IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 19, 2019, accepted May 27, 2019, date of publication June 4, 2019, date of current version July 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2920677

A Siphon-Based Deadlock Prevention Strategy for S^3PR

XIN GUO[1], SHOUGUANG WANG[1], (Senior Member, IEEE), DAN YOU[1,2], (Student Member, IEEE), ZHIFU LI[1], AND XIAONING JIANG[1]

[1]School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China
[2]Department of Electrical and Electronic Engineering, University of Cagliari, Italy

Corresponding author: Shouguang Wang (wsg5000@hotmail.com)

This work was supported in part by the Zhejiang Provincial Key R&D Program of China under Grant 2018C01084, in part by the Zhejiang Natural Science Foundation under Grant LY15F030003, in part by the National Natural Science Foundation of China under Grant 61472361, and in part by the Zhejiang Science and Technology Project under Grant 2015C31064.

ABSTRACT Iterative deadlock prevention strategies based on siphons have drawn increasing attention. For iterative strategies, selecting which siphon to control at each iteration has an influence on the final supervisor in structural complexity, computational complexity, and behavioral permissiveness. In this paper, we define two kinds of emptiable siphons and provide two modified mixed-integer programming (MIP) formulations to compute such siphons. On the basis of them, a three-stage iterative deadlock prevention policy that specifies the siphon control order is proposed. The experimental results show that a supervisor with a simpler structure, higher behavioral permissiveness, and lower computational complexity can be obtained by the proposed strategy since neither the exhaustive siphon enumeration nor the reachability analysis is required.

INDEX TERMS Deadlock prevention, discrete event systems, mixed integer programming, Petri nets.

I. INTRODUCTION

The occurrence of deadlocks in Flexible Manufacturing Systems (FMSs) [1]–[5] implies a local or the whole stoppage of normal system operation, which may reduce the system productivity, increase unnecessary costs, and even result in disastrous consequences. Hence, growing attention has been paid on deadlock problems in academic and industrial circles. Petri nets (PNs) are considered to be one of the suitable mathematical models for handling deadlocks in FMSs due to their powerful capabilities of intuitively and compactly characterizing discrete processes. There are a wide variety of methods in the literature dealing with deadlocks based on PNs [6]–[37]. These methods mainly fall into three categories: deadlock prevention [13]–[33], deadlock avoidance [34]–[36], and deadlock detection and recovery [37]. Among the above categories, deadlock prevention strategies have the advantage that they rule out the possibility of deadlock occurrences at the off-line stage. Furthermore, they are generally classified into two types: siphon-based policies [13]–[27] and reachability-graph-based policies [30]–[33]. The latter rely on the reachability analysis and thus suffer from the state explosion problem if applied to large-scale systems. In contrast, siphon-based policies do not require reachability analysis but control deadlocks by structural analysis. In this paper, we focus on siphon-based policies. Specifically, a siphon is a set of places in a PN with a certain property. In particular, whether a siphon is sufficiently marked is closely related to the occurrence of deadlocks. Siphon-based policies prevent a system from reaching deadlocks by controlling siphons, i.e., by making siphons sufficiently marked.

Ezpeleta *et al.* [17] are among the first batch of researchers to deal with deadlock problems. They propose a class of PNs called Systems of Simple Sequential Processes with Resources (S^3PR) that models a class of FMSs and develop a monitor-based liveness-enforcing supervisor for an S^3PR by making every siphon not emptied. Since then, a great many deadlock prevention strategies [13], [14], [31] have been proposed for S^3PR, aiming to solve problems of computational complexity, behavioral permissiveness and structural complexity that exist in the method proposed by Ezpeleta *et al.* It is worth noting that the number of siphons in a PN grows exponentially with respect to the net size. Hence, the complete siphon enumeration is time-consuming. In order to reduce the computation complexity of siphon-based policies, some researchers investigate effective siphon enumeration methods [38]–[41]. Li and Zhou [16] consider the possibility

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

of controlling part of siphons in a PN rather than all siphons when designing policies and thus present a deadlock prevention strategy by controlling elementary siphons only, which is more efficient and obtains a supervisor with lower structural complexity. Also, Abdul-Hussin [18] proposes a deadlock prevention policy based on elementary siphons. Furthermore, some work considers iterative siphon control to avoid the complete siphon enumeration. Huang [15] propose a two-stage deadlock prevention strategy that proceeds in an iterative way. At each iteration, a maximal emptiable siphon is obtained based on the mixed integer programming (MIP) method, which is firstly proposed by Chu and Xie [22]. In addition, Wang *et al*. [24] present a deadlock prevention strategy based on three-stage siphon control. However, these strategies still suffer from the drawbacks of losing some good states as well as a complicated structure of the supervisor. Some work also deals with interesting situations that may appear in application scenarios, such as topics of deadlock control in the presence of uncontrollable and/or unobservable transitions [42], [43] and robust deadlock control with unreliable resources [28], [29].

This work develops a novel iterative method to synthesize a liveness-enforcing supervisor for an S^{3}PR. Note that we consider the case that all transitions are controllable and observable and all resources are reliable. As the existing iterative methods, the proposed method computes a siphon and makes it controlled by adding a control place at each iteration. It is observed that different siphon control orders may lead to supervisors with different structural complexity, computational complexity and behavioral permissiveness. Hence, it is important to decide which siphon is selected to be controlled at each iteration in order to guarantee the good performance of the synthesized supervisor. In this paper, we divide the iterative process into three stages. At each stage, siphons with specific features are controlled. Specifically, we firstly control emptiable siphons containing no control place in Stage 1, then emptiable siphons containing control places in Stage 2, and finally bad siphons in Stage 3. Moreover, roughly speaking, we compute a minimal emptiable siphon with minimal resource places at each iteration in both Stages 1 and 2. Two modified MIP methods are developed respectively to compute such an emptiable siphon at each iteration of Stages 1 and 2. Besides, emptiable siphons in Stages 1 and 2 are all controlled by the invariant-based method [1], which does not remove any good state of the plant system. Concerning Stage 3, a bad siphon is computed and controlled at each iteration by the method proposed by Tricas *et al.* [19].

It is proven that the liveness of the resultant net derived by the proposed three-stage policy is guaranteed. Compared with previous work, the proposed policy usually terminates in fewer iterations and devises a supervisor with lower computational and structural complexities. However, the proposed policy is not necessarily maximally permissive because some good states may be removed in Stage 3. Fortunately, experimental results show that it derives a supervisor introducing no monitor by Stage 3 in most cases, which implies that the supervisor is maximally permissive.

The rest of the paper is organized as follows. Section II reviews basic definitions of PNs, S^{3}PR and S^{4}PR. Section III outlines some concepts about siphons and several methods to compute siphons and monitors. An iterative deadlock prevention strategy consisting of three stages is given in section IV. Section V provides the experimental results of the proposed method. Finally, section VI concludes this paper.

II. PRELIMINARIES

A. PETRI NETS [44]

A*generalized* PN is a four-tuple $N = (P, T, F, W)$ where P is the set of *places* and T is the set of *transitions*. P and T are finite, nonempty, and disjoint sets. $F \subseteq (P \times T) \cup (T \times P)$ is the set of *flow relation* represented by directed arcs from places to transitions or from transitions to places. $W: (P \times T) \cup (T \times P) \to \mathbb{N} = \{0, 1, 2, \dots\}$ assigns a weight to an arc such that $W(x,y) > 0$ if $(x, y) \in F$ and $W(x, y) = 0$ otherwise, where $x, y \in P \cup T$. Given a node $x \in P \cup T$, the *preset* of x is ${}^\bullet x = \{y \in P \cup T | (y, x) \in F\}$ and the *post-set* of x is $x^\bullet = \{y \in P \cup T | (x, y) \in F\}$. $\forall X \subseteq P \cup T$, ${}^\bullet X = \cup_{x \in X} {}^\bullet x$ and $X^\bullet = \cup_{x \in X} x^\bullet$. N is called *ordinary*, denoted as $N = (P, T, F)$, if $W(x,y) = 1, \forall (x, y) \in F$. A *state machine* is an ordinary net such that $|{}^\bullet t| = |t^\bullet| = 1, \forall t \in T$. $N_X = (P_X, T_X, F_X, W_X)$ is said to be a *subnet* of N generated by $P_X \subseteq P$ and $T_X \subseteq T$ if $F_X = F \cap [(P_X \times T_X) \cup (T_X \times P_X)]$ and $W_X(f) = W(f)$, $\forall f \in F_X$.

A *marking* of N is a mapping $M: P \to \mathbb{N}$. The sum of tokens in a place set $S \subseteq P$ at a marking M is denoted as $M(S)$, i.e. $M(S) = \sum_{p \in S} M(p)$. A PN N with its initial marking M_0 is said to be a *net system*, denoted by (N,M_0). The *incidence matrix* of N is a matrix $[N]: P \times T \to \mathbb{Z}$ indexed by P and T such that $[N](p, t) = W(t,p) - W(p, t)$.

A transition t is *enabled* at marking M, denoted by $M[t\rangle$, if $M(p) \geq W(p,t), \forall p \in {}^\bullet t$. t can *fire* at M if it is enabled at M. The firing of t at M reaching a marking M' is denoted as $M[t\rangle M'$, where $M'(p) = M(p) - W(p,t) + W(t,p), \forall p \in P$. Furthermore, a transition sequence $\sigma = t_1 t_2 \dots t_k$ is said to be *enabled* at M, denoted as $M[\sigma\rangle$, if $M[t_1\rangle M_1[t_2\rangle M_2[t_3\rangle \dots M_{k-1}[t_k\rangle$. We use $M[\sigma\rangle M_k$ to denote that marking M_k is reached after the firing of σ from M. The set of all reachable markings of N from M_0 is denoted by $R(N,M_0)$.

A transition t is *live* at a marking M if $\forall M' \in R(N,M)$, $\exists M'' \in R(N,M'), M''[t\rangle$. A transition t is *dead* at a marking M if $\forall M' \in R(N,M)$, t is disabled at M'. A net system (N,M_0) is *live* if $\forall t \in T$, t is live at M_0.

B. S^{4}PR AND S^{3}PR

Definition 1 [26]: A generalized connected self-loop free PN $N = (P, T, F, W)$ is called a *sequential system with shared resources* (S^{4}PR) if all the following conditions are true:

1. $P = P_A \cup P_0 \cup P_R$ is a partition such that
   (a) $P_A = \cup_{i=1}^{n} P_{Ai}$ is the set of *activity places*, where $\forall i, j \in \{1, 2, \ldots, n\}$, $i \neq j$, $P_{Ai} \neq \emptyset$, $P_{Aj} \neq \emptyset$ and $P_{Ai} \cap P_{Aj} = \emptyset$;
   (b) $P_0 = \cup_{i=1}^{n} \{p_o^i\}$ is the set of *idle places*;
   (c) $P_R = \{r_1, r_2, \ldots, r_m\}$ is the set of *resource places* where $m > 0$.
2. $T = \cup_{i=1}^{n} T_i$, where $\forall i, j \in \{1, 2, \ldots, n\}$, $i \neq j$, $T_i \neq \emptyset$, $T_j \neq \emptyset$ and $T_i \cap T_j = \emptyset$.
3. $\forall i \in \{1, 2, \ldots, n\}$, the subnet N_i composed by $P_{Ai} \cup \{p_0^i\} \cup T_i$ is a strongly connected state machine such that every circuit of contains idle place p_0^i.
4. $\forall r \in P_R$, there exists a unique minimal P-semiflow, $I_r \in N^{|P|}$ such that $\{r\} = ||I_r|| \cap P_R$, $P_0 \cap ||I_r|| = \emptyset$, $P_A \cap ||I_r|| \neq \emptyset$, and $I_r(r) = 1$. $H(r) = ||I_r|| \backslash \{r\}$, named the *holders* of r, is the set of activity places that use r. The set of holders of a subset of resources $\Omega \subseteq P_R$ is defined as $H(\Omega) = \cup_{r \in \Omega} H(r)$.
5. $P_A = \cup_{x \in P_R} (||I_r|| \backslash \{r\})$.

Definition2: Let $N = (P_0 \cup P_A \cup P_R, T, F, W)$ be an S^{4}PR. An initial marking M_0 is called *acceptable* for N if 1) $\forall p \in P_0$, $M_0(p) \geq 1$; 2) $\forall p \in P_A$, $M_0(p) = 0$; and 3) $\forall r \in P_R$, $M_0(r) \geq \max\{I_r(p) | \, p \in P_A\}$.

A System of Simple Sequential Process with Resources (S^{3}PR) proposed by Ezpeleta *et al.* [17] is actually a subclass of S^{4}PR. In more detail, given an S^{4}PR$N = (P_A \cup P_0 \cup P_R, T, F, W)$, N is an S^{3}PR if 1) N is an ordinary net; and 2) $\forall p \in P_A$, $^{\bullet\bullet}p \cap P_R = p^{\bullet\bullet} \cap P_R \neq \emptyset$ and $|P^{\bullet\bullet} \cap P_R| = 1$.

III. SIPHON COMPUTATION AND CONTROL

In this section, we firstly introduce notions about siphons. Next, we review the MIP-based method proposed by Chu and Xie [22] that computes a maximal emptiable siphon in a PN system, and the siphon control method based on place invariants [1]. Finally, the notion of bad siphons and their control method are presented.

A. BASIC NOTIONS ABOUT SIPHONS

Given a PN $N = (P, T, F, W)$ and a place set $S \subseteq P$ with $S \neq \emptyset$, S is called a *siphon* if $S^{\bullet} \supseteq {}^{\bullet}S$. We use Π to denote the set of all siphons in N.

A siphon S is said to be *minimal* if $\nexists S' \in \Pi$ such that $S' \subset S$, and said to be *maximal* if $\nexists S' \in \Pi$ such that $S' \supset S$.

Consider the PN shown in Fig 1. We can see that $\Pi = \{S_1 = \{p_2, p_4\text{-}p_6\}, S_2 = \{p_1, p_2, p_4\text{-}p_6\}, S_3 = \{p_2, p_3, p_6\}, S_4 = \{p_1\text{-}p_6\}, S_5 = \{p_1, p_4, p_5\}, S_6 = \{p_2\text{-}p_6\}\}$. Clearly, S_1, S_3, and S_5 are minimal, and S_4 is maximal.

A siphon S is said to be *marked* at a marking M if $M(S) > 0$, and otherwise is said to be *unmarked* at M.

Given a PN system (N, M_0), a siphon S is said to be an *emptiable siphon* if $\exists M \in R(N, M_0)$ such that $M(S) = 0$. Otherwise, S is said to be a non-emptiable siphon.

We note that once a siphon is emptied at a reachable marking of a net system, the deadlock arises. Thus, to avoid deadlocks, it is important to compute emptiable siphons in a net system and make them controlled, i.e., make them marked at all reachable states.

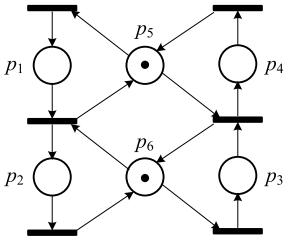


FIGURE 1. A PN system.

Theorem 1 [19]: An ordinary S^{4}PR is live iff there is no emptiable siphon in the net system.

B. EMPTIABLE SIPHON COMPUTATION BY MIP [22]

In this subsection, we review the approach in [22] that computes a maximal emptiable siphon from a bounded net system.

Let (N, M_0) be a bounded net system where $N = (P, T, F, W)$. The approach is based on two main indicators, i.e.,

$$v_p \in \{0, 1\}, \quad \forall p \in P \text{ and } z_t \in \{0, 1\}, \ \forall t \in T.$$

Let S be a siphon. Note that $v_p = 1$ indicates that $p \notin S$ and $z_t = 1$ indicates that $t \notin S^{\bullet}$. On the contrary, $v_p = 0$ indicates that $p \in S$ and $z_t = 0$ indicates that $t \in S^{\bullet}$.

By solving the MIP problem (1)−(8), we can obtain a maximal emptiable siphon. More precisely, all places with $v_p = 0$ in the solution constitute a maximal emptiable siphon.

However, note that the computed siphon is not necessarily emptiable in the net system (N, M_0). This is because constraint (6) not only computes all reachable states of (N, M_0) but also may include some unreachable states of (N, M_0). In other words, it could happen that the computed siphon is emptiable at the unreachable state computed by constraint (6) but is not emptiable at any reachable state of (N, M_0). In this case, the computed siphon is clearly not an emptiable siphon of the given net system.

Moreover, it is true that no maximal emptiable siphon exists in a bounded net system (N, M_0) if no solution exists for the MIP problem (1)-(8).

The MIP problem [22]:

$$G^{MIP} = Maximize(\sum\nolimits_{p \in P} v_p) \tag{1}$$

$$\text{s.t. } z_t \geq \sum\nolimits_{p \in {}^{\bullet}t} v_p - |{}^{\bullet}t| + 1, \quad \forall t \in T \tag{2}$$

$$v_p \geq z_t, \quad \forall (t, p) \in F \tag{3}$$

$$v_p \quad z_t \in \{0, 1\} \tag{4}$$

$$v_p \geq M(p)/B(p), \quad \forall p \in P \tag{5}$$

$$M = M_0 + [N]Y, \quad M \geq 0, \ Y \geq 0 \tag{6}$$

$$B(p) = max\{M(p) \mid M = M_0 + [N]Y\}, \quad M \geq 0, \quad Y \geq 0 \tag{7}$$

$$\sum\nolimits_{p \in P} v_p < |P| \tag{8}$$

C. M-CONTROL OF EMPTIABLE SIPHONS [1]

Yamalidou *et al*. [1] propose a method based on place invariants that enforces linear constraints on the reachable states of a net system by introducing control places (or we say monitors). This method can be used to guarantee a siphon to be a non-emptiable siphon, i.e., guarantee a siphon marked at all reachable states. We review the method as follows.

Suppose that (N, M_0) is a net system to be controlled, which contains n places and m transitions, and the linear constraint on markings is

$$L \cdot M \leq b, \tag{9}$$

where M is the marking vector of the PN model, L is a $1 \times n$ integer vector, and b is an integer.

We use c to denote a control place, $[N_c]$ to denote a $1 \times m$ matrix that shows the connection relationship between the control place c and transitions of the net N, and $M_0(c)$ to denote the initial marking of c. The linear constraint (9) on markings is guaranteed by adding the control place c to the given net system such that

$$[N_c] = -L \cdot [N]; \text{ and}$$
$$M_0(c) = b - L \cdot M_0,$$

where $[N]$ is the incidence matrix of N.

Now, consider the siphon control using the place-invariant method. Let S be an emptiable siphon. The control goal is to guarantee that S is never emptied during the evolution of the system (N, M_0), i.e.,

$$M(S) \geq 1, \quad \forall M \in R(N, M_0) \tag{10}$$

Clearly, constraint (10) can be reformulated as the inequality $L \cdot M \leq b$ with $L(p) = -1$ $\{\forall p \in S\}$ and $b = -1$. The control place c with $[N_c] = -L \cdot [N]$ and $M_0(c) = -1 - L \cdot M_0$ is computed to guarantee that S is a non-emptiable siphon in the resultant net system.

Note that the place-invariant method only removes markings that violate the linear constraint (9) from the reachable markings. It implies that the monitor computed by the place-invariant method for siphon control is maximally permissive since it only forbids markings where the siphon S is unmarked.

In the remainder of this paper, we say that a siphon S is *M-controlled* if a monitor V_S is added for it according to the place-invariant method [1].

D. BAD SIPHON COMPUTATION AND CONTROL [19]

In this subsection, we recall the definition of bad siphons and methods of bad siphon computation and control in [19].

We note that, given a net system containing some weighted arcs, i.e., a generalized PN system, even though there is no emptiable siphon, a deadlock may still arise. In particular, for an S^{4}PR, its liveness is related to the existence of bad siphons. The formal definition of bad siphons is as follows.

Definition 3 [19]: Let (N, M_0) be an S^{4}PR, where $N = (P_0 \cup P_A \cup P_R, T, F, W)$. A siphon S is said to be a *bad siphon* at a marking $M \in R(N, M_0)$, if the following conditions hold:

1) $S_R = S \cap P_R = \{r \in P_R | \exists t \in r^\bullet$ such that $M(r) < W(r, t)$ and $M(^\bullet t \cap P_A) > 0\} \neq \emptyset$;
2) $S_A = S \cap P_A = \{p \in H(S_R) | M(p) = 0\} \neq \emptyset$.

We can see that emptiable siphons are also bad siphons and bad siphons are a notion more general than emptiable siphons. Moreover, the following result holds.

Theorem 2 [19]: An S^{4}PR is live iff there is no bad siphon in the net system.

In other words, to guarantee the liveness of an S^{4}PR, it is not enough to ensure the absence of emptiable siphons. Indeed, it is necessary to guarantee that there is no bad siphon in the net system.

There exists an MIP method proposed by Tricas *et al.* allowing us to compute and control bad siphons in an S^{4}PR. It is proven in Lemma 16 in [19] that the net obtained by adding a control place to an S^{4}PR using the method in [19] is still an S^{4}PR. For the sake of brevity, this method will not be introduced here in order to highlight the focus of this paper. For more details, please refer to [19].

IV. DEADLOCK PREVENTION STRATEGY

In this section, we firstly introduce two rules that specify siphon control orders, which will be involved in our deadlock prevention policy. Next, we define two kinds of emptiable siphons with specific structural features and present two modified MIP problems to compute them. Finally, based on the two modified MIPs and the methods recalled in the last section, an iterative deadlock prevention strategy consisting of three stages is proposed.

A. SIPHON CONTROL ORDERS

The existing iterative deadlock control policies basically compute an emptiable siphon in a ''random'' way at each iteration and add a monitor to make the siphon controlled. Actually, different siphon control orders may lead to supervisors with different behavioral permissiveness, structural complexity as well as computational complexity, which is verified through a large number of case studies.

In more detail, it could happen that redundant control places and/or weighted arcs are introduced by a control order, which however can be avoided if another control order is chosen. Besides, note that the introduction of weighted arcs implies that an original ordinary net is transformed into a generalized net. If the generalized net still suffers from deadlocks, we have to consider deadlock control methods for generalized nets instead of ordinary nets. Clearly, deadlock control methods for generalized nets are often much more complex than those for ordinary nets. In some cases, we cannot optimally control a generalized net in the sense that we have to lose some good states to enforce liveness. In addition, it is easy to see that the number of iterations may be different if we choose different siphon control orders.

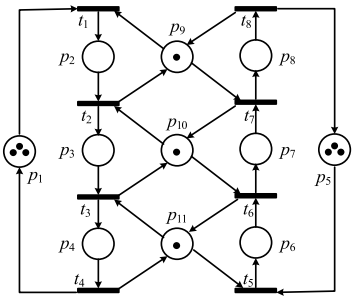


FIGURE 2. An S^{3}PR.

In this paper, we specify the siphon control order in our deadlock control policy mainly based on the following two rules.

Rule1: Emptiable siphons in the original plant system have the priority to be controlled.

Clearly, we consider an S^{3}PR as a plant net. An augmented net after adding some monitors to the S^{3}PR by the place-invariant method is indeed an S^{4}PR since monitors are regarded as resource places. In more detail, it can be checked that condition 4 in Definition 1 is satisfied for each monitor.

Note that, during the introduction of monitors, it could happen that new emptiable siphons that do not exist in the original S^{3}PR appear. Such an emptiable siphon contains some added monitors. In other words, emptiable siphons that we need to consider can be divided into two categories, i.e., those existing and not existing in the original plant system, or equivalently, those containing and not containing monitors.

It can be verified that any monitor added to M-control an emptiable siphon in the original S^{3}PR is connected via unitary arcs, which is due to the structural characteristics of emptiable siphons in an S^{3}PR. In contrast, a monitor added to M-control an emptiable siphon not in the original S^{3}PR may introduce weighted arcs. As we analyze above, deadlock control of generalized nets is often much more complex than ordinary nets. Hence, hoping to introduce weighted arcs as late as possible or ideally avoid weighted arcs, we prefer to M-control emptiable siphons in the original plant system rather than those caused by adding monitors if both of them exist.

Rule 2: Minimal emptiable siphons with minimal number of resource places have the priority to be controlled.

We prefer to control a minimal emptiable siphon rather than a maximal emptiable siphon. This is because once an emptiable siphon is controlled by adding a monitor, some emptiable siphons with more places can be controlled as well. It means that the control of minimal emptiable siphons instead of maximal emptiable siphons may reduce the chance of introducing redundant monitors. In addition, it is discovered that, once emptiable siphons with fewer resource places are controlled by adding monitors, some emptiable siphons with more resource places can be controlled as well. As a result, we prefer to control minimal emptiable siphons with minimal number of resource places.

For example, $S_1 = \{p_3, p_8\text{-}p_{10}\}$, $S_2 = \{p_4, p_7, p_{10}, p_{11}\}$ and $S_3 = \{p_4, p_8\text{-}p_{11}\}$ are three minimal emptiable siphons in the S^{3}PR in Fig. 2. S_1 and S_2 both have two resource places, while S_3 has three. If S_1 and S_2 are firstly M-controlled by adding two monitors, S_3 is always marked at any reachable marking without adding more monitors. On the contrary, if S_3 is firstly M-controlled by adding a monitor, two monitors are still needed to M-control S_1 and S_2, respectively, since otherwise they cannot be controlled.

B. EMPTIABLE SIPHONS WITH SPECIAL CHARACTERISTICS

Motivated by Rules 1 and 2, we define two emptiable siphons in an S^{4}PR. We note that although the plant net in this paper is an S^{3}PR, the intermediate augmented nets after adding monitors by the place-invariant method and/or the method in [19] become S^{4}PR instead of S^{3}PR. Thus, we have to consider siphons in an S^{4}PR as follows.

Definition 4: Let (N, M_0) be an S^{4}PR, where $N = (P_0 \cup P_A \cup P_R, T, F, W)$, and Π_{min-r} be the set of emptiable siphons with minimal number of resource places in (N, M_0). A siphon $S \in \Pi_{min-r}$ is said to be an $\boldsymbol{\omega}$-siphon in (N, M_0) if $S' \in \Pi_{min-r}$ such that $|S' \cap P_A| < |S \cap P_A|$.

Proposition 1: Let (N, M_0) be an S^{4}PR and$S$ be a siphon. If S is an $\boldsymbol{\omega}$-siphon, S is a minimal emptiable siphon.

Proof: It is obvious that S is emptiable. Now we prove that S is minimal. By contradiction, suppose that S is not minimal. That is to say, there exists a siphon S' such that $S' \subset S$. Since $S' \subset S$, we can see $|S' \cap P_A| < |S \cap P_A|$ or/and $|S' \cap P_R| < |S \cap P_R|$, which contradicts that S is an $\boldsymbol{\omega}$-siphon. Therefore, S is a minimal emptiable siphon. ■

Definition 5: Let (N, M_0) be an S^{4}PR, where $N = (P_0 \cup P_A \cup P_R, T, F, W)$, and $P_c \subseteq P_R$ be a set of resource places. Let Π' be the set of emptiable siphons containing no places of P_c in (N, M_0) and Π'_{min-r} be the set of siphons with minimal number of resource places in Π'.A siphon $S \in \Pi'_{min-r}$ is said to be a P_c-excluded $\boldsymbol{\omega}$-siphon if $S' \in \Pi'_{min-r}$ such that $|S' \cap P_A| < |S \cap P_A|$.

Proposition 2: Let (N, M_0) be an S^{4}PR, where $N = (P_0 \cup P_A \cup P_R, T, F, W)$, $P_c \subseteq P_R$ be a set of resource places, and S be a siphon. If S is a P_c-excluded $\boldsymbol{\omega}$-siphon, S is a minimal emptiable siphon in (N, M_0).

Proof: Similar to the proof of Proposition 1. ■

C. TWO MODIFIED MIPS TO COMPUTE EMPTIABLE SIPHONS

In this subsection, we present two modified MIPs, denoted as MMIP-1 and MMIP-2, to compute an $\boldsymbol{\omega}$-siphon and a P_c -excluded $\boldsymbol{\omega}$-siphon given a set of resource places P_c from an S^{4}PR, respectively.

First, we introduce MMIP-1 as follows, which computes an $\boldsymbol{\omega}$-siphon in an S^{4}PR (N, M_0) with $N = (P_0 \cup P_A \cup P_R, T, F, W)$.

MMIP-1:

$$G^{\text{MMIP1}} = Maximize(\sum_{p\in P_R} |P_A| \cdot v_p + \sum_{p\in P_A} v_p) \quad (11)$$

s.t. constraints (2-8).

We can see that MMIP-1 has a new objective function (11) compared to the original MIP. It is clear that the solution space of constraints (2)–(8) describes the set of all emptiable siphons in the net system (N, M_0). The objective function (11) actually implies that it firstly searches emptiable siphons with minimal resource places and then selects one with minimal activity places from them, i.e. an $\boldsymbol{\omega}$-siphon. In the following, we present and prove some results related to MMIP-1.

Proposition 3: Given an S^{4}PR (N, M_0), a solution of MMIP-1 corresponds to an $\boldsymbol{\omega}$-siphon.

Proof: Let $V_P = (v_{p1}, v_{p2}, \ldots, v_{pn}) \in \{0, 1\}^n$ be a solution of MMIP-1, where n is the number of places in the S^{4}PR, and S be the place set corresponding to V_P, i.e., $S = \{p \in P | v_p = 0\}$. It is known that the solution space of constraints (2)–(8) describes the set of all emptiable siphons of the S^{4}PR [22]. Hence, S is an emptiable siphon.

Let Π_{min-r} be the set of emptiable siphons with minimal number of resource places in N.

Now, we prove that $S \in \Pi_{min-r}$. By contradiction, suppose that the number of resource places of S is not minimal. That is to say, there exists a minimal emptiable siphon S' that contains fewer resource places than S. Construct a vector $V'_P = (v'_{p1}, v'_{p2}, \ldots, v'_{pn}) \in \{0, 1\}^n$ corresponding to S' such that $\forall p \in P$, $v'_p = 0$ *iff* $p \in S'$. Clearly, V'_P is a solution to constraints (2)–(8) since S' is an emptiable siphon. Since S' has fewer resource places, we can see that $\sum_{p\in P_R} v'_p > \sum_{p\in P_R} v_p$. Thus $\sum_{p\in P_R} v_{p'} - \sum_{p\in P_R} v_p \geq 1$. By multiplying both sides with $|P_A|$, i.e. the number of activity places, we can derive that $\sum_{p\in P_R} |P_A| \cdot v'_p - \sum_{p\in P_R} |P_A| \cdot v_p \geq |P_A|$. Note that an emptiable siphon in an S^{4}PR consists of resource places and activity places. For activity places, we can see that $\sum_{p\in P_A} v_p - \sum_{p\in P_A} v'_p < |P_A|$ since $1 < \sum_{p\in P_A} v_p \leq |P_A|$ and $1 < \sum_{p\in P_A} v'_p \leq |P_A|$. Hence, it is trivial to see that $\sum_{p\in P_R} |P_A| \cdot v'_p + \sum_{p\in P_A} v'_p > \sum_{p\in P_R} |P_A| \cdot v_p + \sum_{p\in P_A} v_p$, which contradicts that V_P is the solution of MMIP-1. Therefore, S has minimal numbers of resource places, i.e. $S \in \Pi_{min-r}$.

Next, we prove that $S' \in \Pi_{min-r}$ such that $|S' \cap P_A| < |S \cap P_A|$. By contradiction, there exists a minimal emptiable siphon $S' \in \Pi_{\text{min-r}}$ that contains fewer activity places than S. Construct a vector $V'_P = (v'_{p1}, v'_{p2}, \ldots, v'_{\text{pn}}) \in \{0, 1\}^n$ corresponding to S' such that $\forall p \in P$, $v'_p = 0$ *iff* $p \in S'$. Since S' is an emptiable siphon, V'_P is a solution to constraints (2)–(8). Since S' has fewer activity places and at the same time contains the same number of resource places, it is trivial to see that $\sum_{p\in P_R} |P_A| \cdot v'_p + \sum_{p\in P_A} v'_p > \sum_{p\in P_R} |P_A| \cdot v_p + \sum_{p\in P_A} v_p$, which contradicts that V_P is the solution of MMIP-1.

Therefore, S is an $\boldsymbol{\omega}$-siphon. ■

Proposition 4: Given an S^{4}PR (N, M_0), it contains no emptiable siphon if no solution exists for MMIP-1.

Proof: Since no solution exists for MMIP-1, it means that the solution space of constraints (2)–(8) is empty. Since the solution space of constraints (2)–(8) describes the set of all emptiable siphons in the net system, clearly, the net system contains no emptiable siphon.

Now, we introduce MMIP-2 that is applicable to an S^{4}PR given a set $P_c \subseteq P_R$. The only difference between MMIP-1 and MMIP-2 is the introduction of constraint (12) in MMIP-2. It is easy to see that, due to the constraint (12), an emptiable siphon excluding places in P_c is computed by solving the MMIP-2 problem. To be exact, MMIP-2 computes a P_c-excluded $\boldsymbol{\omega}$-siphon.

MMIP-2:

$G^{\text{MMIP2}} = Maximize(\sum_{p\in p_R} |P_A| \cdot v_p + \sum_{p\in P_A} v_p)$

s.t. constraints (2-8) and

$$v_p = 1, \quad \forall p \in P_C \quad (12)$$

Proposition 5: Given an S^{4}PR (N, M_0) with $N = (P_0 \cup P_A \cup P_R, T, F, W)$, and a set of resource places $P_c \subseteq P_R$, a solution of MMIP-2 corresponds to a P_c-excluded $\boldsymbol{\omega}$-siphon.

Proof: Similar to the proof of Proposition 3. ■

Proposition 6: Given an S^{4}PR (N, M_0) and a set of resource places $P_c \subseteq P_R$, it contains no emptiable siphon that excludes places in P_c if no solution exists for MMIP-2.

Proof: Similar to the proof of Proposition 4. ■

D. AN ITERATIVE DEADLOCK PREVETION STRATEGY

In this subsection, we propose a new iterative deadlock prevention policy for S^{3}PR. For simplicity, we call it G-policy.

As the existing iterative deadlock prevention policies, G-policy computes and controls siphons iteratively and finally synthesizes a liveness-enforcing supervisor for an S^{3}PR. The novelty of G-policy is that it consists of three stages that specify the control order on three kinds of siphons, i.e., P_c-excluded $\boldsymbol{\omega}$-siphons, $\boldsymbol{\omega}$-siphons and bad siphons, where P_c is the set of monitors added to the original S^{3}PR. Note that, in the remainder of this paper, P_c is assumed to be the set of monitors added to the original S^{3}PR by default. We explain the three stages in more detail as follows.

In Stage 1 (i.e., Steps 3-6), we compute and M-control P_c-excluded $\boldsymbol{\omega}$-siphons iteratively. More specifically, we look for a minimal emptiable siphon with minimal number of resource places, which contains no monitors, by the siphon detection method MMIP-2 at each iteration. Once such a siphon is computed, a control place is added, making the siphon M-controlled. We note that the resultant net from Stage 1 is an S^{4}PR if all control places are regarded as resource places and the S^{4}PR is guaranteed to be ordinary. In addition, there is no emptiable siphon that excludes control places in the resultant net by Proposition 6. In other words, all emptiable siphons in the original S^{3}PR have been controlled.

In Stage 2 (i.e., Steps 7-9), we compute and M-control $\boldsymbol{\omega}$-siphons iteratively. At each iteration, an $\boldsymbol{\omega}$-siphon is computed by solving an MMIP-1 problem. It is clear that the $\boldsymbol{\omega}$-siphons computed in Stage 2 definitely contain some control places since no emptiable siphon excluding control places

G-policy 1 A Three-Stage Deadlock Prevention Policy

Input: An S^3PR net system (N, M_0)
Output: A live S^4PR net system (N^c, M_0^c)
1) Let $(N^c, M_0^c) := (N, M_0)$;
2) $P_c := \emptyset$;
/* * * Stage One: P_c-excluded $\boldsymbol{\omega}$-siphon Control * * */
3) **while** there exists a P_c-excluded $\boldsymbol{\omega}$-siphon S in (N^c, M_0^c) computed by solving an MMIP-2 **do**
4) add a monitor Vs to (N^c, M_0^c) such that S is M-controlled and denote the resultant net as (N^c, M_0^c);
5) $P_c := P_c \cup \{Vs\}$;
6) **end while**
/* * * Stage Two: $\boldsymbol{\omega}$-siphon Control * * */
7) **while** there exists an $\boldsymbol{\omega}$-siphon S in (N^c, M_0^c) computed by solving an MMIP-1 **do**
8) add a monitor Vs to (N^c, M_0^c) such that S is M-controlled and denote the resultant net as (N^c, M_0^c);
9) **end while**
/* * * Stage Three: BadSiphon Control * * */
10) **if** there exists a weighted arc in (N^c, M_0^c) **then**
11) **while** there exists a bad siphon S in (N^c, M_0^c) computed using the method in [19] **do**
12) add a monitor Vs to (N^c, M_0^c) according to [19] and denote the resultant net as (N^c, M_0^c);
13) **end while**
14) **end if**
15) **End**

exists in the resultant net of Stage 1. In addition, we note that weighted arcs may be introduced while adding monitors in Stage 2. Thus, the resultant net after Stage 2 may be not ordinary. But it is still an S^4PR if all control places are regarded as resource places. In addition, the resultant net after Stage 2 contains no emptiable siphons by Proposition 4.

Concerning Stages 1 and 2, we can say that the difference between them is that we look for and control emptiable siphons in the original plant system in Stage 1, while we look for and control emptiable siphons in the augmented net that contain control places in Stage 2.

In Stage 3 (i.e., Steps 10-14), we firstly determine if the resultant net after Stage 2 is ordinary. If so, it is already a live net system according to Theorem 1. Otherwise, it may be not live since the liveness of a generalized net cannot be guaranteed by the absence of emptiable siphons. In this case, we compute and control bad siphons iteratively using the method in [19]. The resultant net after Stage 3 contains no bad siphon. By Theorem 2, a live net system is obtained after the control of three stages.

Based on the above analysis, we can derive the following result.

Theorem 3: Let (N, M_0) be an S^3PR. The net (N^c, M_0^c) resulting from applying G-policy to (N, M_0) is a live S^4PR.

Proof: Note that S^3PR is a subclass of S^4PR. It is clear that the resultant net after applying G-policy to an S^3PR is an S^4PR. Besides, if the resultant net is ordinary, G-policy guarantees that it has no emptiable siphons, which indicates it is live by Theorem 1. If the resultant net is generalized, G-policy guarantees that it contains no bad siphons, which implies it is live by Theorem 2. Hence, the final resultant net is a live S^4PR. ■

In the following, we make some comments on G-policy in terms of structural complexity, computational complexity and behavioral permissiveness.

Remark1 (Behavioral Permissiveness): Although G-policy computes a liveness-enforcing supervisor, the supervisor is not necessarily maximally permissive in the sense that some good states of the plant net system may be missing in the controlled net system. Indeed, if the resultant net after Stage 2 is ordinary, the computed supervisor is definitely maximally permissive since M-control of an emptiable siphon does not forbid any good states of the plant net system; otherwise, the behavioral permissiveness of the supervisor computed by G-policy depends on the computation and control of bad siphons in Stage 3. Since the control of a bad siphon by the method in [19] may forbid good states of the plant net system, the final computed supervisor is not guaranteed to be maximally permissive.

However, it is worth noting that large quantities of numerical examples show that no bad siphon can be found in Stage 3 when G-policy is applied. In other words, the computed supervisor is liveness-enforcing and maximally permissive. Hence, we conjecture that the liveness-enforcing supervisor for an S^3PR computed by G-policy is guaranteed to be maximally permissive but it is not proved now.

Remark 2 (Computational Complexity): We can see that G-policy does not require the reachability analysis, which in general leads to the state explosion problem. Besides, G-policy can avoid the complete siphon enumeration, which is time-consuming especially in large-size nets since the number of siphons in a net grows exponentially with respect to the net size. As a result, G-policy typically has lower computational complexity than those approaches requiring the reachability analysis or the complete siphon enumeration.

Remark 3 (Structural Complexity): As we analyzed before, an inappropriate siphon control order can introduce redundant control places, thereby increasing the structural complexity of the final supervisor. G-policy controls emptiable siphons in a specified order, aiming to introduce as few control places and weighted arcs as possible to simplify the structure of the supervisor. Typically, compared with most existing siphon-based deadlock prevention policies that iteratively control siphons, G-policy computes a supervisor with a simpler structure. Note that it does not guarantee that the structure of the supervisor is definitely the simplest.

Finally, we make a comparison between G-policy and the policy proposed in the paper [24]. G-policy is similar

TABLE 1. Siphon computation and control at each iteration in example 1.

Stage	MIPs	Siphon	Monitors $V_{s(i)}$
1	1	S_1 p_4, p_7, p_{10}, p_{11}	$V_{s(1)}$
	2	S_2 p_3, p_8, p_9, p_{10}	$V_{s(2)}$
	3	No P_c-excluded $\boldsymbol{\omega}$-siphon exists.	—
2	4	S_3 $p_3, p_7, V_{s(1)}, V_{s(2)}$	$V_{s(3)}$
	5	No $\boldsymbol{\omega}$-siphon exists.	—
3	—	—	—

TABLE 2. Monitors added in example 1.

$V_{s(i)}$	$M_0(V_{s(i)})$	*Pre*	*Post*
$V_{s(1)}$	1	t_3, t_6	t_2, t_5
$V_{s(2)}$	1	t_2, t_7	t_1, t_6
$V_{s(3)}$	1	t_2, t_6	t_1, t_5

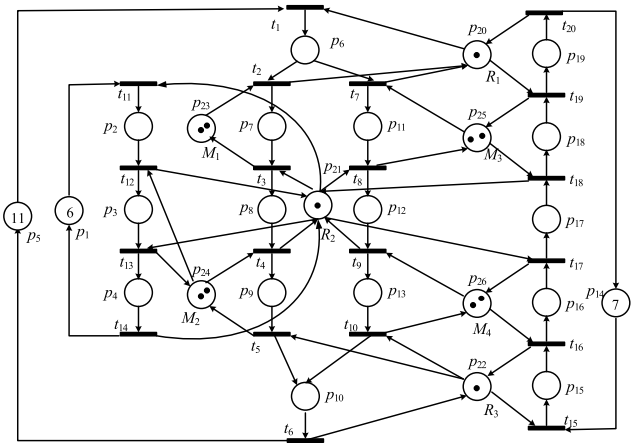


FIGURE 3. An S^{3}PR.

to the policy in [24]. Both of them consist of three stages and iteratively control siphons in each stage. In more detail, both of them iteratively and selectively search and control emptiable siphons based on solving MIP problems in the first two stages, and search and control bad siphons by the method in [19] in the third stage. The difference between the two policies mainly lies in the search of emptiable siphons in the first two stages. The policy in [24] finds a maximal emptiable siphon at each iteration and then extracts a minimal emptiable siphon from it. In contrast, G-policy finds a minimal emptiable siphon directly at each iteration by solving MMIP-1 or MMIP-2 problem. In addition, the policy in [24] extracts a minimal emptiable siphon from a maximal emptiable siphon in a "random" way, while G-policy breaks the random mechanism, searching and controlling siphons with specific structural characteristics. G-policy is indeed the improved version of the policy in [24], which is verified by illustrative examples in the next section.

V. EXAMPLES

In this section, we provide two numerical examples to demonstrate the proposed policy and compare it with some other methods in the literature. Here, the software "Lingo" is used as a tool to solve MIP problems, and the software "INA" to verify and analyze the final results.

A. EXAMPLE I

Consider the S^{3}PR (N, M_0) in Fig. 2. Let us compute the liveness-enforcing supervisor for it by G-policy.

In Stage 1, we repeatedly compute a P_c-excluded $\boldsymbol{\omega}$-siphon by solving an MMIP-2 problem and then make it M-controlled. Firstly, the P_c-excluded $\boldsymbol{\omega}$-siphon $S_1 = \{p_4, p_7, p_{10}, p_{11}\}$ is found in (N, M_0), where $P_c = \emptyset$, and the monitor $V_{s(1)}$ is added to (N, M_0) such that S_1 is M-controlled, resulting in an augmented net denoted as (N^c, M_0^c). Now, P_c is updated as $P_c = \{V_{s(1)}\}$. Next, the P_c-excluded $\boldsymbol{\omega}$-siphon $S_2 = \{p_4, p_6, p_{10}, p_{11}\}$ is found in (N^c, M_0^c) and (N^c, M_0^c) is updated by introducing the monitor $V_{s(2)}$ that makes S_2 M-controlled. Accordingly, P_c is updated as $P_c = \{V_{s(1)}, V_{s(2)}\}$. Then, no P_c-excluded $\boldsymbol{\omega}$-siphon can be found by solving an MMIP-2 problem, which means the termination of Stage 1. Now all emptiable siphons in the original S^{3}PR (N, M_0) have been controlled in the resultant net of Stage 1.

In Stage 2, we repeatedly compute an $\boldsymbol{\omega}$-siphon by solving an MMIP-1 problem and then make it M-controlled. At the first iteration, the $\boldsymbol{\omega}$-siphon $S_3 = \{p_3, p_7, V_{s(1)}, V_{s(2)}\}$ is found and the monitor $V_{s(3)}$ is designed to M-control S_3. Then no $\boldsymbol{\omega}$-siphon is found at the next iteration. Hence, Stage 2 terminates. Now, there exists no emptiable siphon in the resultant net.

We can easily determine that the resultant net after Stage 2 is ordinary, which indicates that it is already live by Theorem 1. Hence, there is no need to perform Stage 3.

To be intuitive, Table 1 shows the selected siphon and the monitor added at each iteration. Specifically, the first and second columns indicate the stage numbers and the iteration numbers, respectively, the third column presents the computed siphon at each iteration, and the last column shows the name of each added control place. The details of added control places are shown in Table 2, where the initial marking $(M_0(c_i))$, pre-transitions $({}^\bullet p_{ci})$ and post-transitions $(p_{ci}^\bullet)$ are presented.

B. EXAMPLE II

Consider the S^{3}PR (N, M_0) shown in Fig. 3, which is the PN model of an FMS firstly proposed in [17] and then widely used to evaluate the performance of deadlock control strategies in, e.g., [45]. The system contains 26750 reachable states with 21581 good states. In the case that the system is optimally controlled, or equivalently, the liveness-enforcing supervisor is maximally permissive, the controlled system should have 21581 reachable states that are all good states.

Let us see the performance of G-policy applying to this system. As shown in Table 3, seven P_c-excluded $\boldsymbol{\omega}$-siphons and

TABLE 3. Siphon computation and control at each iteration in example 2.

Stage	MIPs	Siphon		Monitors $V_{s(i)}$
1	1	S_1	$p_{10}, p_{18}, p_{22}, p_{26}$	$V_{s(1)}$
	2	S_2	$p_4, p_9, p_{12}, p_{17}, p_{21}, p_{24}$	$V_{s(2)}$
	3	S_3	$p_2, p_4, p_8, p_{12}, p_{16}, p_{21}, p_{25}$	$V_{s(3)}$
	4	S_4	$p_2, p_4, p_8, p_{13}, p_{17}, p_{21}, p_{26}$	$V_{s(4)}$
	5	S_5	$p_2, p_4, p_8, p_{10}, p_{17}, p_{21}, p_{22}, p_{26}$	$V_{s(5)}$
	6	S_6	$p_4, p_{10}, p_{17}, p_{21}, p_{22}, p_{24}, p_{26}$	$V_{s(6)}$
	7	S_7	$p_2, p_4, p_8, p_{12}, p_{15}, p_{20}, p_{21}, p_{23}, p_{25}$	$V_{s(7)}$
	8	No P_c-excluded $\boldsymbol{\omega}$-siphon exists.		—
2	9	S_8	$p_{12}, p_{17}, V_{s(3)}, V_{s(4)}$	$V_{s(8)}$
	10	S_9	$p_{10}, p_{17}, p_{22}, V_{s(3)}, V_{s(5)}, V_{s(8)}$	$V_{s(9)}$
	11	S_{10}	$p_4, p_{10}, p_{12}, p_{17}, p_{21}, p_{22}, p_{24}, V_{s(8)}$	$V_{s(10)}$
	12	S_{11}	$p_2, p_3, p_{10}, p_{15}, p_{20}, p_{22}, p_{23}, V_{s(6)}, V_{s(7)}, V_{s(8)}$	$V_{s(11)}$
	13	S_{12}	$p_2, p_3, p_{10}, p_{15}, p_{20}, p_{22}, p_{23}, p_{25}, p_{26}, V_{s(6)}, V_{s(7)}$	$V_{s(12)}$
	14	S_{13}	$p_2, p_4, p_8, p_{10}, p_{16}, p_{21}, p_{22}, p_{26}, V_{s(12)}$	$V_{s(13)}$
	15	S_{14}	$p_2, p_4, p_8, p_{10}, p_{15}, p_{20}, p_{21}, p_{22}, p_{23}, p_{25}, V_{s(4)}, V_{s(12)}$	$V_{s(14)}$
	16	No $\boldsymbol{\omega}$-siphon exists.		—
3	17	No bad siphon exists.		—

TABLE 4. Monitors added in example 2.

Stage	Monitors			
	$V_{s(i)}$	$M_0(V_{s(i)})$	*Pre*	*Post*
1	$V_{s(1)}$	2	t_{10}, t_{16}	t_9, t_{15}
	$V_{s(2)}$	2	t_4, t_{13}	t_3, t_{11}
	$V_{s(3)}$	2	t_8, t_{18}	t_7, t_{17}
	$V_{s(4)}$	2	t_9, t_{17}	t_8, t_{16}
	$V_{s(5)}$	3	t_{10}, t_{17}	t_8, t_{15}
	$V_{s(6)}$	5	$t_5, t_{10}, t_{13}, t_{17}$	t_3, t_8, t_{11}, t_{15}
	$V_{s(7)}$	5	t_3, t_8, t_{19}	t_1, t_{17}
2	$V_{s(8)}$	3	t_8, t_{17}	t_7, t_{16}
	$V_{s(9)}$	8	$t_8, t_{10}, 2t_{17}$	$2t_7, 2t_{15}$
	$V_{s(10)}$	6	t_5, t_8, t_{13}, t_{17}	t_3, t_7, t_{11}, t_{15}
	$V_{s(11)}$	16	$t_3, t_5, t_8, t_{10}, t_{17}, t_{19}$	$2t_1, 2t_{15}$
	$V_{s(12)}$	17	$t_3, t_5, t_8, 2t_{10}, t_{17}, 2t_{19}$	$2t_1, t_9, 2t_{15}, t_{18}$
	$V_{s(13)}$	20	$t_3, t_5, 3t_{10}, t_{17}, t_{18}, t_{19}$	$2t_1, t_9, 3t_{15}$
	$V_{s(14)}$	25	$2t_3, t_5, t_9, 2t_{10}, t_{17}, 3t_{19}$	$3t_1, 3t_{15}, t_{18}$

TABLE 5. Performance comparison.

Different Strategies	Number of reachable states	Number of monitors	Number of additional arcs	Complete siphon enumeration	Reachability graph	Computational complexity
Strategy in [35]	166	0	0	√	×	Exponential
Strategy in [34]	2480	7	38	×	×	Polynomial
Strategy in [17]	6287	18	106	√	×	Exponential
Strategy in [16]	6287	6	32	√	×	Exponential
Strategy in [20]	12656	16	88	×	×	NP-hard
Strategy in [19]	14850	8	40	×	×	NP-hard
Strategy in [23]	15999	6	29	×	×	NP-hard
Strategy in [15]	16425	7	34	√	×	Exponential
Strategy in [32]	21562	19	112	×	√	Exponential
Strategy in [33]	21581	17	100	×	√	Exponential
Strategy in [25]	21581	15	95	×	×	NP-hard
Strategy in [24]	21562	15	103	×	×	NP-hard
G-policy	21581	14	86	×	×	NP-hard

seven $\boldsymbol{\omega}$-siphons are computed and M-controlled by monitors in Stage 1 and Stage 2, respectively. According to the information of added monitors shown in Table 4, we can see that the resultant net of Stage 1 is still ordinary, but the monitors added in Stage 2 introduce weighted arcs, which forces us to search bad siphons in Stage 3. Fortunately, by solving an MIP problem according to [19], no bad siphon can be found. Thus, we obtain the final resultant system that is live.

We note that the liveness-enforcing supervisor is maximally permissive since no monitor is added in Stage 3. Indeed, it can be checked by "INA" that the controlled system is live with 21581 reachable states. In addition, we can see that 14 monitors are added in total by G-policy, where $V_{s(11)}$ is actually the only redundant monitor through the redundancy analysis.

Finally, we compare G-policy with other deadlock prevention strategies in the literature by applying them to the net system in Fig. 3 as well. The performance of all the considered strategies is summarized in Table 5.

As shown in Table 5, strategies in [33] and [25] also result in maximally permissive liveness-enforcing supervisors for the S^3PR in Fig. 3. We can see that the computation of the reachability graph is required by the strategy in [33], leading to its exponential computational complexity. In contrast, the computational complexities of G-policy and the strategy in [25] are both NP-hard. Moreover, it is clear that both the supervisors obtained by strategies in [33] and [25] contain more monitors and additional arcs than the supervisor obtained by G-policy.

Concerning the structural complexities of strategies in Table 5, we can see that all of the supervisors that contain less monitors than the supervisor obtained by G-policy lose 5000 or even much more good states, while G-policy reserves all good states, i.e., is maximally permissive.

Finally, let us focus on the strategy in [34], which is the only strategy in Table 5 with polynomial complexity. Although it has the advantage in computational complexity, it loses almost 20000 good states.

Consequently, we may conclude that, for the S^3PR in Fig. 3, G-policy is the best one among all the strategies in Table 5 concerning the overall performance on structural complexity, behavioral permissiveness and computational complexity.

VI. CONCLUSION AND FUTURE WORK

In this paper, a three-stage iterative deadlock prevention policy named G-policy is proposed for S^{3}PR that specifies the control order on siphons. Specifically, we firstly iteratively control P_c-excluded $\boldsymbol{\omega}$-siphons in Stage 1, where P_c is the set of monitors added to the plant system, then $\boldsymbol{\omega}$-siphons in Stage 2, and finally bad siphons in Stage 3. In particular, two modified MIPs are developed to compute a P_c-excluded $\boldsymbol{\omega}$-siphon and an $\boldsymbol{\omega}$-siphon in Stages 1 and 2, respectively. Compared with the existing policies, G-policy usually synthesizes a liveness-enforcing supervisor with lower computational, structural complexities and higher behavioral permissiveness. In addition, the supervisor is guaranteed to be maximally permissive in the case that no bad siphon can be found in Stage 3.

In our future work, we intend to design a siphon control order for a class of PNs that allows us to obtain a liveness-enforcing supervisor by M-controlling emptiable siphons only. In this case, the supervisor can be guaranteed to be maximally permissive. Also, we investigate how to achieve the minimality on the supervisor structure. Moreover, we plan to consider extending G-policy to more general PN systems than S^{3}PR.

REFERENCES

[1] K. Yamalidou, J. Moody, M. Lemmon, and P. Antsaklis, "Feedback control of Petri nets based on place invariants," *Automatica*, vol. 32, no. 1, pp. 15–28, Jan. 1996.

[2] F. Yang, N. Q. Wu, Y. Qiao, and R. Su, "Polynomial approach to optimal one-wafer cyclic scheduling of treelike hybrid multi-cluster tools via Petri nets," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 270–280, Jan. 2018.

[3] D. Xiang, G. Liu, C. Yan, and C. Jiang, "Detecting data-flow errors based on Petri nets with data operations," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 251–260, Jan. 2018.

[4] Q. Zhu, M. Zhou, Y. Qiao, and N. Wu, "Petri net modeling and scheduling of a close-down process for time-constrained single-arm cluster tools," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 3, pp. 389–400, Mar. 2018.

[5] H. Zhang, L. Feng, and Z. Li, "A learning-based synthesis approach to the supremal nonblocking supervisor of discrete-event systems," *IEEE Trans. Autom. Control*, vol. 63, no. 10, pp. 3345–3360, Oct. 2018.

[6] X. Wang and H. Hu, "A robust control approach to automated manufacturing systems allowing multitype and multiquantity of resources with Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published. doi: 10.1109/TSMC.2018.2852946.

[7] Y. Yang and H. Hu, "A distributed control approach to automated manufacturing systems with complex routes and operations using Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published. doi: 10.1109/TSMC.2018.2883083.

[8] H. Hu, Y. Liu, and L. Yuan, "Supervisor simplification in FMSs: Comparative studies and new results using Petri nets," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 1, pp. 81–95, Jan. 2016.

[9] Y. Chen, Z. Li, K. Barkaoui, and A. Giua, "On the enforcement of a class of nonlinear constraints on Petri nets," *Automatica*, vol. 55, pp. 116–124, May 2015.

[10] Y. Chen, Z. Li, K. Barkaoui, and M. Uzam, "New Petri net structure and its application to optimal supervisory control: Interval inhibitor arcs," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 10, pp. 1384–1400, Oct. 2014.

[11] Y. Chen and G. Liu, "Computation of minimal siphons in Petri nets by using binary decision diagrams," *Acm Trans. Embedded Comput. Syst.*, vol. 12, no. 1, pp. 1–15, 2013.

[12] M. H. Abdul-Hussin and Z. A. Banaszak, "On liveness and a class of generalized Petri nets," in *Proc. 8th Annu. Ind. Automat. Electromech. Eng. Conf. (IEMECON)*, Aug. 2017, pp. 257–267.

[13] K. Barkaoui and I. B. Abdallah, "A deadlock prevention method for a class of FMS," in *Proc. IEEE Int. Conf. Syst., Man Cybern. Intell. Syst. 21st Century*, vol. 5, Oct. 1995, pp. 4119–4124.

[14] D. Y. Chao, "Computation of elementary siphons in Petri nets for deadlock control," *Comput. J.*, vol. 49, no. 4, pp. 470–479, Jul. 2006.

[15] Y.-S. Huang, "Design of deadlock prevention supervisors using Petri nets," *Int. J. Adv. Manuf. Technol.*, vol. 35, nos. 3–4, pp. 349–362, 2007.

[16] Z. Li and M. Zhou, "Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 34, no. 1, pp. 38–51, Jan. 2004.

[17] 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.

[18] M. H. Abdul-Hussin, "Design of a Petri net based deadlock prevention policy supervisor for S^{3}PR," in *Proc. 6th Int. Conf. Intell. Syst., Modelling Simulation*, Feb. 2015, pp. 46–52.

[19] F. Tricas, F. Garcia-Valles, J. M. Colom, and J. Ezpeleta, "A Petri net structure-based deadlock prevention solution for sequential resource allocation systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 271–277.

[20] Y. Huang, M. Jeng, X. Xie, and S. Chung, "Deadlock prevention policy based on Petri nets and siphons," *Int. J. Prod. Res.*, vol. 39, no. 2, pp. 283–305, 2001.

[21] M. H. Abdul-Hussin and Z. A. Banaszak, "Siphon-based deadlock prevention for a class of S^{4}PR generalized Petri nets," in *Proc. Int. Conf. Control, Autom. Inf. Sci. (ICCAIS)*, Oct./Nov. 2017, pp. 239–244.

[22] F. Chu and X.-L. Xie, "Deadlock analysis of Petri nets using siphons and mathematical programming," *IEEE Trans. Robot. Autom.*, vol. 13, no. 6, pp. 793–804, Dec. 1997.

[23] Z. W. Li and M. C. Zhou, "Two-stage method for synthesizing liveness-enforcing supervisors for flexible manufacturing systems using Petri nets," *IEEE Trans. Ind. Informat.*, vol. 2, no. 4, pp. 313–325, Nov. 2006.

[24] S. Wang, C. Seatzu, and L. Huang, "A three-stage deadlock prevention strategy for S^{3}PR nets," in *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, Aug. 2015, pp. 286–291.

[25] Q. Zhuang, W. Dai, S. Wang, and F. Ning, "Deadlock prevention policy for S^{4}PR nets based on siphon," *IEEE Access*, vol. 6, pp. 50648–50658, 2018.

[26] F. Tricas, F. García-Vallés, J. M. Colom, and J. Ezpeleta, "An iterative method for deadlock prevention in FMS," in *Discrete Event Systems*, R. Boel and G. Stremersch, Eds. Boston, MA, USA: Springer, 2000, pp. 139–148.

[27] S. Wang, M. Zhou, and W. Wu, "Design of a maximally permissive liveness-enforcing supervisor with reduced complexity for automated manufacturing systems," *Asian J. Control*, vol. 17, no. 1, pp. 190–201, Jan. 2015.

[28] F. Wang, K.-Y. Xing, M.-C. Zhou, X.-P. Xu, and L.-B. Han, "A robust deadlock prevention control for automated manufacturing systems with unreliable resources," *Inf. Sci.*, vol. 345, pp. 243–256, Jun. 2016.

[29] J. Luo, Z. Liu, M. Zhou, K. Xing, X. Wang, X. Li, and H. Liu, "Robust deadlock control of automated manufacturing systems with multiple unreliable resources," *Inf. Sci.*, vol. 479, pp. 401–415, Apr. 2019.

[30] Y. Chen, Z. Li, and K. Barkaoui, "Maximally permissive liveness-enforcing supervisor with lowest implementation cost for flexible manufacturing systems," *Inf. Sci.*, vol. 256, no. 1, pp. 74–90, Jan. 2014.

[31] Y. Chen, Z. Li, and M. Zhou, "Behaviorally optimal and structurally simple liveness-enforcing supervisors of flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 42, no. 3, pp. 615–629, May 2012.

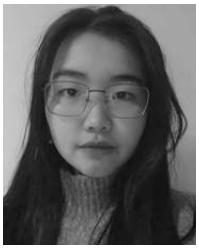
[32] M. Uzam and M. C. Zhou, "An improved iterative synthesis method for liveness enforcing supervisors of flexible manufacturing systems," *Int. J. Prod. Res.*, vol. 44, no. 10, pp. 1987–2030, 2006.

[33] 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.

[34] 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.

[35] N. Wu, M. Zhou, and G. Hu, "One-step look-ahead maximally permissive deadlock control of AMS by using Petri nets," *Acm Trans. Embedded Comput. Syst.*, vol. 12, no. 1, pp. 1–23, Jan. 2013.

[36] K. Xing, M. Zhou, H. Liu, and F. Tian, "Optimal Petri-net-based polynomial-complexity deadlock-avoidance policies for automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 1, pp. 188–199, Jan. 2009.
[37] Y. Chen, Z. Li, A. Al-Ahmari, N. Wu, and T. Qu, "Deadlock recovery for flexible manufacturing systems modeled with Petri nets," *Inf. Sci.*, vol. 381, pp. 290–303, Mar. 2017.
[38] S. Wang, C. Wang, M. Zhou, and Z. Li, "A method to compute strict minimal siphons in a class of Petri nets based on loop resource subsets," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 42, no. 1, pp. 226–237, Jan. 2012.
[39] D. You, S. Wang, W. Dai, W. Wu, and Y. Jia, "An approach for enumerating minimal siphons in a subclass of Petri nets," *IEEE Access*, vol. 6, pp. 4255–4265, 2017.
[40] M. Gan, S. Wang, Z. Ding, M. Zhou, and W. Wu, "An improved mixed-integer programming method to compute emptiable minimal siphons in S^3PR nets," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 6, pp. 2135–2140, Nov. 2018.
[41] S. Wang, D. You, and M. Zhou, "A necessary and sufficient condition for a resource subset to generate a strict minimal siphon in S^4PR," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 4173–4179, Aug. 2017.
[42] S. Wang, C. Wang, and M. Zhou, "Design of optimal monitor-based supervisors for a class of Petri nets with uncontrollable transitions," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 5, pp. 1248–1255, Sep. 2013.
[43] D. You, S. Wang, and C. Seatzu, "Supervisory control of a class of Petri nets with unobservable and uncontrollable transitionsc," *Inf. Sci.*, to be published. doi: 10.1016/j.ins.2018.10.018.
[44] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.
[45] Z. Li, M. Zhou, and N. Wu, "A survey and comparison of Petri net-based deadlock prevention policies for flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 173–188, Mar. 2008.



XIN GUO received the B.S. degree in measuring and control technology and instrumentations from Hangzhou Dianzi University, Hangzhou, China, in 2017. She is currently pursuing the master's degree with the School of Information and Electronic Engineering, Zhejiang Gongshang University. Her research interests include Petri net theory and application and supervisory control of discrete event systems.

SHOUGUANG WANG (M'10–SM'12) received the B.S. degree in computer science from the Changsha University of Science and Technology, Changsha, China, in 2000, and the Ph.D. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 2005.

In 2005, he joined Zhejiang Gongshang University, where he is currently a Professor with the School of Information and Electronic Engineering, the Director of the Discrete-Event Systems Group, and the Dean of the System modeling and Control Research Institute, Zhejiang Gongshang University. He was a Visiting Professor with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA, from 2011 to 2012. He was a Visiting Professor with the Electrical and Electronic Engineering Department, University of Cagliari, Cagliari, Italy, from 2014 to 2015. He was the Dean of the Department of Measuring and Control Technology and Instrument, from 2011 to 2014. He is also an Associate Editor of IEEE Access and the IEEE/CAA Journal of Automatica Sinica.

DAN YOU received the B.S. and M.S. degrees from the School of Information and Electronic Engineering, Zhejiang Gongshang University, China, in 2014 and 2017, respectively.

She is the author or the coauthor of papers published in the IEEE Transactions on Automatic Control, *Information Sciences*, and the IEEE Transactions on Systems, Man, and Cybernetics: Systems, and at the IEEE Conference on Decision and Control, the IEEE International Conference on Robotics and Automation, and others. Her research interests include supervisory control of discrete event systems, fault prediction, and deadlock control and siphon computation in Petri nets.

ZHIFU LI received the B.S. degree from the School of Information and Electronic Engineering, Zhejiang Gongshang University, China, in 2005, and the M.I.M. degree from the School of Business Administration, Zhejiang Gongshang University, in 2012. In 2005, he joined Zhejiang Gongshang University, where he is currently a Senior Economist with the School of Information and Electronic Engineering and a member of the Discrete-Event System Group, School of Information and Electronic Engineering. His research interests include supervisory control of discrete event systems and Petri net theory and application.

XIAONING JIANG received the M.E. degree in electronic engineering from Hangzhou Dianzi University, Hangzhou, China, in 1993, and the Ph.D. degree in computer science and technology from Zhejiang University, Hangzhou, in 2000. He is currently an Associate Professor and a Senior Engineer with Zhejiang Gongshang University, where he is also the Vice Dean of the IoT Research Institute. He has published over 30 research papers. He holds ten invention patents. His research interests include applied information systems, network and information security, the industrial IoT, visual analytic, and fintech.