ with $j \in \mathbb{N}^+$ such that $\bar{u}_j^s \in R^s(o)$ & $u_j^s \in R^s(o')$ holds, denoted by $o \nparallel o'$; otherwise o and o' are compatible, denoted by $o \parallel o'$. A set of operations is said to be maximally conflicting, denoted by $O_{\max}$, if $o \nparallel o'$ holds for any different operations o and o' in $O_{\max}$, and there exists $o \in O_{\max}$ satisfying $o \parallel o'$ for each $o' \notin O_{\max}$.* □

One can know that any two operations in $O_{\max}$ are conflicting and cannot be performed simultaneously from Definition 2. In addition, if $O_{\max}$ is maximally conflicting, one cannot find another conflicting set $O'_{\max}$ such that $O_{\max} \subsetneqq O'_{\max}$. For a batch process, different processing sequences result in different makespans, i.e., processing times. An archetypal scheduling problem is to find a sequence to execute operations with a minimal makespan [8]. In the following, we adopt a typical chemical plant to demonstrate the presented notations and concepts.

Example 1: Fig. 1 shows a chemical batch production system [26] containing three supply tanks u_1, u_2 and u_3, two reactors u_4 and u_5, two storage tanks u_6 and u_7, and a pipeline system with 13 valves. The production process of this system is as follows. Raw materials are added in tanks u_1, u_2 and u_3. Then, the contents of u_1 are loaded into u_4 or mixed with that of u_2 in u_5 for chemical reaction. The contents of u_3 are unloaded into u_6 to mix with that of u_4 or unloaded into u_7 to mix with that of u_5 for chemical reaction. Finally, chemical products in u_6 and u_7 are transported out. Specifically, recipes of the batch process are shown in Table 1.

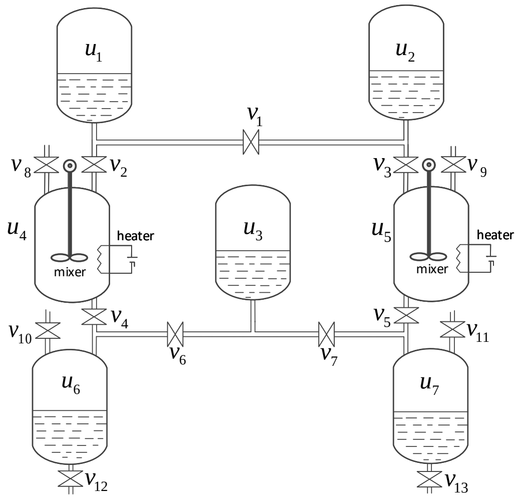

Figure 1: A batch production system.

Assume that the recipes $r_1 = o_{1,1}o_{1,2}o_{1,3}o_{1,4}o_{1,5}$ and $r_2 = o_{2,1}o_{2,2}o_{2,3}o_{2,4}o_{2,5}$ are executed k times in this production system, i.e., $\rho(r_1) = \rho(r_2) = k$. It is obvious that some operations cannot be performed at the same time. For example, $o_{1,1}$ with $R^s(o_{1,1}) = \{v_1^s, v_3^s, \bar{v}_2^s, \bar{v}_5^s, \bar{u}_1^s, \bar{u}_2^s, u_5^s\}$ and $o_{2,1}$ with $R^s(o_{2,1}) = \{v_2^s, \bar{v}_1^s, \bar{v}_4^s, \bar{u}_1^s, u_4^s\}$ are conflicting and cannot be performed simultaneously since $o_{1,1}$ requires v_1 to be opened and $o_{2,1}$ requires v_1 to be closed. In addition, some operations can be performed at the same time. For example, one can perform $o_{1,1}$ and $o_{2,2}$ simultaneously. Different sequences executing the required operations lead to different makespans. For the scheduling problem, the key is to find a suitable sequence executing the required operations to minimize the processing time. $\triangle$

Table 1: Recipes of the batch process shown in Fig. 1.

Recipe	Operation	Operation Description	$R^s(o_{i,j})$	$d(o_{i,j})$ (min.)
r_1	$o_{1,1}$	Transport reactant from u_1 and u_2 to u_5	$\{v_1^s, v_3^s, \bar{v}_2^s, \bar{v}_5^s, \bar{u}_1^s, \bar{u}_2^s, u_5^s\}$	20
	$o_{1,2}$	Chemical reaction in u_5	$\{v_9^s, \bar{v}_3^s, \bar{v}_5^s, \bar{u}_5^s\}$	30
	$o_{1,3}$	Transport material from u_5 and u_3 to u_7	$\{v_5^s, v_7^s, \bar{v}_3^s, \bar{v}_6^s, \bar{v}_{13}^s, \bar{u}_3^s, \bar{u}_5^s, u_7^s\}$	30
	$o_{1,4}$	Chemical reaction in u_7	$\{v_{11}^s, \bar{v}_5^s, \bar{v}_7^s, \bar{v}_{13}^s, \bar{u}_7^s\}$	40
	$o_{1,5}$	The chemical product in u_7 is transported out	$\{v_{13}^s, \bar{v}_5^s, \bar{v}_7^s, \bar{u}_7^s\}$	40
r_2	$o_{2,1}$	Transport material from u_1 to u_4	$\{v_2^s, \bar{v}_1^s, \bar{v}_4^s, \bar{u}_1^s, u_4^s\}$	30
	$o_{2,2}$	Chemical reaction in u_4	$\{v_8^s, \bar{v}_2^s, \bar{v}_4^s, \bar{u}_4^s\}$	40
	$o_{2,3}$	Transport material from u_4 and u_3 to u_6	$\{v_8^s, \bar{v}_2^s, \bar{v}_4^s, \bar{u}_4^s\}$	40
	$o_{2,4}$	Chemical reaction in u_6	$\{v_8^s, \bar{v}_2^s, \bar{v}_4^s, \bar{u}_4^s\}$	50
	$o_{2,5}$	The chemical product in u_6 is transported out	$\{v_8^s, \bar{v}_2^s, \bar{v}_4^s, \bar{u}_4^s\}$	60

3. Timed Petri nets model

A Petri net [7] is enhanced with time specifications and semantics to obtain a timed Petri net. The detailed introduction about timed Petri nets is presented below.

Definition 3. *[7] A timed Petri net structure is a pair $N_{td} = (N, \delta_s)$, in which $N = (\boldsymbol{P}, \boldsymbol{T}, Pre, Post)$ is a Petri net structure with a set of places denoted by $\boldsymbol{P}$, a set of transitions denoted by $\boldsymbol{T}$, and the pre- and post-incidence functions that are defined by Pre: $\boldsymbol{P} \times \boldsymbol{T} \to \mathbb{N}$ and Post: $\boldsymbol{P} \times \boldsymbol{T} \to \mathbb{N}$ specify the flow of tokens from places to transitions and transitions to places, respectively; and $\delta_s : \boldsymbol{T} \to \mathbb{R}^+$ is the static time function that specifies how long the transition $t \in \boldsymbol{T}$ should be enabled before it could fire.* $\square$

We define a marking m by a function m: $\boldsymbol{P} \rightarrow \mathbb{N}$. Given a place $p \in \boldsymbol{P}$, $m(p)$ is the token count in p at m. We enhance an initial marking m_0 with structure N_{td} to get a timed Petri net system (N_{td}, m_0). At marking m, for all $p \in \boldsymbol{P}$ satisfying $Pre(p,t) > 0$, if $m(p) \geq Pre(p,t)$ holds, then t is enabled at m. The enabling degree of t at m is $e_m(t) = \min_{p\in\boldsymbol{P}, Pre(p,t)>0}\{\lfloor m(p)/Pre(p,t)\rfloor\}$, where $\lfloor c \rfloor$ is the round down operation on the real number c. We denote the set of all enabled transitions at m as $T_e(m) = \{t \in \boldsymbol{T} | m[t\rangle\}$. For $t \in T_e(m)$ at m, firing t generates a new marking m', denoted as $m[t\rangle m'$, where $m'(p) = m(p) + Post(p,t) - Pre(p,t)$ for all $p \in \boldsymbol{P}$.

In this article, we focus on bounded timed Petri nets, in which the number of tokens for all places at each reachable marking has an upper bound. In addition, we assume that there exists at most one transition between two reachable markings (i.e., $m[t\rangle m'$ and $m[t'\rangle m'$ imply $t = t'$). For a timed Petri net system, its time semantics [27] are defined under the enabling memory policy and infinite server policy in this paper. The choice policy is a preselection executed by an exterior agent such as a scheduler (i.e., the scheduling solution discussed in Section V): the scheduler decides the selection of transitions to fire when concurrency or conflict occurs. Given a timed Petri net system (N, δ_s, m_0), we write a timed firing sequence as $\sigma = (t_1, \tau_1)(t_2, \tau_2) \cdots (t_h, \tau_h)$, where h is the number of transitions in σ, $t_1, t_2, \ldots, t_h \in \boldsymbol{T}$, and $\tau_1, \tau_2, \ldots, \tau_h$ are the firing time instants of $t_1, t_2, \ldots, t_h$ satisfying $0 \leq \tau_1 \leq \tau_2 \leq \cdots \leq \tau_h$. In addition, $\sigma_{log} = t_1 t_2 \cdots t_h$ is the logical firing sequence associated with σ, neglecting the time instants. We assume that the initial time instant is $\tau_0 = 0$ at marking m_0 and define the timed trajectory corresponding to σ by

$$(\sigma, m_0) = m_0[(t_1, \tau_1)\rangle m_1[(t_2, \tau_2)\rangle \cdots m_{h-1}[(t_h, \tau_h)\rangle m_h, \tag{1}$$

such that $m_1, \ldots, m_{h-1}$ are the markings visited by (σ, m_0), and m_h is the final marking, where $m_0[t_1\rangle m_1$, $m_1[t_2\rangle m_2, \ldots,$ and $m_{h-1}[t_h\rangle m_h$ hold.

Before presenting the definition of the states for a timed Petri net system, the multiset $\mathcal{D}_k$ with $k = 0, \ldots, h$ [28] that consists of dynamic time constraints of enabled transitions is calculated iteratively. Given a trajectory (σ, m_0) taking the form (1) and time τ_k, we define $\boldsymbol{U}_k$ as the set of dynamic time constraints (t, δ) at marking m_k, t being a transition and δ being the residual time before t can fire. We also define $n_k((t, \delta))$ as the number of times that t may fire after δ. Observe that, in general, $n_k((t, \delta))$ is different from $e_{m_k(t)}$.

- When $k = 0$, define $\mathcal{D}_0 = \{(u, n_0(u)) | u \in \boldsymbol{U}_0\}$, where $\boldsymbol{U}_0 = \{(t, \delta) | t \in T_e(m_0), \delta = \delta_s(t)\}$ and $n_0(u) = e_{m_0}(t)$.

- When $k > 0$, define $\mathcal{D}_k = \{(u, n_k(u)) | u \in \boldsymbol{U}_k\}$, where $\boldsymbol{U}_k = \{(t, \delta) \mid t \in T_e(m_k), \delta = \max(0, \delta' - (\tau_k - \tau_{k-1}))\}$ and $n_k(u) = n_{k-1}(u)$ if $(t, \delta') \in \mathcal{D}_{k-1}$ and for all $p \in {}^{\bullet}t, m(p) - Pre(p, t_k) \geq Pre(p, t)$; $\boldsymbol{U}_k = \{(t, \delta) \mid t \in T_e(m_k), \delta = \delta_s(t)\}$ and $n_k(u) = e_{m_k}(t)$ otherwise.

We denote the number of dynamic time constraints in $\mathcal{D}_k$ by $|\mathcal{D}_k|$. Note that $\mathcal{D}_k$ is not empty, i.e., $|\mathcal{D}_k| \neq 0$, if there exist enabled transitions at m_k. When $|\mathcal{D}_k| \neq 0$, we present an order, referred to the $\mathcal{D}$-order, to arrange the dynamic time constraints of $\mathcal{D}_k$. Specifically, given two dynamic time constraints $(t_{i,j}, \delta)$ and $(t_{i',j'}, \delta')$ in $\mathcal{D}_k$, write $(t_{i,j}, \delta) \prec (t_{i',j'}, \delta')$ if one of the next statements is true: (i) $i < i'$, (ii) $i = i'$ and $j < j'$, and (iii) $i = i'$, $j = j'$, and $\delta < \delta'$.

Definition 4. *A state of a timed Petri net system is a pair $S = (m, \mathcal{D})$, where m is a marking and $\mathcal{D}$ is the multiset of dynamic time constraints of the transitions enabled at m.* □

The detailed algorithms to model the batch production system with a timed Petri net are shown in [26]. In particular, monitors are introduced to solve conflicting situations in maximal conflicting sets.

Example 2: Reconsider the batch process described in Example 1, and its timed Petri net model (N, δ_s, m_0) is shown in Fig. 2, with

1) two starting places $p_{1,0}$ and $p_{2,0}$ with $m_0(p_{1,0}) = \rho(r_1) = k$ and $m_0(p_{2,0}) = \rho(r_2) = k$;

2) the places $p_{i,j}$ corresponding to the operations $o_{i,j}$, $i = 1, 2$, $j = 1, \ldots, 5$;

3) the transitions $t_{i,j}$ representing the execution of $o_{i,j}$, $i = 1, 2$, $j = 1, \ldots, 5$.

In addition, all maximally conflicting operation sets are computed, which are $\{o_{1,1}, o_{2,1}\}$, $\{o_{1,1}, o_{1,2}, o_{1,3}\}$, $\{o_{2,1}, o_{2,2}, o_{2,3}\}$, $\{o_{1,3}, o_{1,4}, o_{1,5}\}$, $\{o_{2,3}, o_{2,4}, o_{2,5}\}$, and $\{o_{1,3}, o_{2,3}\}$. The monitor places $p_{c1}, p_{c2}, \ldots, p_{c6}$ with $m_0(p_{c1}) = m_0(p_{c2}) = \cdots = m_0(p_{c6}) = 1$ are designed to ensure $\sum_{j=1}^{3} m(p_{i,j}) \leq 1$ and $\sum_{j=3}^{5} m(p_{i,j}) \leq 1$, $m(p_{1,1}) + m(p_{2,1}) \leq 1$, and $m(p_{1,3}) + m(p_{2,3}) \leq 1$ for $i = 1, 2$, where m is a reachable marking at m_0. That is, transitions representing the execution of operations in $O_{\max}$ cannot fire synchronously.

The initial state is $S_0 = (m_0, \mathcal{D}_0)$ at $\tau_0 = 0$. In detail, at m_0, we have $T_e(m_0) = \{t_{1.1}, t_{2.1}\}$ and $e_{m_0}(t_{1.1}) = e_{m_0}(t_{2.1}) = k$. Time constraints $(t_{1.1}, 20)$ and $(t_{2.1}, 30)$ belong to $\mathcal{D}_0$. According to the $\mathcal{D}$-order, since $i < i'$, $(t_{i,j}, \delta) = (t_{1,1}, 20) \prec (t_{i',j'}, \delta') = (t_{2,1}, 30)$ holds. Thus, we have $\mathcal{D}_0 = \{((t_{1.1}, 20), k), ((t_{2.1}, 30), k)\}$. △

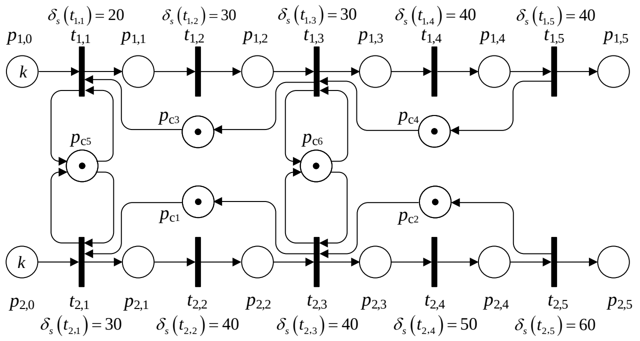

Figure 2: The timed Petri net system to describe the batch process in Fig. 1.

4. Clustering approximated time reachability graph

The state space of the resulting Petri net model can be explored to calculate a sequence of operations that complete the recipes in a minimal time. We can analyze the timed Petri net system through constructing the TERG, or other time graphs preserving the temporal properties for the net. The scheduling problem is reformulated as a searching problem to discover the optimal path from a weighted graph with the Dijkstra algorithm [29].

In this article, we focus on timed Petri nets under the Earliest Firing Policy (EFP). In detail, if a transition with multiple enabling degree is preselected for the following firing, it will fire when its residual time with minimal value elapses [20, 21]. Given a timed Petri net system (N, δ_s, m_0) under the EFP, a timed trajectory (σ, m_0) taking the form (1) is feasible if the following conditions are satisfied [22]: for all $k = 1, \ldots, h$ (i) t_k is enabled at m_{k-1}; (ii) $\delta \geq 0$ holds for all time transitions (t, δ) of $\mathcal{D}_k$; (iii) there exists $((t_k, \tau_k - \tau_{k-1}), n_k) \in \mathcal{D}_{k-1}$; (iv) $\delta' \geq \tau_k - \tau_{k-1}$ holds for all time constraints (t_k, δ') of $\mathcal{D}_{k-1}$. Based on the notion of the feasible trajectory, we define the reachable states of (N, δ_s, m_0).

Definition 5. *Given (N, δ_s, m_0) under the EFP, states $S_0 = (m_0, \mathcal{D}_0)$ and $S = (m, \mathcal{D})$, S is reachable from S_0 if there is a feasible trajectory (σ, m_0) taking the form (1) satisfying $\mathcal{D}_h = \mathcal{D}$ and $m_h = m$, denoted as $S_0 \xrightarrow{(\sigma, m_0)} S$.* □

Definition 6. *[20] Let (N, δ_s, m_0) be a timed Petri net system. We define its TERG as $G_E = (\boldsymbol{S_E}, \Omega_E, B_E, S_0)$, where*

- *$S_0 = (m_0, \mathcal{D}_0)$ is the initial state;*
- *$\boldsymbol{S_E}$ is a set of states, defined by $\boldsymbol{S_E} = \{S | \exists (\sigma, m_0) : S_0 \xrightarrow{(\sigma, m_0)} S\}$;*
- *Ω_E is the transition function: $\boldsymbol{S_E} \times \boldsymbol{S_E} \to \boldsymbol{T} \cup \{\varepsilon\}$. In detail, for states $S, S' \in \boldsymbol{S_E}$ and $S = (m, \mathcal{D})$, we have $\Omega_E(S, S') = t$ if $S \xrightarrow{(\sigma, m)} S'$ with $\sigma_{log} = t$; otherwise $\Omega_E(S, S') = \varepsilon$ holds with ε standing for the empty sequence;*
- *B_E is the firing time function: $\boldsymbol{S_E} \times \boldsymbol{S_E} \to \mathbb{R}^+ \cup \{+\infty\}$. In detail, for state $S, S' \in \boldsymbol{S_E}$ and $S = (m, \mathcal{D})$, $B_E(S, S') = \delta$ holds if (1) $S \xrightarrow{(\sigma, m)} S'$ and $\sigma_{log} = t$, and (2) for all $(u', n(u')) \in \mathcal{D}$ satisfying $u' = (t, \delta')$, $\delta \leq \delta'$ holds; otherwise $B_E(S, S') = +\infty$ holds.* □

Example 3: A timed Petri net system that has one place p_1, two transitions $t_{1,1}$ and $t_{1,2}$, $\delta_s(t_{1,1}) = 3$, $\delta_s(t_{1,2}) = 5$, and initial marking m_0 with $m_0(p_1) = 2$, is depicted in Fig. 3. Its TERG is given in Fig. 4. △

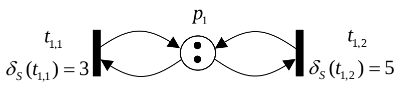

Figure 3: A timed Petri net system having a single marking.

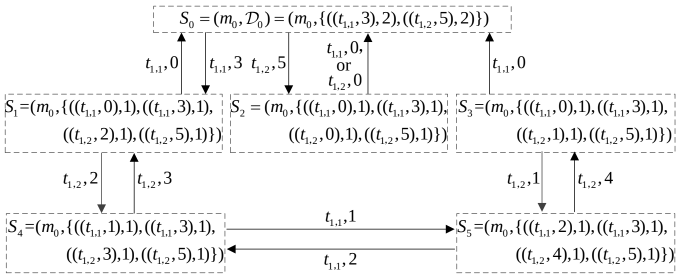
$S_0 = (m_0, \mathcal{D}_0) = (m_0, \{((t_{1,1}, 3), 2), ((t_{1,2}, 5), 2)\})$
$t_{1,1}, 0$
$t_{1,1}, 3$
$t_{1,2}, 5$
$t_{1,1}, 0,$ or $t_{1,2}, 0$
$t_{1,1}, 0$
$S_1 = (m_0, \{((t_{1,1}, 0), 1), ((t_{1,1}, 3), 1), ((t_{1,2}, 2), 1), ((t_{1,2}, 5), 1)\})$
$S_2 = (m_0, \{((t_{1,1}, 0), 1), ((t_{1,1}, 3), 1), ((t_{1,2}, 0), 1), ((t_{1,2}, 5), 1)\})$
$S_3 = (m_0, \{((t_{1,1}, 0), 1), ((t_{1,1}, 3), 1), ((t_{1,2}, 1), 1), ((t_{1,2}, 5), 1)\})$
$t_{1,2}, 2$
$t_{1,2}, 3$
$t_{1,2}, 1$
$t_{1,2}, 4$
$S_4 = (m_0, \{((t_{1,1}, 1), 1), ((t_{1,1}, 3), 1), ((t_{1,2}, 3), 1), ((t_{1,2}, 5), 1)\})$
$t_{1,1}, 1$
$t_{1,1}, 2$
$S_5 = (m_0, \{((t_{1,1}, 2), 1), ((t_{1,1}, 3), 1), ((t_{1,2}, 4), 1), ((t_{1,2}, 5), 1)\})$

Figure 4: The TERG of the time Petri net system presented in Fig. 3.

Definition 6 is a simplification of the TERG defined in [22] in the case that the maximal residual time is not considered. However, the number of states in the TERG may increase unmanageably. To mitigate the state explosion problem in the TERG, some different states with the same marking can be merged. Let $\boldsymbol{S} = \{S_i = (m, \mathcal{D}_i) | i = 1, 2, \ldots, q\}$ be a set of q different states owning the identical marking m. If $|\mathcal{D}_i| = 0$ holds for a given state $S_i \in \boldsymbol{S}$, there is no enabled transition at m; thus $|\mathcal{D}_{i'}| = 0$ holds also for $i' = 1, 2, \ldots, q$. In this case, $\boldsymbol{S}$ has only a state $(m, \emptyset)$, and we do not consider any merging. Otherwise, if $|\mathcal{D}_i| \neq 0$, $i = 1, 2, \ldots, q$, the time constraints $(t_j, \delta_{i,j})$ with $j = 1, \ldots, k$ and $k = \sum\limits_{t \in T_e(m)} e(t)$ in $\mathcal{D}_i$ are enumerated and sorted according to the $\mathcal{D}$-order. The merging of the states with the same marking leads to a five-tuple named as an information vertex. The detailed definition is shown below.

Definition 7. *Let $\boldsymbol{S} = \{S_i = (m, \mathcal{D}_i) | i = 1, 2, \ldots, q\}$ be a set of states with the same marking. The merging of these states results in a five-tuple $V = (m, q, \mathcal{D}^{\min}, \mathcal{D}^{\max}, \mathcal{D}^{\text{ave}})$, named as an information vertex, where*

- *m is the marking of all merged states;*
- *q is a positive integer recording the number of merged states;*

- $\mathcal{D}^{\min} = \{(t_j, \delta_j^{\min})|j = 1, \ldots, k, t_j \in T_e(m)\}$ *with* $\delta_j^{\min} = \min_{i=1,\ldots,q}\{\delta_{i,j}\}$;
- $\mathcal{D}^{\max} = \{(t_j, \delta_j^{\max})|j = 1, \ldots, k, t_j \in T_e(m)\}$ *with* $\delta_j^{\max} = \max_{i=1,\ldots,q}\{\delta_{i,j}\}$;
- $\mathcal{D}^{\text{ave}} = \{(t_j, \delta_j^{\text{ave}})|j = 1, \ldots, k, t_j \in T_e(m)\}$ *with* $\delta_j^{\text{ave}} = \sum\limits_{i=1}^{q}(\delta_{i,j})/q$. $\square$

When a certain number of states are generated, we merge them as aforementioned. Some resulting information vertices may be further merged. More generally, let $\boldsymbol{X}(m) = \bigcup\limits_{i=1}^{l}\{V_i = (m, q_i, \mathcal{D}_i^{\min}, \mathcal{D}_i^{\max}, \mathcal{D}_i^{\text{ave}})\}$ be a set of l different information vertices with the same marking m. Given $\mathcal{D}_i^{\min}$, $\mathcal{D}_i^{\max}$, and $\mathcal{D}_i^{\text{ave}} \neq \emptyset$, $i = 1, 2, \ldots, l$, let $\mathcal{D}_i^{\min} = \{(t_j, \delta_{i,j}^{\min})|j = 1, \ldots, k, t_j \in T_e(m)\}$, $\mathcal{D}_i^{\max} = \{(t_j, \delta_{i,j}^{\max})|j = 1, \ldots, k, t_j \in T_e(m)\}$, and $\mathcal{D}_i^{\text{ave}} = \{(t_j, \delta_{i,j}^{\text{ave}})|j = 1, \ldots, k, t_j \in T_e(m)\}$, with $k = \sum\limits_{t \in T_e(m)} e(t)$. The merging of these already existing information vertices with the same marking also leads to an information vertex $V = (m, q, \mathcal{D}^{\min}, \mathcal{D}^{\max}, \mathcal{D}^{\text{ave}})$, where $q = \sum\limits_{i=1}^{l} q_i$, $\mathcal{D}^{\min} = \{(t_j, \delta_j^{\min})|j = 1, \ldots, k, t_j \in T_e(m)\}$ with $\delta_j^{\min} = \min_{i=1,\ldots,l}\{\delta_{i,j}^{\min}\}$, $\mathcal{D}^{\max} = \{(t_j, \delta_j^{\max})|j = 1, \ldots, k, t_j \in T_e(m)\}$ with $\delta_j^{\max} = \max_{i=1,\ldots,l}\{\delta_{i,j}^{\max}\}$, and $\mathcal{D}^{\text{ave}} = \{(t_j, \delta_j^{\text{ave}})|j = 1, \ldots, k, t_j \in T_e(m)\}$ with $\delta_j^{\text{ave}} = (\sum\limits_{i=1}^{l}(q_i \times \delta_{i,j}^{\text{ave}}))/\sum\limits_{i=1}^{l} q_i$.

Moreover, considering a set of information vertices $\boldsymbol{X}(m)$ with the same marking m, we introduce a metric, named $\mathcal{D}$-distance, in order to measure the maximal difference among these vertices. Specifically, we calculate the $\mathcal{D}$-distance within $\boldsymbol{X}(m)$ as follows:

$$D_{\mathcal{D}}(\boldsymbol{X}(m)) = \max_{V_i, V_{i'} \in \boldsymbol{X}(m)}\{\max_{j=1,\ldots,k}|\delta_{i,j}^{\min} - \delta_{i',j}^{\max}|\} \tag{2}$$

Example 4: Consider the timed Petri net system and its TERG shown in Figs. 3 and 4. Each state in Fig. 4 corresponds to an information vertex. For example, $S_0 = (m, \mathcal{D}_0)$ corresponds to $V_0 = (m$, 1, $\mathcal{D}_0^{\min}$, $\mathcal{D}_0^{\max}$, $\mathcal{D}_0^{\text{ave}})$ with $\mathcal{D}_0^{\min}$ $=\mathcal{D}_0^{\max}=\mathcal{D}_0^{\text{ave}}= \mathcal{D}_0 =\{(t_{1,1}, 3)$, $(t_{1,1}, 3)$, $(t_{1,2}, 5)$, $(t_{1,2}, 5)\}$. Then, similar to the computation of S_1 and S_2 from S_0, one can obtain V_1 and V_2 from V_0. Specifically, V_1=$(m$, 1, $\mathcal{D}_1^{\min}$, $\mathcal{D}_1^{\max}$, $\mathcal{D}_1^{\text{ave}})$ with $\mathcal{D}_1^{\min}$ $=\mathcal{D}_1^{\max}$=$\mathcal{D}_1^{\text{ave}}$=$\{(t_{1,1}, 0)$, $(t_{1,1}, 3)$, $(t_{1,2}, 2)$, $(t_{1,2}, 5)\}$, and V_2=$(m$, 1, $\mathcal{D}_2^{\min}$, $\mathcal{D}_2^{\max}$, $\mathcal{D}_2^{\text{ave}})$ with $\mathcal{D}_2^{\min}$ $=\mathcal{D}_2^{\max}$= $\mathcal{D}_2^{\text{ave}}$=

$\{(t_{1,1}, 0)$, $(t_{1,1}, 3)$, $(t_{1,2}, 0)$, $(t_{1,2}, 5)\}$. If the above vertices are merged, one can obtain a new vertex $V = (m, q, \mathcal{D}^{\min}, \mathcal{D}^{\max}, \mathcal{D}^{\text{ave}})$, where $q = 3$, $\mathcal{D}^{\min} = \{(t_{1,1}, 0)$, $(t_{1,1}, 3)$, $(t_{1,2}, 0)$, $(t_{1,2}, 5)\}$, $\mathcal{D}^{\max} = \{(t_{1,1}, 3)$, $(t_{1,1}, 3)$, $(t_{1,2}, 5)$, $(t_{1,2}, 5)\}$, and $\mathcal{D}^{\text{ave}} = \{(t_{1,1}, 1)$, $(t_{1,1}, 3)$, $(t_{1,2}, 7/3)$, $(t_{1,2}, 5)\}$.

In order to compute the $\mathcal{D}$-distance for $\boldsymbol{X}(m) = \{V_0, V_1, V_2\}$, we first calculate the $\mathcal{D}$-distance within any two information vertices in $\boldsymbol{X}(m)$. Specifically, the $\mathcal{D}$-distance within $\{V_0, V_1\}$ is $D_{\mathcal{D}}(\{V_0, V_1\}) = \max\{|3-0|, |3-3|, |5-2|, |5-5|\} = 3$. Moreover, $D_{\mathcal{D}}(\{V_0, V_2\}) = 5$ and $D_{\mathcal{D}}(\{V_1, V_2\}) = 2$. The $\mathcal{D}$-distance within $\boldsymbol{X}(m)$ is $D_{\mathcal{D}}(\boldsymbol{X}(m)) = \max\{3, 5, 2\} = 5$. $\triangle$

Furthermore, given a set of information vertices $\boldsymbol{X} = \{V_1, \ldots, V_l\}$ and parameter λ, the logical condition $C_m(\boldsymbol{X}, \lambda)$ to merge these information vertices is as follows:

$$C_m(\boldsymbol{X}, \lambda) = (m_1 = \cdots = m_l) \wedge (D_{\mathcal{D}}(\boldsymbol{X}) \leq \lambda) \tag{3}$$

where $m_1, \ldots, m_l$ are the markings of $V_1, \ldots, V_l$, respectively.

Since the states of a TERG can be represented by numerical data points, clustering algorithms [30] such as the hierarchical clustering can be adopted to cluster and merge these states to obtain a new time graph. We name the new graph as a clustering approximated TERG (ClusTERG). In data mining and statistics, the hierarchical clustering [31, 32] is a common algorithm to cluster data points. This algorithm firstly regards each data points as one cluster. Then, it groups two clusters having the minimal difference as a new cluster and repeats the process iteratively.

For the hierarchical clustering algorithm, the difference of clusters is quantified by the linkage criterion such as the maximum linkage clustering (MLC), the minimum linkage clustering, the centroid linkage clustering, and so on [32]. In this article, we adopt the MLC since it calculates the distance of the two farthest data points as the distance of clusters.

One can notice that information vertices are data points, and sets $\boldsymbol{X}(m)$ and $\boldsymbol{X}'(m)$ of information vertices are regarded as two clusters. The MLC can be computed directly with the $\mathcal{D}$-distance in the hierarchical clustering algorithm for two sets $\boldsymbol{X}(m)$ and $\boldsymbol{X'}(m)$ of information vertices. We calculate the MLC by $\mathrm{MLC}(\boldsymbol{X}(m), \boldsymbol{X'}(m)) = D_{\mathcal{D}}(\boldsymbol{X}(m) \cup \boldsymbol{X'}(m))$. Considering a set of information vertices $\boldsymbol{X}(m) = \{V_1, \dots, V_l\}$ where $V_1, \dots, V_l$ have the same marking m, the detailed processes of the hierarchical clustering for $\boldsymbol{X}(m)$ are:

1. Initialize information vertices of the data set $\boldsymbol{X}(m)$ to l clusters $\boldsymbol{X}_j$ with $j = 1, \dots, l$, i.e., $\boldsymbol{X}_j = \{V_j\}$. At initialization, the amount n of clusters is identical with the amount l of information vertices in $\boldsymbol{X}(m)$.
2. For all $i, j = 1, \dots, n$, $i \neq j$, compute $\mathrm{MLC}(\boldsymbol{X}_i, \boldsymbol{X}_j)$. If there exist two clusters such that $\mathrm{MLC}(\boldsymbol{X}_i, \boldsymbol{X}_j) \leq \lambda$, then perform Step 3; otherwise, perform Step 4.
3. Search for the two clusters $\boldsymbol{X}$ and $\boldsymbol{X}'$ with the minimal $\mathrm{MLC}(\boldsymbol{X}, \boldsymbol{X}')$, and group them into one cluster $\boldsymbol{X}'' = \{V\}$, where V is obtained by the merging of information vertices in $\boldsymbol{X}$ and $\boldsymbol{X}'$.
4. Finally, a data set $\boldsymbol{X}'(m)$ with n clusters is obtained.

The time complexity of the hierarchical clustering for a set of information vertices $\boldsymbol{X}(m)$ is quadratic. Next, a new graph, referred to the ClusTERG, is obtained by combining the construction of the TERG with the clustering algorithm. By calculating the MLC, each time γ new information vertices are generated, the hierarchical clustering is adopted to merge information vertices with MLC $< \lambda$, where λ and γ are input parameters to be selected depending on the size of the system.

Definition 8. *Given a timed Petri net system (N, δ_s, m_0) and the parameters λ and γ, we define the ClusTERG as a five-tuple $G_C = (\boldsymbol{\mathcal{V}}, \Omega, B, V_0)$, where*

- *$\boldsymbol{\mathcal{V}}$ is a finite set of information vertices, and for any two different vertices $V, V' \in \boldsymbol{\mathcal{V}}$ with the same marking, $D_{\mathcal{D}}(\{V, V'\}) \geq \lambda$ holds;*
- *Ω is the transition function: $\boldsymbol{\mathcal{V}} \times \boldsymbol{\mathcal{V}} \to \boldsymbol{T} \cup \{\varepsilon\}$. In detail, for vertices $V, V' \in \boldsymbol{\mathcal{V}}$ with $V = (m, q, \mathcal{D}^{\min}, \mathcal{D}^{\max}, \mathcal{D}^{\mathrm{ave}})$ and $V' = (m', q', \mathcal{D}'^{\min}, \mathcal{D}'^{\max},$*

$\mathcal{D}'^{\text{ave}}$), $\Omega(V,V')= t$ if (1) m' can be obtained at m by firing t and (2) there is $(u,n(u)) \in \mathcal{D}^{\text{ave}}$ satisfying $u = (t,\delta)$; otherwise $\Omega(V,V') = \varepsilon$;

- *B is the firing time function: $\boldsymbol{\mathcal{V}} \times \boldsymbol{\mathcal{V}} \to \mathbb{R}^+ \cup \{+\infty\}$. For vertices $V,V' \in \boldsymbol{\mathcal{V}}$ with $V = (m,q,\mathcal{D}^{\min},\mathcal{D}^{\max},\mathcal{D}^{\text{ave}})$ and $V' = (m',q',\mathcal{D}'^{\min},\mathcal{D}'^{\max},\mathcal{D}'^{\text{ave}})$, $B(V,V') = \delta$ if (1) $\Omega(V,V') = t$, and (2) for all $(u',n(u')) \in \mathcal{D}^{\text{ave}}$ with $u' = (t,\delta')$, $\delta \le \delta'$ holds; otherwise $B(V,V') = +\infty$;*

- *$V_0 = (m_0, 1, \mathcal{D}_0, \mathcal{D}_0, \mathcal{D}_0)$ is the initial information vertex.* □

Algorithm 1 shows the detailed construction of the ClusTERG. This algorithm uses a temporary list $\mathcal{L}$ of unexplored information vertices sorted according to the generation order of them and $\mathcal{L}[i]$ refers to the ith element in list $\mathcal{L}$.

In Algorithm 1, we first initialize V_0 and the temporary list $\mathcal{L}$ according to Steps 1–6. Then, $\mathcal{L}[1] = V_0$ is selected. The possible successors of V_0 are explored and evaluated whether they are newly generated according to Steps 9–24. Finally, each time γ information vertices are newly generated, the hierarchical clustering algorithm is called to merge vertices with same marking and MLC$< \lambda$ according to Steps 25–31. Obviously, for information vertices merged in one cluster, the maximal difference among their time constraints is not more than λ [22].

5. Experimental results

This section evaluates the performance of the TERG, the approximated TERG proposed in [20] and the ClusTERG from agglomeration errors and computation time aspects. The scheduling results based on the different graphs are also compared.

We compare the agglomeration errors generated by using the algorithm in [20] with that of Algorithm 1. In detail, the whole agglomeration error for a set of clusters $\boldsymbol{Z} = \{\boldsymbol{X_1}, \ldots, \boldsymbol{X_n}\}$ is defined by

$$E(\boldsymbol{Z}) = \sum_{\boldsymbol{X}_i \in \boldsymbol{Z}} D_{\mathcal{D}}(\boldsymbol{X}_i) \tag{4}$$

Algorithm 1: Construction of the ClusTERG

```
Input: Parameters $\lambda$ and $\gamma$, and $(N,\ m_0,\ \delta_s)$
Output: The ClusTERG $G = (\boldsymbol{\mathcal{V}},\ \Omega,\ B,\ V_0)$
1  Initialize the number of information vertices: $N_C \leftarrow 0$
2  for each transition $t \in T_e(m_0)$ do
3      compute the enabling degree $e_{m_0}(t)$ of $t$
4      $((t, \delta_s(t), \Delta_s(t)), e_{m_0}(t))$ is added in $\mathcal{D}_0$
5  sort time constraints of $\mathcal{D}_0$ based on the $\mathcal{D}$-order
6  $V_0 \leftarrow (m_0, 1, \mathcal{D}_0, \mathcal{D}_0, \mathcal{D}_0)$, $\boldsymbol{\mathcal{V}} \leftarrow \{V_0\}$, add $V_0$ into $\mathcal{L}$
7  while $\mathcal{L}$ is not empty do
8      $V \leftarrow \mathcal{L}[1]$, remove $\mathcal{L}[1]$ from $\mathcal{L}$
9      if $V = (m, q, \mathcal{D}^{\min}, \mathcal{D}^{\max}, \mathcal{D}^{\text{ave}})$ and $\mathcal{D}^{\text{ave}} \neq \emptyset$ then
10         for each time constraint of $\mathcal{D}^{\text{ave}}$ do
11             extract $((t_h, \delta_h, \Delta_h), n_h)$ from $\mathcal{D}^{\text{ave}}$
12             if there is no $((t_{h'}, \delta_{h'}), n_{h'})$ in $\mathcal{D}^{\text{ave}}$ such that $n_h = n_{h'}$,
               $t_{h'} = t_h$, and $\delta_{h'} < \delta_h$ then
13                 calculate $m'$ at $m$ by firing $t_h$
14                 calculate $\mathcal{D}'$ at $\mathcal{D}^{\text{ave}}$
15                 sort $\mathcal{D}'$ on the basis of the $\mathcal{D}$-order
16                 ${\mathcal{D}'}^{\min} \leftarrow \mathcal{D}'$, ${\mathcal{D}'}^{\max} \leftarrow \mathcal{D}'$, ${\mathcal{D}'}^{\text{ave}} \leftarrow \mathcal{D}'$
17                 $V' \leftarrow (m', 1, {\mathcal{D}'}^{\min}, {\mathcal{D}'}^{\max}, {\mathcal{D}'}^{\text{ave}})$
18                 if there is an information vertex $V^* \in \boldsymbol{\mathcal{V}}$ with $V^* = V'$
                   then
19                     $\Omega(V, V^*) \leftarrow t_h$, $B(V, V^*) \leftarrow \delta_h$
20                 else
21                     $V_{|\boldsymbol{\mathcal{V}}|} \leftarrow V'$, $\boldsymbol{\mathcal{V}} \leftarrow \boldsymbol{\mathcal{V}} \cup \{V_{|\boldsymbol{\mathcal{V}}|}\}$
22                     $\mathcal{L} \leftarrow \mathcal{L} \cup \{V_{|\boldsymbol{\mathcal{V}}|}\}$
23                     $\Omega(V, V_{|\boldsymbol{\mathcal{V}}|}) \leftarrow t_h$
24                     $B(V, V_{|\boldsymbol{\mathcal{V}}|}) \leftarrow \delta_h$
25             if $(\mathcal{L} = \emptyset)$ or $(|\boldsymbol{\mathcal{V}}| - N_C > \gamma)$ then
26                 for $V \in \boldsymbol{\mathcal{V}}$ do
27                     put information vertices with the same marking into
                       $\boldsymbol{X}(m)$
28                 for each $\boldsymbol{X}(m) \subseteq \boldsymbol{\mathcal{V}}$ do
29                     get $\boldsymbol{X'}(m)$ by using the hierarchical clustering
                       algorithm to cluster $\boldsymbol{X}(m)$
30                     $\boldsymbol{\mathcal{V}} \leftarrow (\boldsymbol{\mathcal{V}}/\boldsymbol{X}(m)) \cup \boldsymbol{X'}(m)$
31                 $N_C \leftarrow |\boldsymbol{\mathcal{V}}|$
```

The maximal agglomeration error $\mathcal{E}(\boldsymbol{Z})$ is also considered

$$\mathcal{E}(\boldsymbol{Z}) = \max_{\boldsymbol{X}_i \in \boldsymbol{Z}} D_{\mathcal{D}}(\boldsymbol{X}_i) \tag{5}$$

We focus on the timed Petri net model in Fig. 2 that behaves under the EFP for $k = 2$ (k being the number of repetitions of each recipe). Approximations of the timed language of this system are obtained by the algorithm in [20] and Algorithm 1. The agglomeration errors, the numbers N_A of states in approximated TERG computed in [20], the number N_C of information vertices in ClusTERG, and the time to get the result are reported in Table 2.

Table 2: Outcomes of diverse algorithms for the timed Petri model in Fig. 2

The algorithm in [20]						
λ	0	10	20	30	50	$+\infty$
N_A	1551	906	603	442	259	225
Time to get the result	5.41s	1.63s	0.93s	0.52s	0.17s	0.14s
$E(\boldsymbol{Z})$	0	3290	5530	6910	7330	7480
$\mathcal{E}(\boldsymbol{Z})$	0	30	40	60	60	60
Algorithm 1 (with $\gamma = 5$)						
λ	0	10	20	30	50	$+\infty$
N_C	1551	995	670	445	229	225
Time to get the result	90.39s	56.74s	25.13s	15.48s	2.06s	1.96s
$E(\boldsymbol{Z})$	0	2225.9	3645.4	5850.6	6808.9	6894.9
$\mathcal{E}(\boldsymbol{Z})$	0	10	20	30	49.7	60
Algorithm 1 (with $\gamma = 50$)						
λ	0	10	20	30	50	$+\infty$
N_C	1551	992	663	464	233	225
Time to get the result	9.54s	7.64s	4.17s	2.48s	0.83s	0.71s
$E(\boldsymbol{Z})$	0	2241.9	3488.8	5417.7	6911.3	6982.6
$\mathcal{E}(\boldsymbol{Z})$	0	10	20	30	50	60

The amount of states in the approximated TERG gained by the algorithm in [20] is close to that gained by Algorithm 1. The sum of agglomeration errors $E(\boldsymbol{Z})$ obtained by Algorithm 1 is smaller than that obtained by the algorithm in [20]. The clustering method detailed in Algorithm 1 is used with $\gamma = 5$ and $\gamma = 50$. Observe that a large value of γ reduces the number of iterations as well as the complexity in time. However, Algorithm 1 with a large value of γ increases the complexity in space since larger subsets of information vertices must be temporarily stored.

Moreover, more tests are conducted by selecting different parameter values of γ for a given λ ($\lambda = 10$, $\lambda = 20$, $\lambda = 30$, or $\lambda = 50$). The dependence of the global agglomeration error with respect to parameter γ is shown in Fig. 5. The global agglomeration error has low values when γ is set from 2 to 200. In the following tests, we set $\gamma = 100$. In addition, considering the impact of λ on the complexity in space and time, we set $\lambda = 30$.

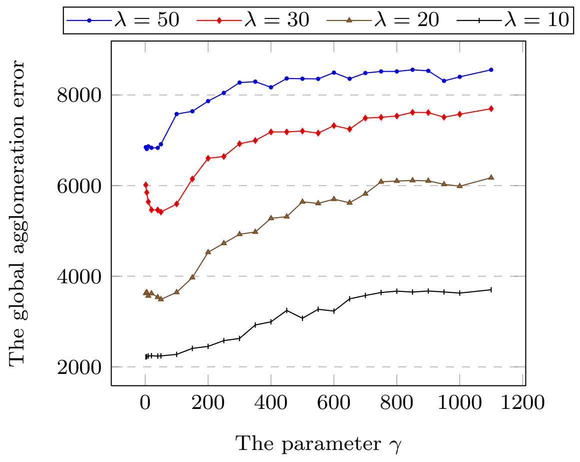

Figure 5: Global agglomeration error with respect to parameter γ.

To illustrate the application of scheduling issues, Table 3 summarizes the outcomes got by Dijkstra algorithm searching for the TERG, approximated TERG computed as in [20] and ClusTERG in this paper. In Table 3, we report in function of k the amount N_E of states in TERG, the amount N_A of states in the approximated TERG, the amount N_C of information vertices in ClusTERG, the makespan $C_{\max}$ based on Dijkstra algorithm, and the time to get these results. For these graphs, the amount of states or information vertices searched by Dijkstra algorithm is restricted to 8,000. Note that $C_{\max}$ based on Dijkstra algorithm and the whole TERG (without merging states) brings about the shortest processing time for $k = 1, 2, 3$. However, from $k \geq 4$, the size of TERG exceeds 8000 states and the method based on TERG is no longer tractable. Increasing k induces a combinatory explosion of the amount of information vertices or states.

In contrast, Dijkstra algorithm based on approximated graphs can be used for larger values of k, since the approximations of graphs induce a decrease in the amount of information vertices or states.

Table 3: The performance of Dijkstra algorithm based on different graphs

TERG						
k	1	2	3	4	5	6
Time (s)	0.24	34.07	889.10	Not found	Not found	Not found
N_E	86	1551	5007	> 8000	> 8000	> 8000
$C_{\max}$ (min.)	220	370	520	Not found	Not found	Not found
The approximated TERG obtained by the algorithm in [20] with $\lambda = 30$						
k	1	2	3	4	5	6
Time (s)	0.24	3.25	18.02	73.32	224.41	2262.67
N_A	55	437	1199	2375	3915	5871
$C_{\max}$ (min.)	240	430	620	730	1010	1180
ClusTERG obtained by Algorithm 1 (with hierarchical clustering, $\lambda = 30$, $\gamma = 100$)						
k	1	2	3	4	5	6
Time (s)	0.25	7.17	54.27	673.79	1036.67	2977.21
N_C	55	472	1336	2604	4279	6398
$C_{\max}$ (min.)	240	410	600	710	860	1010

Most solutions based on the ClusTERG have a better makespan than solutions based on the approximated graph in [20]. Observe that the solutions obtained by the application of the Dijkstra algorithms in the ClusTERG are in general not optimal since approximation errors induce a bias in the computation of the optimal paths (see for example the solution obtained with the ClusTERG for $k = 1$, 2, 3 in Table 3).

In Fig. 6, the scheduling solutions obtained by searching TERG, approximated TERG proposed in [20] and ClusTERG are σ_1, σ_2 and σ_3, respectively. For $k = 6$, we cannot compute the scheduling solution σ_1 because the TERG exceeds the maximal allowed space, but we can execute σ_0 twice, where σ_0 is the scheduling solution obtained with the TERG for $k = 3$. Fig. 6 details the above solutions. Among these solutions, we find the minimal makespan obtained with σ_3 that results from the proposed ClusTERG. Compared to σ_1, we save 30 minutes (1010 minutes instead of 1040 minutes), and compared to σ_2, we save 170 minutes (1010 minutes instead of 1180 minutes). It worths noting that there is

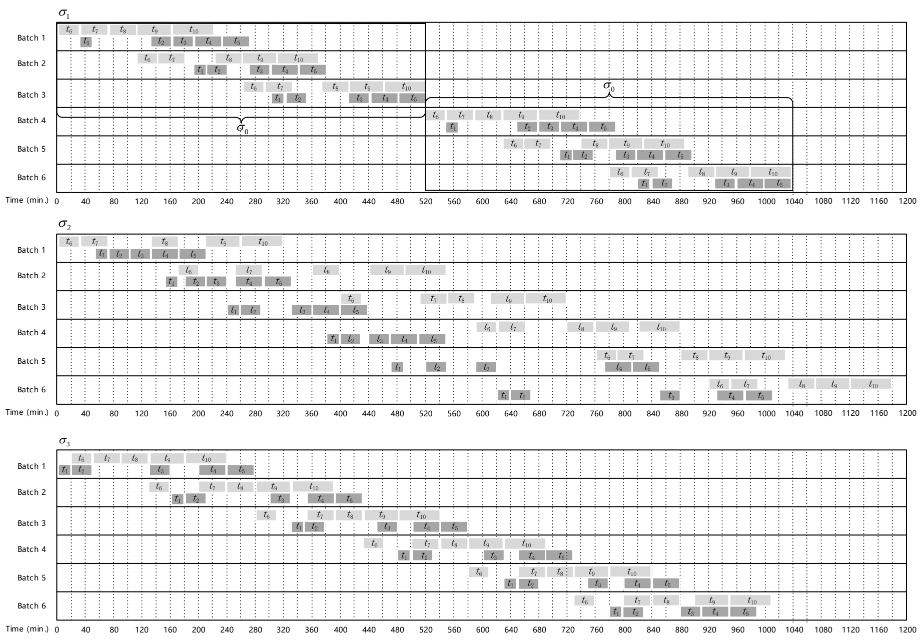

Figure 6: The Gantt chart of solutions σ_1, σ_2, and σ_3

no evidence showing that the solution σ_3 is an optimal solution.

6. Conclusion

In this paper, a systematic approach to model batch processes with timed Petri nets is proposed. Then, an improved approximation - namely ClusTERG - of the timed graphs for timed Petri nets is detailed. By constructing and clustering the information vertices of timed Petri nets, time information about a batch process is directly reflected on the ClusTERG with the maximal error not exceeding a given value λ. Such an approach facilitates to obtain schedules of the batch process by directly and globally searching the ClusTERG of the timed Petri net model.

However, the design of a $\mathcal{D}$-order that is consistent with a ClusTERG having smallest dimensions (for a given granularity parameter λ) is an open question. In addition, since there may exist some uncertain and unobservable faults in a batch process, we will further improve our method based on the ClusTERG for robust and dynamic scheduling the system in future. Finally, we also aim to consider a more realistic case study.

References

[1] T. Gu, P. A. Bahri, A survey of Petri net applications in batch processes, Computers in Industry 47 (1) (2002) 99–111.

[2] R. Zurawski, M. Zhou, Petri nets and industrial applications: A tutorial, IEEE Transactions on Industrial Electronics 41 (6) (1994) 567–583.

[3] J. Luo, M. Zhou, J. Wang, AB & B: An anytime branch and bound algorithm for scheduling of deadlock-prone flexible manufacturing systems, IEEE Transactions on Automation Science and Engineering 18 (4) (2021) 2011–2021.

[4] P. Merlin, D. Farber, Recoverability of communication protocols – Implications of a theoretical study, IEEE Transactions on Communications 24 (9) (1976) 1036–1043.

[5] M. Tittus, K. Akesson, Petri net models in batch control, Mathematical and Computer Modelling of Dynamical Systems 5 (2) (1999) 113–132.

[6] C. Kim, T. Yu, T. Lee, Reachability tree-based optimization algorithm for cyclic scheduling of timed Petri nets, IEEE Transactions on Automation Science and Engineering 18 (3) (2021) 1441–1452.

[7] C. Cassandras, S. Lafortune, Introduction to Discrete Event Systems, Second Edition, 2008.

[8] M. Ghaeli, P. Bahri, P. Lee, T. Gu, Petri-net based formulation and algorithm for short-term scheduling of batch plants, Computers and Chemical Engineering 29 (2) (2005) 249–259.

[9] G. Mejia, G. Odrey, An approach using Petri nets and improved heuristic search for manufacturing system scheduling, Journal of Manufacturing Systems 24 (2) (2005) 79–92.

[10] X. Li, K. Xing, M. Zhou, X. Wang, Y. Wu, Modified dynamic programming algorithm for optimization of total energy consumption in flexible manufacturing systems, IEEE Transactions on Automation Science and Engineering 16 (2) (2019) 691–705.

[11] G. Mejia, K. Nino, A new hybrid filtered beam search algorithm for deadlock-free scheduling of flexible manufacturing systems using Petri nets, Computers & Industrial Engineering 108 (2017) 165–176.

[12] N. Wu, M. Zhou, Z. Li, Short-term scheduling of crude-oil operations: Enhancement of crude-oil operations scheduling using a Petri net based control-theoretic approach, IEEE Robotics & Automation Magazine 22 (2) (2015) 64–76.

[13] B. Berthomieu, M. Menasche, An enumerative approach for analyzing time Petri nets, in: Proc. IFIP Congress, 1983, pp. 41–46.

[14] H. Boucheneb, H. Rakkay, A more efficient time Petri net state space abstraction useful to model checking timed linear properties, Fundamenta Informaticae 88 (4) (2008) 469–495.

[15] B. Berthomieu, F. Vernadat, State class constructions for branching analysis of time Petri nets, in: Proc. Tools and Algorithms for the Construction and Analysis of Systems, 2003, pp. 442–457.

[16] R. Hadjidj, H. Boucheneb, Efficient reachability analysis for time Petri nets, IEEE Transactions on Computers 60 (8) (2011) 1085–1099.

[17] F. Basile, M. P. Cabasino, C. Seatzu, State estimation and fault diagnosis of labeled time Petri net systems with unobservable transitions, IEEE Transactions on Automatic Control 60 (4) (2015) 997–1009.

[18] G. Gardey, O. Roux, O. Roux, Using zone graph method for computing the state space of a time Petri net, in: Proc. Formal Modeling and Analysis of Timed Systems, 2003, pp. 246–259.

[19] D. Lime, O. Roux, Model checking of time Petri nets using the state class timed automaton, Discrete Event Dynamic Systems 16 (2) (2006) 179–205.

[20] D. Lefebvre, Approximated timed reachability graphs for the robust control of discrete event systems, Discrete Event Dynamic Systems 29 (1) (2019) 31–56.

[21] D. Lefebvre, C. Daoui, Control design for bounded partially controlled TP-Ns using timed extended reachability graphs and MDP, IEEE Transactions on Systems, Man, and Cybernetics: Systems 50 (6) (2018) 2273–2283.

[22] J. Zhou, D. Lefebvre, Z. Li, A clustering approach to approximate the timed reachability graph for a class of time Petri nets, IEEE Transactions on Automatic Control (2021) 1–7doi:10.1109/TAC.2021.3110010.

[23] Standard: Batch Control–Part 1: Models and Terminology, ANSI/ISA-S88.01-1995, Instrument Society of America, 1995.

[24] Batch Control Part I: Models and Terminology, IEC 61512-1, Int. Electrotechnical Commission, 1998.

[25] P. N. Sharratt, Handbook of Batch Process Design, Springer, Blackie A& P, 1997.

[26] J. Zhou, J. Luo, D. Lefebvre, Z. Li, Modeling and scheduling methods for batch production systems based on Petri nets and heuristic search, IEEE Access 8 (2020) 163458–163471.

[27] S. Haddad, P. Moreaux, Stochastic Petri nets in Petri nets: Fundamental Models and Applications, Wiley, 2009.

[28] K. P. Girish, S. J. John, Relations and functions in multiset context, Information Sciences 179 (6) (2009) 758–768.

[29] T. Cormen, C. Leiserson, R. Rivest, Introduction to Algorithms, Second Edition, MIT Press, Cambridge, 2001.

[30] P. Berkhin, A Survey of Clustering Data Mining Techniques in Grouping Multidimensional Data, Springer-Verlag, 2006.

[31] F. Murtagh, P. Contreras, Algorithms for hierarchical clustering: an overview, Data Mining and Knowledge Discovery 2 (1) (2012) 86–97.

[32] A. Bouguettaya, Q. Yu, X. Liu, X. Zhou, A. Song, Efficient agglomerative hierarchical clustering, Expert Systems with Applications 42 (5) (2015) 2785–2797.