Symbolically Scheduling of FMS on Timed Petri Nets

Bo Huang, *Senior Member, IEEE,* Jian Yang, *Member, IEEE,* and MengChu Zhou, *Fellow, IEEE*

***Abstract*—To compactly represent and efficiently schedule flexible manufacturing systems (FMSs), this paper proposes a symbolic approach based on timed Petri nets (PNs). First, FMSs are modeled via PNs with time information on activity places. Then, the methods of functionally representing, evolving, and scheduling of such timed PNs using binary decision diagrams are formulated. To accelerate the schedule, a monotone and admissible heuristic function suitable for the symbolic method and its efficient functional implementation are given. When compared with explicit-state methods, our symbolic one can represent large sets of states with compact shared structures and allow efficient computation on those sets to find an optimal schedule. Finally, some experiments are conducted to show the effectiveness and efficiency of our method. To our knowledge, this is the first work that employs symbolic techniques to compactly represent timed PN models and fast find optimal schedules with heuristics for FMSs. This helps to advance the state-of-the-art in the modeling and scheduling scalability of FMSs.**

***Index Terms*—Binary decision diagrams, flexible manufacturing systems, heuristic scheduling, timed Petri nets.**

I. Introduction

Flexible manufacturing systems (FMSs) consist of a number of numerically controlled machines and an automated material handling system which transports parts between any pair of the machines. By integrating them, FMSs have the ability to reasonably allocate manufacturing resources so as to effectively produce different products and, thus, can achieve the better efficiency than the conventional job-shop. Although the increased flexibility of FMSs provides a great number of choices of resources and processing routings, and allows high productivity, it imposes a challenging problem, that is, the allocation of given resources to different processes and the scheduling of the sequence of activities to accomplish the best efficiency. However, such a scheduling problem belongs to be class of NP-hard ones since its computational time to obtain an optimal schedule grows exponentially with the problem size [1].

Petri Nets (PNs) can naturally model the flow of information and the control of events in a discrete event system with concurrency, synchronization, sequencing, and sharing of resources [2]. This makes PNs an ideal and popular tool to handle scheduling problems with routing flexibility and shared resources in FMSs. Lee and DiCesare [3] pioneered in combining of PN simulation capabilities and intelligent A^* search within the PN's reachability graph to schedule FMSs. Their A^* search only needs to explore a partial reachability graph with an admissible heuristic to obtain the shortest scheduling path. However, a problem observed is the state explosion [4], which means that, although not the whole reachability graph is needed to be generated, the number of the explored states still grows exponentially in the number of state variables (problem components) and makes the optimally scheduling methods only applicable to small systems. To reduce the search effort in the reachability graph, researchers have proposed several improvements for it, e.g., A^* search with controlled or limited backtracking capability [5], [6], A^* search with relaxation of the evaluation scope [7], [8], deadlock-free dynamic window search [9], and iterative deepening A^* with backtracking [10]. We have also proposed some techniques to accelerate it, such as hybrid heuristic search [11], [12], dynamic weighting strategy [13], and more informed heuristics [14], [15]. However, the above improvements are either inapplicable or "slow" for the optimal scheduling of FMSs.

Binary Decision Diagrams (BDDs) have the capability of representing large set of encoded data with small data structures and allow an efficient manipulation of those sets. For the untimed PN models, Pastor *et al*. [16] use BDDs to alleviate the inherent complexity in the generation of reachable markings and the analysis of structural and behavioral properties (such as liveness and concurrency) in untimed PN models. Chen *et al*. use BDD to efficiently compute legal markings [17] and minimal siphons [18] in untimed models. They inspire us to combine BDDs with the PN model to accelerate the scheduling of FMSs. However, FMS scheduling can never be done without time information which is not yet considered in the above methods. Thus, there is a strong interest for us to incorporate time information into BDD representations and then efficiently find an optimal scheduling sequences for FMSs.

In this paper, we present a symbolic method to optimally schedule FMSs by combining BDD representations with timed PN modeling. First, FMSs are modeled via PNs with time information on activity places. Then, the state representa-

This work was supported in part by the National Natural Science Foundation of China under Grant 61773206, in part by the Foundation of Fujian Engineering Research Center of Motor Control and System Optimal Schedule (Huaqiao University) under Grant FERC002, and in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20170131. *(Corresponding authors: B. Huang and M. C. Zhou.)*

B. Huang is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China, and also with Fujian Engineering Research Center of Motor Control and System Optimal Schedule (Huaqiao University), Xiamen 361021, China (e-mail: huangbo@njust.edu.cn).

J. Yang is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: csjyang@njust.edu.cn).

M.C. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA (e-mail: zhou@njit.edu).

tion and model evolution are symbolically formulated by using BDD characteristic functions. Based on such Boolean representations, A* search and its admissible heuristic are designed to obtain optimal schedules for the systems. The symbolic scheduling method uses efficient data-structures of BDD to reason about sets of states, instead of individual states, and executes an efficient functional exploration in the PN's reachability graph. So, it is a promising way to alleviate the state-explosion problem in the optimal schedule problem. Experiments show that it is more efficient when compared with the traditional explicit-state ones. Our work has made the following contributions:

(1) The PN models of FMSs in the literature are unified and expanded with time information for scheduling.

(2) For the timed PN models of FMSs, Boolean representations of states and functional computations of system evolution are symbolically formulated, which make the symbolic A* search feasible for FMS models.

(3) A monotone and admissible heuristic function for symbolic search is given. It guarantees the optimality of the obtained schedules and can be easily implemented.

In the sequel, Section II reviews the preliminaries of BDDs and PNs. The definitions of timed PN for FMS scheduling and the approaches of the symbolically representing, evolving, and scheduling of timed PN models of FMSs are presented in Section III. Section IV provides experimental results for some widely studied FMSs. Finally, conclusions are given in Section V.

II. Preliminaries

A. Boolean Algebra

A Boolean algebra [19] is a five-tuple $(B, +, \cdot, 0, 1)$, where B is a set called the carrier, $+$ and $\cdot$ are binary operations on B, and 0 and 1 are elements of B. The elements in B satisfy the commutative, distributive, identity, and complement laws. The algebra of subsets of a set C, denoted $(2^C, \bigcup, \bigcap, \emptyset, C)$, is a Boolean algebra, where 2^C is the set of subsets of C, and $\bigcup$ and $\bigcap$ are the union and intersection operations.

For an integer $l \geq 0$, an l-variable Boolean function is a function $f : B^l \to B$. Let $F_l(B)$ be the power set of B^l (the characteristic functions of subsets of B^l). The system $(F_l(B), +, \cdot, \mathbf{0}, \mathbf{1})$ is the Boolean algebra of Boolean functions, in which $+$ and $\cdot$ represent disjunction and conjunction of l-variable Boolean functions, respectively, and $\mathbf{0}$ and $\mathbf{1}$ represent the "*zero*" and "*one*" functions, $f(x_1, \cdots, x_l) = 0$ and $f(x_1, \cdots, x_l) = 1$, respectively.

B. Binary Decision Diagram

A binary decision diagram (BDD) [20] is a rooted, directed acyclic graph with two sink nodes, labeled 0 (called bddfalse) and 1 (called bddtrue), representing the Boolean functions $\mathbf{0}$ and $\mathbf{1}$. Each nonsink node is labeled with a Boolean variable v and has two out-edges labeled 0 and 1. It represents the Boolean functions corresponding to its 0-edge if $v = 0$ or the Boolean function corresponding to its 1-edge if $v = 1$.

A BDD is ordered (OBDD for short) if on all paths through the graph the variables respect a given linear order. An OBDD is reduced (ROBDD for short) if no two distinct nodes has identical 1-edge and 0-edge. ROBDD provides compact representations of Boolean expression and efficient algorithms for performing all kinds of logical operations, which are all based on the crucial fact that for any function $f : B^l \to B$ there is exactly one ROBDD representing it. In the following, we use "BDD" to mean "ROBDD".

C. Petri Nets

A generalized Petri net [21] is a four-tuple $N = (P, T, F, W)$ where $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places and $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions meeting $m, n \in \mathbb{N}^+ = \{1, 2, \ldots\}$, $P \cup T \neq \emptyset$, and $P \cap T = \emptyset$. $F \subseteq (P \times T) \cup (T \times P)$ is called the flow relation or directed arcs between P and T. $W : (P \times T) \cup (T \times P) \to \mathbb{N} = \mathbb{N}^+ \cup \{0\}$ is a mapping that assigns a weight to an arc. $M : P \to \mathbb{N}$ is called a marking that assigns tokens to each place $p \in P$. A net N is said to be an ordinary one if $\forall (x, y) \in F$, $W(x, y) = 1$. An ordinary net is a state machine if $|{}^\bullet t| = |t^\bullet| = 1$. A net N is said to be pure or self-loop free if $\forall p_x \in P$ and $\forall t_y \in T$, $W(p_x, t_y) \cdot W(t_y, p_x) = 0$. A pure net can be represented by an incidence matrix $[N] = [N^+] - [N^-]$ where $[N^+] = \{W(t_y, p_x)\}$ and $[N^-] = \{W(p_x, t_y)\}$.

Given a place $p \in P$, ${}^\bullet p = \{t \in T | (t, p) \in F\}$ (resp. $p^\bullet = \{t \in T | (p, t) \in F\}$) is called the set of pre-transitions (resp. post-transitions) of p. Similarly, given a transition ${}^\bullet t = \{p \in P | (p, t) \in F\}$ (resp. $t^\bullet = \{p \in P | (t, p) \in F\}$) is called the set of pre-places (resp. post-places) of t. A transition $t \in T$ is said to be enabled at a marking M if $\forall p \in {}^\bullet t, M(p) \geqslant W(p, t)$. Firing an enabled transition t yields a new marking M' such that $\forall p \in P$, $M'(p) = M(p) - W(p, t) + W(t, p)$. This process is denoted by $M[t\rangle M'$. A marking M is reachable from the initial marking M_0 if there exist a sequence of transition firings from M_0 to M. The pair (N, M_0) is called a net system and $R(N, M_0)$ is called the set of all markings reachable from M_0 in N. A net system (N, M_0) is bounded if $\forall M \in R(N, M_0)$, $\forall p \in P$, $\exists k \in \mathbb{N}^+$, $M(p) \leq k$. If a place $p \in P$ is bounded by some k for which $\forall M \in R(N, M_0)$, $M(p) \leq k$, we say that p is k-bounded. (N, M_0) is k-bounded if each place of the net is k-bounded. A reachability graph $G(N, M_0)$ is a directed graph where vertices are markings in $R(N, M_0)$ and each edge connects two markings M and M', which is denoted as $(M, M') \in G(N, M_0)$ and satisfies that $\exists t \in T$, the firing of t leads the system from M to M'. In such a case, M is called an immediate predecessor of M' and M' an immediate successor of M in $G(N, M_0)$.

Given a net system (N, M_0), $t \in T$ is live at M_0 if $\forall M \in R(N, M_0)$, $\exists M' \in R(N, M)$, t is enabled at M'. (N, M_0) is live if $\forall t \in T$, t is live at M_0. (N, M_0) is said to be deadlock-free if at least one transition is enabled at any marking in $R(N, M_0)$. (N, M_0) is said to be reversible if $\forall M \in R(N, M_0)$, $M_0 \in R(N, M)$. Let $[0]$ be a column vector whose every entry equals 0. A place vector $I : P \to \mathbb{Z}$ (the set of integers) is called a P-invariant (PI) if $I \neq [0]$ and $I^T[N] = [0]^T$. Let I be a PI of (N, M_0), then $\forall M \in R(N, M_0)$, $I^T M = I^T M_0$. I is called a P-semiflow if all of its elements are nonnegative. Let $\|I\| = \{p \in P | I(p) \neq 0\}$ be

the support of I. I is called a minimal PI if its support is not a superset of the support of any other PI and its elements are coprime.

III. Symbolically Scheduling of P-timed PNs

A. P-timed PNs for FMS Scheduling

In the literature, several kinds of PNs have been proposed to model FMSs, for example, PPN [22], S^{3}PR [23], ES^{3}PR [24], LS^{3}PR [25], ELS^{3}PR [26], GLS^{3}PR [27], S^{2}LSPR [28], S*PR [29], S^{4}PR [30], S^{3}PGR^2 [31], and S^{3}PMR [32]. They have the restrictions that each operation requires just one resource and/or two successive operations need different resource types. In addition, they have no time information which is important to scheduling. For example, in an event sequence of them, a transition must be fired first and then another transition is fired, but no indication exists to show how much time has elapsed between the firings of these two transitions. The most common way of introducing time into a PN model is through the use of timed PN. Timed PNs work like the untimed ones, but include a new function defined on either the transitions or the places, which correspond to T-timed PNs [8] and P-timed ones [5], respectively. Such a function indicates the amount of time required for particular transitions to fire in T-timed PNs or the amount of time that must elapse between the arrival of a token in a place and when it is allowed to take part in enabling and firing another transition in P-timed PNs. Either method can be used, but P-timed PNs can avoid certain ambiguities in the meaning of sate/event constraints and seamlessly implement standard supervisory control constraints [33]. In the sequel, we unify the nets in [22]- [32] and revise them as a P-timed PN without the above restrictions and suitable for modeling and scheduling of FMSs.

Definition 1: A P-timed PN $\mathcal{N} = (N, D) = (P_S \cup P_E \cup P_R \cup P_A, T, F, W, D) \in \Phi$, which models FMS for scheduling, is defined as the composition of a set of nets $\mathcal{N}^x = (\{p_s^x\} \cup \{p_e^x\} \cup P_R^x \cup P_A^x, T^x, F^x, W^x, D^x)$ sharing some common places, i.e., $\mathcal{N} = \bigcirc_{x\in\mathbb{N}_S}\mathcal{N}^x$ where $\mathbb{N}_S = \{x \in \mathbb{N}^+ | \mathcal{N}^x$ is the xth production sequence of the FMS system.$\}$ and the following statements are true.

1) p_s^x is called the start place and p_e^x is called the end one of $\mathcal{N}^x$. P_A^x (resp. P_R^x) is called the set of activity (resp. resource) places of $\mathcal{N}^x$.

2) $P_A^x \neq \emptyset$; $P_R^x \neq \emptyset$; $p_s^x \notin P_A^x$; $p_e^x \notin P_A^x$; $(\{p_s^x\} \cup \{p_e^x\} \cup P_A^x) \cap P_R^x = \emptyset$.

3) $W = W_A \cup W_R$, where $W_A : ((P_A \cup P_S \cup P_E) \times T) \cup (T \times (P_A \cup P_S \cup P_E)) \to \{0, 1\}$ such that $\forall x, y \in \mathbb{N}_S$ with $x \neq y$, $((P_A^y \cup \{p_s^y\} \cup \{p_e^y\}) \times T^x) \cup (T^x \times (P_A^y \cup \{p_s^y\} \cup \{p_e^y\})) \to \{0\}$, and $W_R : (P_R \times T) \cup (T \times P_R) \to \mathbb{N}$.

4) $\forall r \in P_R$, $\exists$ a unique minimal P-semiflow I_r such that $\|I_r\| \cap P_R = \{r\}$, $\|I_r\| \cap P_0 = \emptyset$, $\|I_r\| \cap P_A \neq \emptyset$, and $I_r(r) = 1$. In addition, $P_A = \bigcup_{r\in P_R}(\|I_r\| \setminus r)$.

5) $\forall r \in P_R^x$, ${}^{\bullet\bullet}r \cap P_A^x = r^{\bullet\bullet} \cap P_A^x \neq \emptyset$, ${}^{\bullet}r \cap r^{\bullet} = \emptyset$.

6) If the places in P_R^x and their connected arcs are removed, then the resultant net $N^{x\prime}$ is a state machine.

7) $\mathcal{N}^x$ and $\mathcal{N}^y (x \neq y)$ are composable when they share a set of common places which must be resource places.

8) For any $r \in P_R$, the support of a minimal P-semiflow I_r without r, i.e., $H(r) = \|I_r\| \setminus \{r\}$, is called the set of holders of r, and $\forall M \in R(N, M_0)$, $M_0(r) = \sum_{p\in\{r\}\cup H(r)} I_r(p) \cdot M(p)$ is called the capacity of r, which is denoted as $K(r)$.

9) For any $p \in P_A$, a $|P_R|$-dimensional vector $U_p(i), i = 1, \ldots, |P_R|$ is called the resource requirement of p, which indicates how many units of $r_i \in P_R$ are required to support the operation of p.

10) $D : P_A \to \mathbb{N}$ is a mapping that assigns processing time to each activity place.

11) An initial marking M_0 is called an acceptable one for $\mathcal{N}$ if: 1) $\forall x \in \mathbb{N}_S$, $M_0(p_s^x) \geq 1$ and $M_0(p_e^x) = 0$; 2) $\forall p \in P_A$, $M_0(p) = 0$; and 3) $\forall r \in P_R$, $M_0(r) \geq \max_{p\in\|I_r\|} I_r(p)$.

The PN models defined in [22]- [32] can be revised to meet Definition 1 for scheduling by dividing each idle place into a start place and a end one and adding processing time to each activity place.

B. Timed PN Evolution with BDD

This section shows how to symbolically model the evolution of a P-timed net $\mathcal{N} \in \Phi$ via Boolean algebras. First, we give the representations of two relations between Boolean variables. Suppose that a and b are respectively represented by the sets of Boolean variables $a^0, a^1, \ldots,$ and a^{k_a} and $a^0, a^1, \ldots,$ and a^{k_b} with $k_a = k_b$. Then, a is equivalent to b can be represented as follows:

$$\begin{aligned} &a \bowtie b \equiv \\ &(a^{k_a} \equiv b^{k_a}) \cdot (a^{k_a-1} \equiv b^{k_a-1}) \cdots (a^1 \equiv b^1) \cdot (a^0 \equiv b^0) \\ &\equiv \prod_{i=0}^{k_a} (a^i \equiv b^i). \end{aligned} \tag{1}$$

The relation, a is greater than or equal to b, can be represented as below:

$$\begin{aligned} &a \trianglerighteq b \equiv (a^{k_a} > b^{k_a}) + \\ &(a^{k_a} \equiv b^{k_a}) \cdot (a^{k_a-1} > b^{k_a-1}) + \\ &\cdots\cdots \\ &(a^{k_a} \equiv b^{k_a}) \cdot (a^{k_a-1} \equiv b^{k_a-1}) \cdots (a^1 \equiv b^1) \cdot (a^0 > b^0) + \\ &(a^{k_a} \equiv b^{k_a}) \cdot (a^{k_a-1} \equiv b^{k_a-1}) \cdots (a^1 \equiv b^1) \cdot (a^0 \equiv b^0) \\ &\equiv \sum_{i=0}^{k_a} \left[(a^i > b^i) \cdot \prod_{j=i+1}^{k_a} (a^i \equiv b^i) \right] + \prod_{i=0}^{k_a} (a^i \equiv b^i). \end{aligned} \tag{2}$$

For a PN model, the number of tokens $M(p)$ in a place p at a marking M, which is up to k, can be represented by a set of Boolean variables, $p^0, p^1, \ldots,$ and p^{k_p}. If such a binary encoding scheme is used, the number of Boolean variables required for a k-bounded place is $\lceil \log_2(k+1) \rceil$. For example, a place p_1 which contains two tokens can be represented by a vector of Boolean variables $p_1^1 \bar{p}_1^0$. To model the transition relations of a PN model via Boolean variables, the weight of an arc is also required to be encoded with the same type of encoding. The weight of an arc $W(p, t)$ is represented by the same number of binary encoded Boolean variables $\omega^0, \omega^1, \ldots,$ and ω^{k_p}. Then, the condition under which a transition t is enabled can be defined by the following characteristic function

$\mathcal{E}_t$

$$\prod_{p\in{}^\bullet t}[M(p)\trianglerighteq W(p,t)]\equiv \prod_{p\in{}^\bullet t}\left[\sum_{i=0}^{k_p}\left[(p^i>\omega^i)\cdot\prod_{j=i+1}^{k_p}(p^i\equiv\omega^i)\right]+\prod_{i=0}^{k_p}(p^i\equiv\omega^i)\right]. \tag{3}$$

However, Function (3) cannot ensure that t is enabled at a marking M for a P-timed net $\mathcal{N}\in\Phi$. In such a net, if a token arrives at an activity place p_i at time τ, it will be available at $\tau+D(p_i)$. Let τ_{clock} be the current clock time. We use $\max\{0,\tau+D(p_i)-\tau_{\text{clock}}\}$ to denote the remaining time of the token in p_i. If the remaining time is nonzero, the token is unavailable to enable any transition. So, in a P-timed net $\mathcal{N}\in\Phi$, we should consider not only the token distribution but also the remaining time of tokens in activity places for a marking. Suppose that $\forall p\in P_A$, p is λ-bounded. In the sequel, we use $\mathcal{S}=(M,R_1,\ldots,R_\lambda)$ to denote a marking of a net $\mathcal{N}\in\Phi$ where $R_u(u\in\{1,\ldots,\lambda\}):P_A\to\mathbb{N}$ is a vector of the remaining processing time of the uth token in each activity place. For example, the initial marking $\mathcal{S}_0$ means that $M=M_0$ and $\forall u\in\{1,\ldots,\lambda\}, R_u=[0]$ since no token exists in any activity place.

Let $G(\mathcal{N},\mathcal{S}_0)$ be the reachability graph of a net $\mathcal{N}\in\Phi$. Then, each edge $(\mathcal{S},\mathcal{S}')$ in $G(\mathcal{N},\mathcal{S}_0)$ leads the system from the marking $\mathcal{S}$ to $\mathcal{S}'$ by firing a transition with a time delay. This is because, before the transition t can be fired, the needed tokens in each of its activity pre-place must be delayed for at least the time associated with the activity place. Hence, the cost of an edge $c(\mathcal{S},\mathcal{S}')$, which means the time needed in the system transfer from $\mathcal{S}$ to $\mathcal{S}'$, can be intuitively considered to be the maximum time delay of all the activity pre-places of the transition, that is,

$$c(\mathcal{S},\mathcal{S}')=\max_{p_i\in{}^\bullet t\cap P_A}\{D(p_i)\}. \tag{4}$$

But this intuitive argument is not quite correct. Let $D_{\max}$ be the maximum value in Equation (4). When the marking $\mathcal{S}$ converts to $\mathcal{S}'$, tokens in the activity places other than the pre-places of t are also concurrently delayed by $D_{\max}$. This is true for every marking update. In addition, for a net $\mathcal{N}\in\Phi$, we have $(P_A\times T)\cup(T\times P_A)\to\{0,1\}$, that is, the firing of any transition needs only one token from each of its activity pre-places. Such a token should have the least remaining processing time among the tokens n the place. So, the time spent on the system transfer from $\mathcal{S}$ to $\mathcal{S}'$, $c(\mathcal{S},\mathcal{S}')$, is not really the maximum processing time of all the pre-places of t, but rather the maximum of the least remaining processing time of tokens in each activity pre-place of t, that is,

$$c(\mathcal{S},\mathcal{S}')=\max_{p_i\in{}^\bullet t\cap P_A}\left\{\min_{u\in\{1,\ldots,\lambda\}}\{R_{i,u}\}\right\} \tag{5}$$

where $R_{i,u}$ denotes $R_u(p_i)$ the remaining time of the uth token in p_i for simplicity. Therefore, the condition under which the transition t satisfying (3) is not enabled due to the remaining processing time is given as the following characteristic function $\widetilde{\mathcal{F}}_t$

$$\sum_{p_i\in{}^\bullet t\cap P_A}\sum_{v=1}^{\lambda}\left[[M(p_i)\bowtie v]\cdot\prod_{u=1}^{v}[R_{i,u}\trianglerighteq 1]\right]. \tag{6}$$

This condition means that for transition t, there exist at least one activity pre-place whose tokens are all unavailable due to nonzero remaining precessing time. Thus, for a transition t satisfying (3), if t is not enabled since (6) is met, we should wait until the remaining processing time of the tokens needed for enabling t is decreased to zero. More importantly, as time goes by, the remaining time of any token in activity places will also be decreased if it is greater than zero. Algorithm 1 gives the implementation of such a time decreasing in the form of Boolean manipulations.

Algorithm 1 Decreases remaining time of tokens in activity places in the preparation of firing t.

Input: A transition t and a BDD X representing a set of markings satisfying (3).
Output: A BDD representing a set of markings ready to fire t in both token distribution and remaining time.

1: **function** $\mathrm{S}_t(X)$
2: $Z_1:=X\cdot\widetilde{\mathcal{F}}_t$;
3: $Y:=X\setminus Z_1$; /*Set difference*/
4: **while** $Z_1\neq$ bddfalse **do**
5: **for all** i such that $p_i\in P_A$ and $D(p_i)>0$ **do**
6: **for** $u:=1$ to λ **do**
7: $Z_2:=Z_1\cdot(R_{i,u}\trianglerighteq 1)$;
8: **if** $Z_2\neq$ bddfalse **then**
9: $Z_1:=Z_1\setminus Z_2$;
10: $Z_2:=\exists_{r_{i,u}}\prod_{j:=0}^{k_{r_{i,u}}}\left[s_{i,u}^j\equiv\beta^j(r_{i,u})\right]\cdot Z_2$;
11: $Z_2[s\leftrightarrow r]$;
12: $Z_1:=Z_1+Z_2$;
13: **end if**
14: **end for**
15: **end for**
16: $Y:=Y+\left[Z_1\setminus(Z_1\cdot\widetilde{\mathcal{F}}_t)\right]$;
17: $Z_1:=Z_1\cdot\widetilde{\mathcal{F}}_t$;
18: **end while**
19: **return** Y;
20: **end function**

In Algorithm 1, $[s\leftrightarrow r]$ denotes the swapping of the sets of variables to allow multiple quantification, $R_{i,u}$ is represented by a set of Boolean variables $r_{i,u}^0, r_{i,u}^1, \ldots,$ and $r_{i,u}^{k_{r_{i,u}}}$, and β denotes the symbolic function of the subtraction of one, which is given as below:

$$\beta^j(r_{i,u})=\begin{cases}\bar{r}_{i,u}^j, & \text{if } j=0\\ r_{i,u}^j\bigoplus B_{j-1}, & \text{if } j\geq 1\end{cases} \tag{7}$$

where

$$B_j=\begin{cases}\bar{r}_{i,u}^j, & \text{if } j=0\\ \bar{r}_{i,u}^j\cdot B_{j-1}, & \text{if } j\geq 1.\end{cases} \tag{8}$$

At any marking obtained by Algorithm 1, t is ready for firing in both the token distribution and the token availability. Then, we handle the token moving during the firing. In a place $p_i \in P$, how the content of the jth variable encoding the tokens in p_i is transformed in the firing of t can be represented as below:

$$\delta^{t^j}(p_i) = \begin{cases} p_i^j \bigoplus \omega^j \bigoplus B_{j-1}, & \text{if } p_i \in {}^\bullet t \\ p_i^j \bigoplus \omega^j \bigoplus C_{j-1}, & \text{if } p_i \in t^\bullet \\ p_i^j, & \text{otherwise} \end{cases} \tag{9}$$

where C_j and B_j are respectively the carry and borrow functions as follows:

$$C_j = \begin{cases} p_i^j \cdot \omega^j + C_{j-1} \cdot (p_i^j \bigoplus \omega^j), & \text{if } j \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

$$B_j = \begin{cases} \bar{p}_i^j \cdot \omega^j + B_{j-1} \cdot (p_i^j \equiv \omega^j), & \text{if } j \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

$$\omega^j = \begin{cases} W^j(p_i, t), & \text{if } p_i \in {}^\bullet t \\ W^j(t, p_i), & \text{if } p_i \in t^\bullet. \end{cases} \tag{12}$$

The transition relation of tokens in places can be defined as:

$$\mathbb{R}_t(p_1, \ldots, p_n, q_1, \ldots, q_n) = \prod_{i=1}^{m} \prod_{j=0}^{k_{p_i}} \left[q_i^j \equiv \delta^{t^j}(p_i) \right] \tag{13}$$

where m denotes the number of places in the net. After the token moving, we give each new token in the activity post-places of t a correct processing time. Such Boolean manipulations is given in Algorithm 2, which adds the processing time to a token whose remaining time is zero in each activity post-place of the fired transition.

Therefore, the image function that expands a set of markings to generate their successors through the firing of t can be given as the follows:

$$\text{Img}(\text{PN}, \mathbb{S}) = \exists_{p_1, \ldots, p_n} \sum_{t \in T} \text{A}_t \left(\text{S}_t(\mathbb{S} \cdot \mathcal{E}_t) \cdot \prod_{i=1}^{m} \prod_{j=0}^{k_{p_i}} \left[q_i^j \equiv \delta^{t^j}(p_i) \right] [q \leftrightarrow p] \right) \tag{14}$$

where $\mathbb{S}$ denotes a BDD representing a set of markings and $[q \leftrightarrow p]$ represents the swap of the sets of variables to allow the multiple quantification after the token moving.

C. Symbolic Heuristic Search

In this subsection, we show the steps of symbolic A* search for a net $\mathcal{N} \in \Phi$. A monotone and admissible heuristic function and its efficient implementation are also given.

To implement the symbolic A* search, we extend the aforementioned marking to a scheduling state $\mathcal{S} = (f, M, R_1, \ldots, R_\lambda)$ with $f(\mathcal{S}) = g(\mathcal{S}) + h(\mathcal{S})$ where $f(\mathcal{S})$ is an estimate of the cost or makespan from $\mathcal{S}_0$ to $\mathcal{S}_G$ along an optimal path which goes through $\mathcal{S}$, $g(\mathcal{S})$ is the current lowest cost or makespan from $\mathcal{S}_0$ to $\mathcal{S}$, and $h(\mathcal{S})$ is a heuristic estimate of the cost or makespan from $\mathcal{S}$ to $\mathcal{S}_G$ along an optimal path. For a P-timed net $\mathcal{N} \in \Phi$, a transition $t \in T$ is said to be enabled at a state $\mathcal{S}$ if $\forall p \in {}^\bullet t$, $M(p) \geqslant W(p, t)$ and $\forall p \in {}^\bullet t \cap P_A$, p has at least one token whose remaining time is zero. Firing an enabled transition t yields a new state $\mathcal{S}'$ such that $g(\mathcal{S}') = g(\mathcal{S}) + c(\mathcal{S}, \mathcal{S}')$, $M(\mathcal{S}')[t\rangle M(\mathcal{S})$, $\forall p_i \in P$, $\forall u \in \{1, \ldots, \lambda\}$, $R_{i,u}(\mathcal{S}') = \max\{0, R_{i,u}(\mathcal{S}) - c(\mathcal{S}, \mathcal{S}')\}$, and $\forall p \in t^\bullet \cap P_A$, there exists a new token in p at $\mathcal{S}'$ whose remaining time is $D(p)$. Such a process is denoted by $\mathcal{S}[t\rangle\mathcal{S}'$ where $\mathcal{S}$ is called an immediate predecessor of $\mathcal{S}'$ and $\mathcal{S}'$ an immediate successor of $\mathcal{S}$ in the reachability graph $G(\mathcal{N}, \mathcal{S}_0)$. A state $\mathcal{S}$ is reachable from the initial state $\mathcal{S}_0$ if there exist a sequence of transition firings from $\mathcal{S}_0$ to $\mathcal{S}$ in $G(\mathcal{N}, \mathcal{S}_0)$. The steps of symbolic A* to minimize the makespan for a net system $(\mathcal{N}, \mathcal{S}_0)$ with $\mathcal{N} \in \Phi$ are shown in Algorithm 3.

Algorithm 3 is different from the explicit-state A* search in [3] since it operates on a set of states denoted by a BDD instead of a single state. The BDD evaluates to bddtrue for the binary representation of a given state if and only if the state is a member of that set. $Open$ and $Closed$ are two BDDs which represent the set of states which have been generated and not yet expanded and the set of states ever expanded, respectively. The algorithm determines all successors of the states which have the minimum f-value in BDD $Open$ via an image operation. When a goal state is reached, by keeping track in the BDD $Closed$, a legal sequence of states linking the initial state to the goal one can be extracted, which in turn can be used to find a corresponding sequence of transition firings (actions). The goal state $\mathcal{S}_G$ is on the optimal path. A state

Algorithm 2 Adds processing time to the new tokens in the activity post-places of t.

Input: A transition t and a BDD X representing a set of markings.

Output: A BDD representing a set of markings with the correct remaining time of the new tokens in the activity post-places of t.

```
1: function A_t(X)
2:     for all i such that p_i ∈ t• ∩ P_A and D(p_i) > 0 do
3:         for u := λ to 1 do
4:             C_1 := [R_{i,u} ⋈ 0];
5:             C_2 := bddtrue;
6:             for v := 1 to u − 1 do
7:                 C_2 := C_2 · [R_{i,v} ⊵ 1];
8:             end for
9:             Z := X · C_1 · C_2;
10:            if Z ≠ bddfalse then
11:                X := X \ Z;
12:                for j := 0 to k_{r_{i,u}} do
13:                    Z|_{r^j_{i,u} := D(p_i)^j} ; /*Substitution by the processing time of p_i.*/
14:                end for
15:                X := X + Z;
16:            end if
17:        end for
18:    end for
19:    return X;
20: end function
```

Algorithm 3 Symbolic A* search for a net $\mathcal{N} \in \Phi$.

Input: A net $\mathcal{N} \in \Phi$, its initial state $\mathcal{S}_0$ and goal state $\mathcal{S}_G$.

Output: A transition firing sequence with the minimal makespan.

1: $Open(\mathcal{S}) := (f \bowtie \mathcal{S}_0.f) \cdot \prod_{p_i \in P} [M(p_i) \bowtie \mathcal{S}_0.M(p_i)] \cdot \prod_{p_i \in P_A} \prod_{u:=1}^{\lambda} (R_{i,u} \bowtie \mathcal{S}_0.R_{i,u})$;
2: $Closed(\mathcal{S})$:= bddfalse;
3: **loop**
4: **if** $Open(\mathcal{S})$ = bddfalse **then**
5: **return** "Exploration completed with failure!";
6: **end if**
7: $f_{\min} := \min\{f' | Open(\mathcal{S}) \cdot (f \bowtie f') \neq \text{bddfalse}\}$;
8: $Front(\mathcal{S}) := Open(\mathcal{S}) \cdot (f \bowtie f_{\min})$;
9: **if** $Front(\mathcal{S}) \cdot \prod_{p_i \in P} [M(p_i) \bowtie \mathcal{S}_G.M(p_i)] \neq$ bddfalse **then**
10: **return** Construct$(Closed(\mathcal{S}), \mathcal{S}_0, \mathcal{S}_G)$;
11: **end if**
12: $Succ(\mathcal{S}) :=$ Img(PN, $Front(\mathcal{S})$);
13: $Closed(\mathcal{S}) := Closed(\mathcal{S}) + Front(\mathcal{S})$;
14: $Succ(\mathcal{S}) := Succ(\mathcal{S}) \setminus Closed(\mathcal{S})$;
15: $Open(\mathcal{S}) := Open(\mathcal{S}) \setminus Front(\mathcal{S})$;
16: $Open(\mathcal{S}) := Open(\mathcal{S}) + Succ(\mathcal{S})$;
17: **end loop**

in an optimal path that comes before $\mathcal{S}_G$ must be contained in $Closed$. Therefore, by intersecting the predecessors of the goal state with $Closed$, all states that are in the intersections are in an optimal path from $\mathcal{S}_0$ to $\mathcal{S}_G$, so any of these states can be chosen to continue solution reconstruction until $\mathcal{S}_0$ is found. The construction of an optimal path from $\mathcal{S}_0$ to $\mathcal{S}_G$ is presented in the Appendix where a transition $t \in T$ is said to be reversely enabled at a state $\mathcal{S}$ if $\forall p \in t^\bullet$, $M(p) \geqslant W(t,p)$ and $\forall p \in t^\bullet \cap P_A$, p has at least one token whose remaining time equals $D(p)$.

However, the heuristic functions h proposed for explicit-state A* [3], [5], [7], [8], [14] are not suitable for the symbolic one since they are dedicated to a single state, not for a BDD which implicitly represents a set of states of the system. Here, we formulate a heuristic function which can be applied to the symbolic A* search. Let $P_{\text{Alt}} \subseteq P_A$ be the set of activity places whose operations may not be performed due to the alternative operation routes in a net $\mathcal{N} \in \Phi$. For a resource places $r \in P_R$, let $H_l(r) = \{p | p \in H(r) \wedge p \notin P_{\text{Alt}}\}$ be the set of its loyal holders whose operations will be definitely performed with r. Let $r_{\max}$ be a resource place whose sum of processing time of its loyal holders is the maximum among the 1-bounded resource places $P_R^1 = \{p | p \in P_R \wedge K(p) = 1\}$. Then, a heuristic function of the symbolic A* search for a net $\mathcal{N} \in \Phi$ can be given as follows:

$$h_{\text{BDD}}(\mathcal{S}) = \begin{cases} 0, & \text{if } P_R^1 = \emptyset \\ \xi_{r_{\max}}, & \text{if } P_R^1 \neq \emptyset \text{ and } \mathcal{S} = \mathcal{S}_0 \\ h_{\text{BDD}}(\mathcal{S}') - \delta_{r_{\max}}(\mathcal{S}', \mathcal{S}), & \text{otherwise} \end{cases} \tag{15}$$

where $\mathcal{S}'$ is an immediate predecessor of $\mathcal{S}$ in $G(\mathcal{N}, \mathcal{S}_0)$, $\delta_{r_{\max}}(\mathcal{S}', \mathcal{S})$ is the time spent by a loyal holder of $r_{\max}$ in the state transfer from $\mathcal{S}'$ to $\mathcal{S}$, and $\xi_{r_{\max}}$ is the weighted sum of processing time of the loyal holders of $r_{\max}$, i.e.,

$$\xi_{r_{\max}} = \sum_{p \in H_l(r_{\max})} [D(p) \cdot M_0(p_s^x)] \tag{16}$$

where p_s^x is the start place of the production sequence $\mathcal{N}^x$ such that $p \in \mathcal{N}^x$. Next, we show that the heuristic function has some important properties.

Definition 2: [4] A heuristic function h is said to be monotone if $\forall(\mathcal{S}', \mathcal{S}) \in G(\mathcal{N}, \mathcal{S}_0),\ h(\mathcal{S}') \leq c(\mathcal{S}', \mathcal{S}) + h(\mathcal{S})$ and $h(\mathcal{S}_G) = 0$.

Definition 3: [4] A heuristic function h is said to be admissible if $\forall \mathcal{S} \in R(\mathcal{N}, \mathcal{S}_0),\ h(\mathcal{S}) \leq h^*(\mathcal{S})$ where $h^*(\mathcal{S})$ is the cheapest cost of paths from $\mathcal{S}_0$ to $\mathcal{S}$ in $G(\mathcal{N}, \mathcal{S}_0)$.

Theorem 1: The heuristic function h_{BDD} is monotone and admissible.

Proof: If $P_R^1 = \emptyset$, h_{BDD} is obviously monotone and admissible according to the above definitions. In the following, we consider the situation when $P_R^1 \neq \emptyset$.

(Monotonicity) $\forall(\mathcal{S}', \mathcal{S}) \in G(\mathcal{N}, \mathcal{S}_0)$, we have $h_{\text{BDD}}(\mathcal{S}') = \delta_{r_{\max}}(\mathcal{S}', \mathcal{S}) + h_{\text{BDD}}(\mathcal{S})$ according to (15). Since there may exist some resources other than $r_{\max}$ used in the state transfer from $\mathcal{S}'$ to $\mathcal{S}$, the time needed by such a transfer is larger than or equals the time spent on $r_{\max}$ during the transfer, i.e., $\delta_{r_{\max}}(\mathcal{S}', \mathcal{S}) \leq c(\mathcal{S}', \mathcal{S})$. So, $h_{\text{BDD}}(\mathcal{S}') \leq c(\mathcal{S}', \mathcal{S}) + h_{\text{BDD}}(\mathcal{S})$ holds. On the other hand, $h_{\text{BDD}}(\mathcal{S}_0)$ is the sum of processing time of all operations which will be definitely performed with $r_{\max}$. In addition, such operations with $r_{\max}$ will be performed without overlap since $K(r_{\max}) = 1$. So, when the system reaches $\mathcal{S}_G$, such a sum of processing time will be just exhausted. Thus, we have $h_{\text{BDD}}(\mathcal{S}_G) = 0$. Therefore, h_{BDD} is monotone.

(Admissibility) We first prove that $\forall \mathcal{S} \in R(\mathcal{N}, \mathcal{S}_0)$, $\forall \mathcal{S}'' \in R(\mathcal{N}, \mathcal{S})$,

$$h_{\text{BDD}}(\mathcal{S}) \leq c^*(\mathcal{S}, \mathcal{S}'') + h_{\text{BDD}}(\mathcal{S}'') \tag{17}$$

where $c^*(\mathcal{S}, \mathcal{S}'')$ is the cost or the time spent on the cheapest path from $\mathcal{S}$ to $\mathcal{S}''$ in $G(\mathcal{N}, \mathcal{S}_0)$. It is proven by induction on the depth-k successors of $\mathcal{S}$ in $G(\mathcal{N}, \mathcal{S}_0)$. $\forall \mathcal{S} \in R(\mathcal{N}, \mathcal{S}_0)$, $\forall \mathcal{S}'' \in R(\mathcal{N}, \mathcal{S})$, if $\mathcal{S}''$ is an immediate successor of $\mathcal{S}$, then $h_{\text{BDD}}(\mathcal{S}) = \delta_{r_{\max}}(\mathcal{S}, \mathcal{S}'') + h_{\text{BDD}}(\mathcal{S}'') \leq c^*(\mathcal{S}, \mathcal{S}'') + h_{\text{BDD}}(\mathcal{S}'')$ since $\delta_{r_{\max}}(\mathcal{S}, \mathcal{S}'')$ is the time which will be definitely spent on operations with resource $r_{\max}$ in the state transfer from $\mathcal{S}$ to $\mathcal{S}''$. Suppose that if $\mathcal{S}''$ is a depth-n successor of $\mathcal{S}$, $h_{\text{BDD}}(\mathcal{S}) \leq c^*(\mathcal{S}, \mathcal{S}'') + h_{\text{BDD}}(\mathcal{S}'')$ holds. Then if $\mathcal{S}''$ is a depth-$(n+1)$ successor of $\mathcal{S}$, we have that

$$\begin{aligned} h_{\text{BDD}}(\mathcal{S}) &= \delta_{r_{\max}}(\mathcal{S}, \mathcal{S}''') + h_{\text{BDD}}(\mathcal{S}''') \\ &\leq c^*(\mathcal{S}, \mathcal{S}''') + h_{\text{BDD}}(\mathcal{S}''') \\ &\leq c^*(\mathcal{S}, \mathcal{S}''') + c^*(\mathcal{S}''', \mathcal{S}'') + h_{\text{BDD}}(\mathcal{S}'') \\ &\leq c^*(\mathcal{S}, \mathcal{S}'') + h_{\text{BDD}}(\mathcal{S}'') \end{aligned}$$

where $\mathcal{S}'''$ is the immediate successor of $\mathcal{S}$ in a path from $\mathcal{S}$ to $\mathcal{S}''$ in $G(\mathcal{N}, \mathcal{S}_0)$. Thus, (17) is confirmed. Then, we replace $\mathcal{S}''$ in (17) by the goal state $\mathcal{S}_G$, obtaining $\forall \mathcal{S} \in R(\mathcal{N}, \mathcal{S}_0)$, $h_{\text{BDD}}(\mathcal{S}) \leq c^*(\mathcal{S}, \mathcal{S}_G) + h_{\text{BDD}}(\mathcal{S}_G)$. Now, since $h_{\text{BDD}}(\mathcal{S}_G) = 0$

and $c^*(\mathcal{S}, \mathcal{S}_G) = h^*_{\text{BDD}}(\mathcal{S})$, we have $h_{\text{BDD}}(\mathcal{S}) \leq h^*_{\text{BDD}}(\mathcal{S})$. Therefore, h_{BDD} is admissible. ■

Optimality and completeness of the symbolic A* given in Algorithm 3 are inherited from the fact that an explicit-state A* with an admissible heuristic will find an optimal solution [4], [11]. Therefore, Algorithm 3 with h_{BDD} will terminate with an optimal solution when one exists. In the following, we present an implementation of h_{BDD} which only slightly revise a function in the symbolic A* search.

For a state $\mathcal{S}'$, we have $f(\mathcal{S}') = g(\mathcal{S}')+h(\mathcal{S}')$. Since we can access the f-value, but usually not the g-value and h-value, the new f-value of a successor $\mathcal{S}$ of $\mathcal{S}'$ can be calculated as the following way:

$$\begin{aligned} f(\mathcal{S}) &= g(\mathcal{S}) + h(\mathcal{S}) = g(\mathcal{S}') + c(\mathcal{S}', \mathcal{S}) + h(\mathcal{S}) \\ &= f(\mathcal{S}') + c(\mathcal{S}', \mathcal{S}) + h(\mathcal{S}) - h(\mathcal{S}') \\ &= f(\mathcal{S}') + c(\mathcal{S}', \mathcal{S}) - \delta_{r_{\max}}(\mathcal{S}', \mathcal{S}). \end{aligned} \tag{18}$$

In addition, only S_t in the image function (14) affects the f-value. Therefore, the calculation of f can be implemented by slightly revising S_t as Algorithm 4.

In Algorithm 4, the f-value of a state is represented by a set of Boolean variables $f^0, f^1, \ldots, f^{k_f}$ and the symbolic function α of the addition of 1 is implemented as follows:

$$\alpha^j(f) = \begin{cases} \bar{f}^j, & \text{if } j = 0 \\ f^j \bigoplus C_{j-1}, & \text{if } j \geq 1 \end{cases} \tag{19}$$

where

$$C_j = \begin{cases} f^j, & \text{if } j = 0 \\ f^j \cdot C_{j-1}, & \text{if } j \geq 1. \end{cases} \tag{20}$$

Algorithm 4 shows that in the gradually decreasing remaining time of tokens in activity pre-places of t as time goes by, according to (18), if such pre-places contain a loyal holder of $r_{\max}$, then the f-value remains unchanged since the time $c(\mathcal{S}', \mathcal{S})$ spent on the state transfer equals the time $\delta_{r_{\max}}(\mathcal{S}', \mathcal{S})$ used by the loyal holder in the transfer; otherwise, the f-value increases gradually as time goes by since $\delta_{r_{\max}}(\mathcal{S}', \mathcal{S}) = 0$.

Algorithm 4 Decreases remaining time of tokens in activity pre-places of t and updates f-values.

Input: A transition t and a BDD X representing a set of state satisfying (3).

Output: A BDD representing the states with the adjusted remaining time of the tokens in the pre-places of t and the adjusted f-values.

1: **function** $\text{S}_t(X)$
2: $Z_1 := X \cdot \widetilde{\mathcal{F}}_t$;
3: $Y := X \setminus Z_1$;
4: **while** $Z_1 \neq$ bddfalse **do**
5: **for all** i such that $p_i \in P_A$ and $D(p_i) > 0$ **do**
6: **for** $u := 1$ to λ **do**
7: $Z_2 := Z_1 \cdot (R_{i,u} \trianglerighteq 1)$;
8: **if** $Z_2 \neq$ bddfalse **then**
9: $Z_1 := Z_1 \setminus Z_2$;
10: $Z_2 := \exists_{r_{i,u}} \prod_{j:=0}^{k_{r_{i,u}}} \left[s^j_{i,u} \equiv \beta^j (r_{i,u}) \right] \cdot Z_2$;
11: $Z_2[s \leftrightarrow r]$;
12: **if** $p_i \in H_l(r_{\max})$ **then** /*Subtract 1.*/
13: $Z_2 := \exists_f \prod_{j:=0}^{k_f} \left[e^j \equiv \beta^j(f) \right] \cdot Z_2$;
14: $Z_2[e \leftrightarrow f]$;
15: **end if**
16: $Z_1 := Z_1 + Z_2$;
17: **end if**
18: **end for**
19: **end for**
20: $Z_1 := \exists_f \prod_{j:=0}^{k_f} \left[e^j \equiv \alpha^j(f) \right] \cdot Z_1$; /*Add 1 to f.*/
21: $Z_1[e \leftrightarrow f]$;
22: $Y := Y + \left[Z_1 \setminus (Z_1 \cdot \widetilde{\mathcal{F}}_t) \right]$;
23: $Z_1 := Z_1 \cdot \widetilde{\mathcal{F}}_t$;
24: **end while**
25: **return** Y;
26: **end function**

IV. Experiments

In this section, some FMS examples are tested with the proposed method. For the application of BDD, C++ programs have been developed with the BDD tool Buddy-2.2 package [34] on a Linux operating system with 8GB memory and Intel i5 Core 2.2GHz CPU. The Boolean variables for states are interleaved, which means that assuming $\mathcal{S} = (A, B)$ in which A needs two Boolean variables a^1 and a^2 and B needs four variables b^1, b^2, b^3 and b^4, then the order of Boolean variables will be a^1, b^1, a^2, b^2, b^3, b^4. The implementation codes of our method and experimental files of all examples are available in [35] for reader's examination and reference.

First, we consider an FMS example from [36]–[40], which has six resources $R_1 - R_6$ that may be machines, robots, etc., two loading buffers $\mathcal{I}_1 - \mathcal{I}_2$, and two unloading buffers $\mathcal{O}_1 - \mathcal{O}_2$. Two types of parts, $\mathcal{J}_1 - \mathcal{J}_2$, are processed in the system. The production sequences are as below.

$$\mathcal{J}_1 : \mathcal{I}_1 \to \mathcal{R}_3 \to \mathcal{R}_2(\text{or } \mathcal{R}_5) \to \mathcal{R}_3 \to \mathcal{R}_6 \to \mathcal{R}_4 \to \mathcal{O}_1$$
$$\mathcal{J}_2 : \mathcal{I}_2 \to \mathcal{R}_4 \to \mathcal{R}_1 \to \mathcal{R}_3 \to \mathcal{R}_2 \to \mathcal{R}_3 \to \mathcal{O}_2$$

Let $O_{i,j,k}$ be the jth operation of the ith job being processed with the kth resource. The processing time of each operation is given in Table I. Its P-timed PN model is shown in Fig. 1 where each idle place of the original net has been split into a start place and an end one to discriminate the goal marking from the others. The net has 14 transitions and 21 places in which $P_S = \{p_1, p_8\}$, $P_E = \{p_{20}, p_{21}\}$, $P_R = \{p_{14} - p_{19}\}$, and the rest belong to activity ones P_A. Eight sets of lot size are tested by the explicit-state A* [11] with $h = 0$ and our symbolic method with $h = 0$ or $h = h_{\text{BDD}}$. The scheduling results of makespan, the number of expanded states ($N_{\mathcal{S}}$), the number of BDD nodes for such expanded states (N_{BDD}), and the computational time ($\mathcal{T}$) are shown in Table II. The following phenomena are observed. (1) The makespans of the results obtained by the A* search with $h = h_{\text{BDD}}$ are the same as those of the optimal results obtained by the ones with $h = 0$. (2) The numbers of expanded states of the symbolic A* search with $h = 0$ are slightly larger than those of the explicit-state one with the same heuristic. This phenomenon is to be

TABLE I
PROCESSING TIME OF EXAMPLE 1.

Operation	Processing Time	Operation	Processing Time
$O_{1,1,3}$	3	$O_{2,1,4}$	2
$O_{1,2,2}$	2	$O_{2,2,1}$	4
$O_{1,2,5}$	4	$O_{2,3,3}$	4
$O_{1,3,3}$	4	$O_{2,4,2}$	3
$O_{1,4,6}$	3	$O_{2,5,3}$	5
$O_{1,5,4}$	5		

expected by considering that, for the sake of computational efficiency, the symbolic algorithm allows inserting identical markings with different f-values into the BDD for expansion, while the explicit-state one will spend time to identify and pick out them. (3) The symbolic methods, especially the one with the heuristic h_{BDD}, are more efficient and faster to schedule FMSs than the explicit-state one, especially for larger cases.

The second FMS consists of four loading buffers $\mathcal{I}_1 - \mathcal{I}_4$, four unloading buffers $\mathcal{O}_1 - \mathcal{O}_4$, and three resources $R_1 - R_3$. Four types of parts, $\mathcal{J}_1 - \mathcal{J}_4$, are processed in the system. The production sequences are as below.

$$\mathcal{J}_1 : \mathcal{I}_1 \to \mathcal{R}_1 \to \mathcal{R}_2 \to \mathcal{R}_3 \to \mathcal{O}_1$$
$$\mathcal{J}_2 : \mathcal{I}_2 \to \mathcal{R}_2 \to \mathcal{R}_3 \to \mathcal{R}_1 \to \mathcal{O}_2$$
$$\mathcal{J}_3 : \mathcal{I}_3 \to \mathcal{R}_3 \to \mathcal{R}_1 \to \mathcal{R}_2 \to \mathcal{O}_3$$
$$\mathcal{J}_4 : \mathcal{I}_4 \to \mathcal{R}_3 \to \mathcal{R}_2 \to \mathcal{R}_1 \to \mathcal{O}_4$$

Note that, there exists an intermediate buffer between any two consecutive operations to hold parts which are ready for their next operation. The processing time of each operation is given in Table III. The P-timed PN model of the system is shown in Fig. 2. It has 24 transitions and 31 places where $P_S = \{p_1, p_8, p_{15}, p_{22}\}$, $P_E = \{p_7, p_{14}, p_{21}, p_{28}\}$, $P_R = \{p_{29}, p_{30}, p_{31}\}$, and the rest belong to activity places P_A. The activity places without time information represent intermediate buffers, and the rest with a nonnegative integer (processing time) denote the operations to be performed with some specific resources. For such a net, five sets of lot size are tested by using the explicit-state A* search and our symbolic method. The scheduling results are given in Table IV. It can be seen that all the methods can obtain the results with the same makespans, which are guaranteed to be the minimal by the optimality of A* search. In addition, our symbolic methods, especially the one with the heuristic h_{BDD}, dramatically outperforms the explicit-state one in the computational cost and can handle some larger cases that cannot be solved by the explicit-state one.

The third example is a more complex one adapted from [17]. There are twelve resources $R_1 - R_{12}$, five loading buffers $\mathcal{I}_1 - \mathcal{I}_5$, and five unloading buffers $\mathcal{O}_1 - \mathcal{O}_5$. Five types of parts, $\mathcal{J}_1 - \mathcal{J}_5$, are processed in the system. The production sequences are as below.

$$\mathcal{J}_1 : \mathcal{I}_1 \to \mathcal{R}_1 \to \mathcal{R}_6 \to \mathcal{R}_2 \to \mathcal{R}_7 \to \mathcal{R}_3 \to \mathcal{O}_1$$
$$\text{or } \mathcal{I}_1 \to \mathcal{R}_1 \to \mathcal{R}_8 \to \mathcal{R}_3 \to \mathcal{R}_9 \to \mathcal{R}_3 \to \mathcal{O}_1$$
$$\mathcal{J}_2 : \mathcal{I}_2 \to \mathcal{R}_2 \to \mathcal{R}_{11} \to \mathcal{R}_2 \to \mathcal{R}_7 \to \mathcal{R}_5 \to \mathcal{O}_2$$
$$\mathcal{J}_3 : \mathcal{I}_3 \to \mathcal{R}_3 \to \mathcal{R}_9 \to \mathcal{R}_3 \to \mathcal{R}_8 \to \mathcal{R}_1 \to \mathcal{O}_3$$

TABLE III
PROCESSING TIME OF EXAMPLE 2.

Operation	Processing Time	Operation	Processing Time
$O_{1,1,1}$	5	$O_{3,1,3}$	5
$O_{1,2,2}$	4	$O_{3,2,1}$	2
$O_{1,3,3}$	4	$O_{3,3,2}$	5
$O_{2,1,2}$	2	$O_{4,1,3}$	2
$O_{2,2,3}$	5	$O_{4,2,2}$	4
$O_{2,3,1}$	2	$O_{4,3,1}$	2

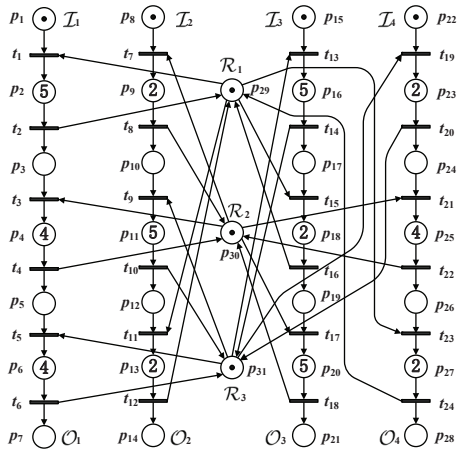

Fig. 2. P-timed PN model of Example 2.

$$\mathcal{J}_4 : \mathcal{I}_4 \to \mathcal{R}_4 \to \mathcal{R}_{11} \to \mathcal{R}_2 \to \mathcal{R}_{10} \to \mathcal{R}_2 \to \mathcal{O}_4$$
$$\text{or } \mathcal{I}_4 \to \mathcal{R}_4 \to \mathcal{R}_9 \to \mathcal{R}_3 \to \mathcal{R}_7 \to \mathcal{R}_2 \to \mathcal{O}_4$$
$$\mathcal{J}_5 : \mathcal{I}_5 \to \mathcal{R}_5 \to \mathcal{R}_{12} \to \mathcal{R}_1 \to \mathcal{R}_6 \to \mathcal{R}_1 \to \mathcal{O}_5$$

The processing time of each operation is given in Table V. Its PN model is shown in Fig. 3 where idle places in the original net have been split into start places and end ones. The net has 38 transitions and 53 places in which $P_S = \{p_1, p_{10}, p_{16}, p_{22}, p_{31}\}$, $P_E = \{p_{49} - p_{53}\}$, $P_R = \{p_{37} - p_{48}\}$, and the rest places are activity ones P_A. Our symbolic A* search with h_{BDD} is used to solve it. Table VI shows the obtained scheduling results in the form of a sequence of transition firings, which is guaranteed to be the optimal by our method. The makespan of the optimal schedule is 36. In our symbolic method, the number of expanded states is 9.96×10^5, the number of BDD nodes for such states is 1.46×10^7, and the computational time is about 41 minutes. While the explicit-state A* search and the symbolic one with $h = 0$ cannot solve it in an acceptable period of time.

V. CONCLUSIONS

This work proposes a method to schedule FMSs based on symbolic techniques and timed PNs. Such a method is never seen in the literature to the best knowledge of the authors. It takes advantages of succinct data structures to represent sets of states and their actions through some binary characteristic functions, and it is thus more efficient in system schedule when compared with the explicit-state ones which operate on individual states of timed PN models.

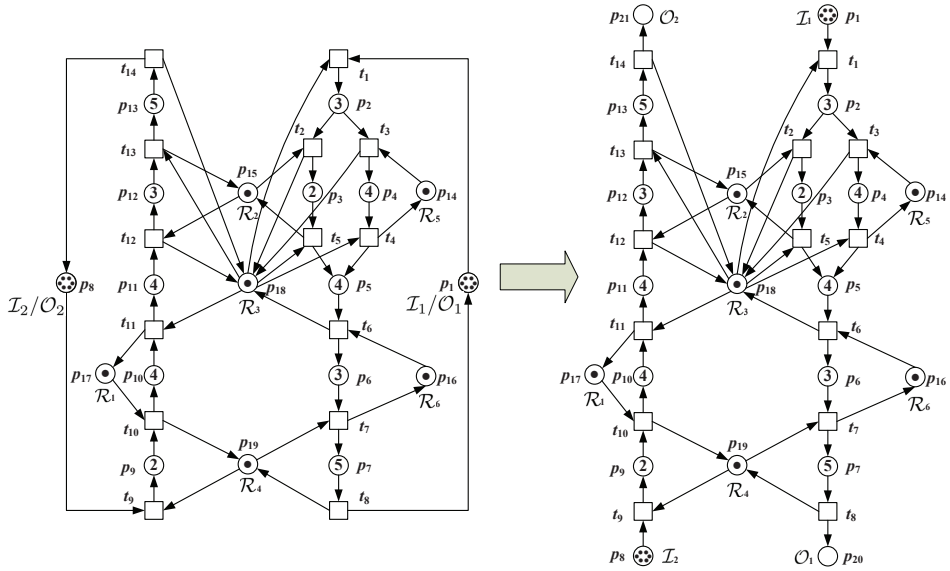

Fig. 1. P-timed PN model of Example 1.

TABLE II
SCHEDULING RESULTS FOR EXAMPLE 1.

p_1	p_8	Makespan	Explicit A$^*(h=0)$		Symbolic A$^*(h=0)$			Symbolic A$^*(h=h_{\text{BDD}})$		
			$N_{\mathcal{S}}$	$\mathcal{T}$	$N_{\mathcal{S}}$	N_{BDD}	$\mathcal{T}$	$N_{\mathcal{S}}$	N_{BDD}	$\mathcal{T}$
1	1	21	5.60×10^{1}	0.01s	2.04×10^{2}	9.23×10^{3}	0.55s	9.60×10^{1}	5.64×10^{3}	0.71s
2	2	35	1.05×10^{2}	0.16s	1.41×10^{4}	1.63×10^{5}	4.62s	1.10×10^{3}	2.77×10^{4}	1.08s
3	3	51	6.17×10^{3}	4.65s	1.45×10^{5}	6.73×10^{5}	40.50s	6.18×10^{3}	1.05×10^{5}	3.11s
4	4	67	1.85×10^{4}	50.18s	5.68×10^{5}	1.37×10^{6}	137.67s	2.18×10^{4}	2.50×10^{5}	9.37s
5	5	83	3.47×10^{4}	194.99s	1.43×10^{6}	2.23×10^{6}	292.97s	4.95×10^{4}	4.05×10^{5}	17.96s
6	6	99	5.32×10^{4}	463.03s	2.86×10^{6}	3.13×10^{6}	460.59s	8.90×10^{4}	5.32×10^{5}	27.29s
7	7	115	7.50×10^{4}	943.46s	4.99×10^{6}	3.76×10^{6}	686.53s	1.40×10^{5}	6.32×10^{5}	35.49s
8	8	131	1.01×10^{5}	1718.39s	7.95×10^{6}	4.23×10^{6}	911.54s	2.03×10^{5}	6.99×10^{5}	46.05s

TABLE IV
SCHEDULING RESULTS FOR EXAMPLE 2.

p_1	p_8	p_{15}	p_{22}	Makespan	Explicit A$^*(h=0)$		Symbolic A$^*(h=0)$			Symbolic A$^*(h=h_{\text{BDD}})$		
					$N_{\mathcal{S}}$	$\mathcal{T}$	$N_{\mathcal{S}}$	N_{BDD}	$\mathcal{T}$	$N_{\mathcal{S}}$	N_{BDD}	$\mathcal{T}$
1	1	1	1	16	2696	1.09s	1.75×10^{4}	2.36×10^{5}	4.71s	2.68×10^{3}	5.29×10^{4}	1.65s
2	1	1	1	20	13934	35.54s	1.51×10^{5}	1.15×10^{6}	28.13s	8.18×10^{3}	1.14×10^{5}	2.99s
2	2	1	1	25	71673	1041.89s	1.05×10^{6}	4.35×10^{6}	146.86s	6.18×10^{4}	4.47×10^{5}	17.97s
2	2	2	1	30	-	-	5.30×10^{6}	1.28×10^{7}	798.84s	3.78×10^{5}	1.37×10^{6}	91.17s
2	2	2	2	32	-	-	2.13×10^{7}	2.88×10^{7}	3442.18s	1.40×10^{6}	3.12×10^{6}	379.28s

In the future, we will focus on how to apply our method to other discrete event systems. Besides, the ordering of Boolean variables in our method also interests us since it can reduce the size of BDDs [41], thus speeding up BDD operations.

REFERENCES

[1] G. Tuncel and G. M. Bayhan, "Applications of Petri nets in production scheduling: a review," *Int. J. Adv. Manuf. Tech.*, vol. 34, no. 7-8, pp. 762–773, Oct. 2007.

[2] M. Zhou and K. Venkatesh, *Modeling, simulation and control of flexible manufacturing systems: A Petri net approach*. Singapore: World Scientific, 1998.

[3] D. Y. 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.

[4] J. Pearl, *Heuristics: intelligent search strategies for computer problem solving*. MA, USA: Addison-Wesley, 1984.

[5] H. H. 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.

[6] G. Mejia, "An intelligent agent-based architecture for flexible manufacturing systems having error recovery capability," Thesis, Lehigh University, 2002.

[7] A. Reyes, H. Yu, G. Kelleher, and S. Lloyd, "Integrating Petri nets and hybrid heuristic search for the scheduling of FMS," *Computers in Industry*, vol. 47, no. 1, pp. 123–138, Jan. 2002.

[8] J. Lee and J. S. Lee, "Heuristic search for scheduling flexible manufacturing systems using lower bound reachability matrix," *Comput. Ind. Eng.*, vol. 59, no. 4, pp. 799–806, Jan. 2010.

[9] J. Luo, K. Xing, M. Zhou, X. Li, and X. Wang, "Deadlock-free scheduling of automated manufacturing systems using Petri nets and

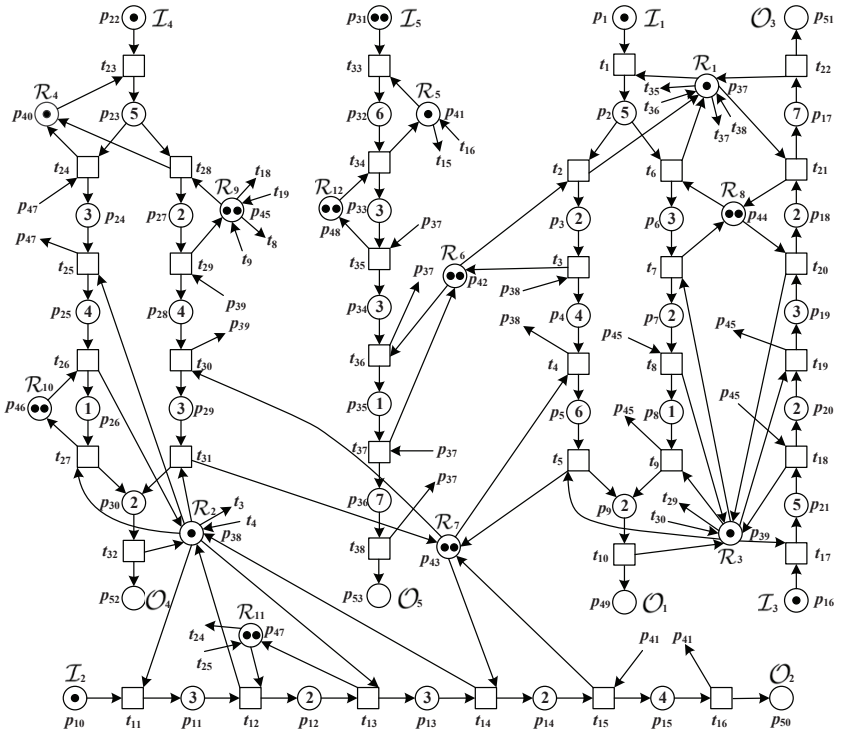

Fig. 3. P-timed PN model of a more complex system in Example 3.

TABLE V
PROCESSING TIME OF EXAMPLE 3.

Operation	Processing Time	Operation	Processing Time
$O_{1,1,1}$	5	$O_{3,4,8}$	2
$O_{1,2,6}$	2	$O_{3,5,1}$	7
$O_{1,2,8}$	3	$O_{4,1,4}$	5
$O_{1,3,2}$	4	$O_{4,2,11}$	3
$O_{1,3,3}$	2	$O_{4,2,9}$	2
$O_{1,4,7}$	6	$O_{4,3,2}$	4
$O_{1,4,9}$	1	$O_{4,3,3}$	4
$O_{1,5,3}$	2	$O_{4,4,10}$	1
$O_{2,1,2}$	3	$O_{4,4,7}$	3
$O_{2,2,11}$	2	$O_{4,5,2}$	2
$O_{2,3,2}$	3	$O_{5,1,5}$	6
$O_{2,4,7}$	2	$O_{5,2,12}$	3
$O_{2,5,5}$	4	$O_{5,3,1}$	3
$O_{3,1,3}$	5	$O_{5,4,6}$	1
$O_{3,2,9}$	2	$O_{5,5,1}$	7
$O_{3,3,3}$	3		

TABLE VI
THE OPTIMAL SCHEDULE PATH OF EXAMPLE 3.

Trans.	Firing Time	Trans.	Firing Time	Trans.	Firing Time
t_{33}	0	t_{20}	10	t_3	22
t_{23}	0	t_{36}	12	t_{36}	22
t_{17}	0	t_{34}	12	t_{37}	22
t_1	0	t_{26}	12	t_4	26
t_{24}	5	t_{21}	12	t_{38}	29
t_{18}	5	t_{11}	12	t_{37}	29
t_2	5	t_{12}	15	t_{27}	29
t_{34}	6	t_{13}	17	t_5	32
t_{33}	6	t_{22}	19	t_{38}	36
t_{19}	7	t_{35}	19	t_{32}	36
t_{25}	8	t_{14}	20	t_{16}	36
t_{35}	9	t_{15}	22	t_{10}	36

hybrid heuristic search," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 45, no. 3, pp. 530–541, 2015.

[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, 2015.

[11] B. Huang, Y. Sun, and Y. Sun, "Scheduling of flexible manufacturing systems based on Petri nets and hybrid heuristic search," *Int. J. Prod. Res.*, vol. 46, no. 16, pp. 4553–4565, Aug. 2008.

[12] B. Huang, Y. Sun, Y. Sun, and C. Zhao, "A hybrid heuristic search algorithm for scheduling FMS based on Petri net model," *Int. J. Adv. Manuf. Tech.*, vol. 48, no. 9-12, pp. 925–933, Jun. 2010.

[13] B. Huang, X. Shi, and N. Xu, "Scheduling FMS with alternative routings using Petri nets and near admissible heuristic search," *Int. J. Adv. Manuf. Tech.*, vol. 63, no. 9-12, pp. 1131–1136, Dec. 2012.

[14] 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.

[15] B. Huang, R. Jiang, and G. Zhang, "Comments on heuristic search for scheduling flexible manufacturing systems using lower bound reachability matrix," *Comput. Ind. Eng.*, vol. 67, pp. 235–236, Jan. 2014.

[16] E. Pastor, J. Cortadella, and O. Roig, "Symbolic analysis of bounded Petri nets," *IEEE Transactions on Computers*, vol. 50, no. 5, pp. 432–448, 2001.

[17] Y. Chen, Z. Li, M. Khalgui, and O. Mosbahi, "Design of a maximally permissive liveness-enforcing Petri net supervisor for flexible manufacturing systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 2, pp. 374–393, Apr. 2011.

[18] Y. Chen and G. Liu, "Computation of minimal siphons in Petri nets by using binary decision diagrams," *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 1, p. 3, 2013.

[19] F. M. Brown, *Boolean reasoning: the logic of Boolean equations*. Springer Science & Business Media, 2012.

[20] H. R. Andersen, "An introduction to binary decision diagrams," *Lecture notes*, 1997.

[21] Z. Li and M. Zhou, *Deadlock resolution in automated manufacturing systems: a novel Petri net approach*. London, U.K.: Springer, 2009.

[22] F. Hsieh and S. Chang, "Dispatching-driven deadlock avoidance controller synthesis for flexible manufacturing systems," *IEEE Trans. Robot. Autom.*, vol. 10, no. 2, pp. 196–209, Apr. 1994.

[23] J. Ezpeleta, J. M. Colom, and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. Robot. Autom.*, vol. 11, no. 2, pp. 173–184, Apr. 1995.

[24] F. Tricas, F. Garcia-Valles, J. Colom, and J. Ezpeleta, "A structural approach to the problem of deadlock prevention in processes with resources," in *Proc. WODES'98*, Italy, Aug. 1998, pp. 273–278.

[25] A. Wang, Z. Li, J. Jia, and M. Zhou, "An effective algorithm to find elementary siphons in a class of Petri nets," *IEEE Trans. Syst., Man, Cybern. A*, vol. 39, no. 4, pp. 912–923, Jul. 2009.

[26] A. Wang, Z. Li, and J. Jia, "Efficient computation of strict minimal siphons for a class of Petri nets models of automated manufacturing systems," *Trans. Inst. Meas. Control*, vol. 33, no. 1, pp. 182–201, Feb. 2011.

[27] Y. Hou, Z. Li, A. M. Al-Ahmari, A.-A. M. El-Tamimi, and E. A. Nasr, "Extended elementary siphons and their application to liveness-enforcement of generalized Petri nets," *Asian J. Control*, vol. 16, no. 6, pp. 1789–1810, Nov. 2014.

[28] J. Park and S. A. Reveliotis, "Algebraic synthesis of efficient deadlock avoidance policies for sequential resource allocation systems," *IEEE Trans. Robot. Autom.*, vol. 16, no. 2, pp. 190–195, Apr. 2000.

[29] J. Ezpeleta, F. Tricas, F. Garcia-Valles, and J. M. Colom, "A banker's solution for deadlock avoidance in FMS with flexible routing and multiresource states," *IEEE Trans. Robot. Autom.*, vol. 18, no. 4, pp. 621–625, Aug. 2002.

[30] F. Tricas, F. Garcia-Valles, J. Colom, and J. Ezpeleta, "An iterative method for deadlock prevention in FMS," in *Discrete Event Systems*. Springer, 2000, pp. 139–148.

[31] J. Park and S. A. Reveliotis, "Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings," *IEEE Trans. Autom. Control*, vol. 46, no. 10, pp. 1572–1583, Oct. 2001.

[32] Y. Huang, M. Jeng, X. Xie, and D. Chung, "Siphon-based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A*, vol. 36, no. 6, pp. 1248–1256, Nov. 2006.

[33] J. O. Moody and P. J. Antsaklis, "Petri net supervisors for DES with uncontrollable and unobservable transitions," *IEEE Trans. Autom. Control*, vol. 45, no. 3, pp. 462–476, Mar. 2000.

[34] J. Lind-Nielsen, "BuDDy: Binary decision diagram package release 2.2," IT-Univ. Copenhagen (ITU), Copenhagen, Denmark, 2002.

[35] "Source codes and experimental files," Mar. 2018. [Online]. Available: http://web.njit.edu/~zhou/BDD.rar

[36] M. Uzam, "An optimal deadlock prevention policy for flexible manufacturing systems using Petri net models with resources and the theory of regions," *Int. J. Adv. Manuf. Tech.*, vol. 19, no. 3, pp. 192–208, Feb. 2002.

[37] Z. Li, M. Zhou, and M. Jeng, "A maximally permissive deadlock prevention policy for FMS based on Petri net siphon control and the theory of regions," *IEEE Trans. Autom. Sci. Eng.*, vol. 5, no. 1, pp. 182–188, Mar. 2008.

[38] L. Piroddi, R. Cordone, and I. Fumagalli, "Selective siphon control for deadlock prevention in Petri nets," *IEEE Trans. Syst., Man, Cybern. A*, vol. 38, no. 6, pp. 1337–1348, Nov. 2008.

[39] B. Huang, M. Zhou, G. Zhang, A. C. Ammari, A. Alabdulwahab, and A. G. Fayoumi, "Lexicographic multiobjective interger programming for optimal and structurally minimal Petri net supervisors of automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 45, no. 11, pp. 1459–1470, Nov. 2015.

[40] B. Huang, M. Zhou, P. Zhang, and J. Yang, "Speedup techniques for multiobjective integer programs in designing optimal and structurally simple supervisors of AMS," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 1, pp. 77–88, Jan. 2018.

[41] S. Friedman and K. Supowit, "Finding the optimal variable ordering for binary decision diagrams," *IEEE Trans. Comput.*, vol. 39, no. 5, pp. 710–713, 1990.

APPENDIX
ALGORITHMS FOR CONSTRUCTING A PATH

Algorithm 5 Constructs a solution.

Input: $Closed$, $\mathcal{S}_0$, $\mathcal{S}_G$.
Output: A firing sequence from $\mathcal{S}_0$ to $\mathcal{S}_G$.

function CONSTRUCT($Closed$, $\mathcal{S}_0$, $\mathcal{S}_G$)
  $path := \emptyset$; $a := 0$; $\mathcal{S}' := \mathcal{S}_G$; $\overleftarrow{g} := 0$;
  while $\mathcal{S}' \neq \mathcal{S}_0$ **do**
    $\mathcal{S} := \mathcal{S}'$;
    for all $t \in \overleftarrow{E_t}(\mathcal{S})$ **do**
      if $(\mathcal{S}', c) :=$ BackT$(Closed, \mathcal{S}_0, \mathcal{S}, t) \neq \emptyset$ **then**
        $path[a] := (\mathcal{S}_G.f - \overleftarrow{g}, t)$;
        $\overleftarrow{g} = \overleftarrow{g} + c$;
        $a := a + 1$;
        break;
      end if
    end for
  end while
  return $path$;
end function

Algorithm 6 Looks for an immediate predecessor of $\mathcal{S}$ in *Closed* while $\lambda = 2$.

Input: $Closed$, $\mathcal{S}_0$, $\mathcal{S}$, t which is reversely enabled at $\mathcal{S}$.
Output: $(\mathcal{S}', c)$ with $\mathcal{S}' \in$ *Closed*, $\mathcal{S}'[t\rangle\mathcal{S}$, $\mathcal{S}.g = \mathcal{S}'.g + c$ and $c = c(\mathcal{S}', \mathcal{S})$; otherwise, $\emptyset$.

function BACKT($Closed, \mathcal{S}_0, \mathcal{S}, t$)
  /* Moves tokens for the reversely firing of t. */
  for all $p \in t^\bullet \cup {}^\bullet t$ **do**
    $\mathcal{S}.M(p) := \mathcal{S}.M(p) - I(p,t)$;
  end for
  if $\mathcal{S}.M = \mathcal{S}_0.M$ **then**
    return $(\mathcal{S}_0, 0)$;
  end if
  /* Handles the remaining time of tokens in the activity post-place of t. */
  if $\{q\} := \{p | p \in t^\bullet$ and $D(p) > \emptyset\} \neq \emptyset$ **then**
    if $\mathcal{S}.R_1(q) = D(p)$ **then**
      $\mathcal{S}.R_1(q) := 0$;
    else
      $\mathcal{S}.R_2(q) := 0$;
    end if
  end if
  $\mathcal{S}' := \mathcal{S}$;
  for all $t' \in \overleftarrow{E_t}(\mathcal{S}'.M)$ **do**
    $C := \prod_{p_i \in P} [M(p_i) \bowtie \mathcal{S}'.M(p_i)]$;
    $\{p'\} := t'^\bullet \cap P_A$;
    $\{p''\} := {}^\bullet t \cap (P \setminus P_R)$;
    $d := D(p') - \max\{\mathcal{S}'.R_1(p'), \mathcal{S}'.R_2(p')\}$;
    if $d > D(p'') - \max\{\mathcal{S}'.R_1(p''), \mathcal{S}'.R_2(p'')\}$ **then**
      continue;
    end if
    if $\max\{\mathcal{S}'.R_i(p')\} > 0$ or $D(p') = 0$ or $p' = p''$ **then** /* If the time needed by $\mathcal{S}'[t\rangle\mathcal{S}$ equals d. */
      $C := C \cdot$ Constraints$(\mathcal{S}', p', p'', d)$;
    else
      $C1 :=$ bddfalse;
      for all k such that $0 \leq k \leq D(p'') - \max\{\mathcal{S}'.R_1(p''), \mathcal{S}'.R_2(p'')\} - d$ **do**
        $C1 := C1 +$ Constraints$(\mathcal{S}', p', p'', d + k)$;
      end for
      $C := C \cdot C1$;
    end if
    /* Looks for a state satisfying C in $Closed$. */
    $found := Closed \cdot C$;
    if $found \neq$ bddfalse **then**
      $\mathcal{S}' :=$ bdd2var(bdd_satone($found$)); /* Finds a satisfying variable assignment. */
      if $\mathcal{S}'.M(p'') = 1$ **then**
        return $(\mathcal{S}', \max\{\mathcal{S}'.R_1(p''), \mathcal{S}'.R_2(p'')\})$;
      else /* $\mathcal{S}'.M(p'') = 2$. */
        return $(\mathcal{S}', \min\{\mathcal{S}'.R_1(p''), \mathcal{S}'.R_2(p'')\})$;
      end if
    end if
  end for
  return $\emptyset$; /* If no such a state in *Closed*. */
end function

Algorithm 7 Formulates constraints of f and R_i for $\mathcal{S}'$.

Input: $\mathcal{S}'$, p' the activity post-place of a transition t', p'' the non-resource pre-place of a transition t, and d the time required for p' to reversely enable t' at $\mathcal{S}'$ such that $\exists \mathcal{S}, \mathcal{S}''$, $\mathcal{S}'[t\rangle\mathcal{S}$ and $\mathcal{S}''[t'\rangle\mathcal{S}'$.
Output: Constraints C.

function CONSTRAINTS($\mathcal{S}'$, p', p'', d)
  /* Constraints of f. */
  $\{l\} := \{p | \mathcal{S}'.M(p) > 0$ and $D(p) > 0$ and $p \in H_l(r_{\max})\}$;
  if $\{l\} = \emptyset$ **then**
    $C := C \cdot [f \bowtie (\mathcal{S}'.f - d)]$;
  else if $l = p''$ **then**
    $C := C \cdot (f \bowtie \mathcal{S}'.f)$;
  else
    if $\mathcal{S}'.R_1(l) > 0$ or $\mathcal{S}'.R_2(l) > 0$ **then**
      $C := C \cdot (f \bowtie \mathcal{S}'.f)$;
    else /*$\mathcal{S}'.R_1(l) = \mathcal{S}'.R_2(l) = 0$*/
      $C_1 :=$ bddfalse;
      for all a such that $0 \leq a \leq d$ **do**
        if $a = 0$ **then**
          $C_1 := C_1 + [f \bowtie (\mathcal{S}'.f - d)] \cdot [R_1(l) \bowtie 0] \cdot [R_2(l) \bowtie 0]$;
        else
          $C_1 := C_1 + [f \bowtie (\mathcal{S}'.f - d + a)] \cdot [R_1(l) \bowtie a + R_2(l) \bowtie a]$;
        end if
      end for
      $C := C \cdot C_1$;
    end if
  end if
  /* Constraints of R in p' and p''. */
  $C := C \cdot [R_1(p') \bowtie D(p') + R_2(p') \bowtie D(p')]$;
  if $D(p'') > 0$ **then**
    if $p'' = p'$ **then**
      $C := C \cdot [R_1(p'') \bowtie (\mathcal{S}'.R_1(p'') + d)] \cdot [R_2(p'') \bowtie (\mathcal{S}'.R_2(p'') + d)]$;
    else
      if $\mathcal{S}'.M(p'') = 1$ **then**
        if $d > 0$ **then**
          $C := C \cdot [R_1(p'') \bowtie d + R_2(p'') \bowtie d]$;
        else /* $d = 0$.*/
          $C := C \cdot [R_1(p'') \bowtie 0] \cdot [R_2(p'') \bowtie 0]$;
        end if
      else
        $C := C \cdot [R_1(p'') \bowtie (\mathcal{S}'.R_1(p'') + d)] \cdot [R_2(p'') \bowtie (\mathcal{S}'.R_2(p'') + d)]$;
      end if
    end if
  end if

/* Constraints of R for the rest activity places. */
for all $q' \in \{p | \mathcal{S}'.M(p) > 0$ and $D(p) > 0$ and $p \neq p'$ and $p \neq p''\}$ **do**
  if $\mathcal{S}'.M(q') = 1$ **then**
    if $\mathcal{S}'.R_1(q') > 0$ **then**
      $C := C \cdot [R_1(q') \bowtie (\mathcal{S}'.R_1(q') + d) \cdot R_2(q') \bowtie 0]$;
    else if $\mathcal{S}'.R_2(q') > 0$ **then**
      $C := C \cdot [R_2(q') \bowtie (\mathcal{S}'.R_2(q') + d) \cdot R_1(q') \bowtie 0]$;
    else
      $C := C \cdot [R_1(q') \trianglelefteq d \cdot R_2(q') \bowtie 0 + R_2(q') \trianglelefteq d \cdot R_1(q') \bowtie 0]$;
    end if
  else /* $\mathcal{S}'.M(q') = 2$. */
    if $\mathcal{S}'.R_1(q') > 0$ **then**
      $C := C \cdot [R_1(q') \bowtie (\mathcal{S}'.R_1(q') + d)]$;
    else
      $C := C \cdot [R_1(q') \trianglelefteq d]$;
    end if
    if $\mathcal{S}'.R_2(q') > 0$ **then**
      $C := C \cdot [R_2(q') \bowtie (\mathcal{S}'.R_2(q') + d)]$;
    else
      $C := C \cdot [R_2(q') \trianglelefteq d]$;
    end if
  end if
end for
return C;
end function