Computation of Minimal Siphons in Petri Nets Using Problem Partitioning Approaches

Dan You, *Member, IEEE*, Oussama Karoui, *Associate Member, IEEE*, and Shouguang Wang, *Senior Member, IEEE*

***Abstract*—A large amount of research has shown the vitality of siphon enumeration in the analysis and control of deadlocks in various resource-allocation systems modeled by Petri nets (PNs). In this paper, we propose an algorithm for the enumeration of minimal siphons in PN based on problem decomposition. The proposed algorithm is an improved version of the global partitioning minimal-siphon enumeration (GPMSE) proposed by Cordone *et al.* (2005) in *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, which is widely used in the literature to compute minimal siphons. The experimental results show that the proposed algorithm consumes lower computational time and memory compared with GPMSE, which becomes more evident when the size of the handled net grows.**

***Index Terms*—Petri nets (PNs), problem decomposition, resource-allocation systems, siphons.**

I. Introduction

PETRI nets (PNs) [1]–[4] are a well-recognized mathematical tool suitable for the modeling, analysis, and control of resource-allocation systems [5], [6]. The analysis of PNs may be performed in different ways such as structural analysis and reachability analysis. A lot of effort has been devoted to structural analysis due to its advantage of low computational cost. Siphons [7], which are particular sets of places in a PN, are often utilized to detect the liveness of a considered net. More specifically, the liveness of a PN is closely related to whether siphons are sufficiently marked [8]–[14]. For ordinary PNs, the occurrence of deadlocks is due to the emptiness of some siphon, which leads to some transitions disabled permanently [15]–[17].

A seminal work to prevent deadlocks in automated manufacturing systems was established by Ezpeleta *et al.* [18] for a class of ordinary PNs called systems of simple sequential processes with resources (S^3PR). In more detail, they add a monitor (i.e., a control place) for each unmarked strict minimal siphon such that all output arcs of the monitor are linked with source transitions of the net. When all siphons are controlled, the augmented net is live. Note that, in this method, complete siphon enumeration is required since a monitor is added for each siphon in the net. As we know, the number of siphons grows exponentially with respect to the net size [19], [20], the method thus suffers from high computational complexity. Indeed, deadlock resolution policies based on siphon control all require the complete or partial siphon enumeration [21], [22]. To improve the computational efficiency of such methods, it is of great importance to improve the efficiency of siphon computation.

A. Literature Review

There are a variety of methods in the literature that compute siphons. These methods can be classified into two categories in terms of application scope. One applies to special classes of PNs. For instance, methods based on parallel algorithms [23], resource circuits [24], [25], pruning-graphs [26], genetic algorithms [27], [28] and loop resource subsets [12]. The other applies to arbitrary classes of PNs such as linear integer programming methods [16], [29], integrated net analyzer (INA)-based methods [30], methods based on semi-tensor product of matrices [31] and problem decomposition [32]. Methods in the first category are basically much more efficient in computation than those in the second category, whereas methods in the second category have the advantage of having no restriction on the class of PN.

This work focuses on siphon computation methods that are applicable to arbitrary classes of PNs. To our best knowledge, among all the methods in the literature applicable to arbitrary classes of PNs, the methods proposed by Cordone *et al.* [32] are the most efficient ones. Their methods are based on the idea of problem decomposition. To put it simply, the idea is that a problem is decomposed into multiple sub-problems so that the solution of the original problem can be deduced by combining the solutions of all the sub-problems. Cordone *et al.* [32] developed two siphon enumeration approaches based on the technique of problem decomposition, namely, global partitioning minimal-siphon enumeration (GPMSE) and local partitioning minimal-siphon enumeration (LPMSE). They have been used extensively in the literature due to their low computational complexity. The main difference between them is that GPMSE allows a current problem to be decomposed using a siphon that is already found before,

Manuscript received April 16, 2021; revised May 28, 2021; accepted June 22, 2021. This is an extended version of our previous paper that was presented in *IEEE Conference on Decision and Control*. Compared with the conference paper, the main differences of this paper lie in: Formal proofs of all the theoretical results, complexity analysis of the proposed method, and more detailed explanations of the proposed method. This work was supported in part by the Zhejiang Natural Science Foundation (LQ20F020009), the Zhejiang Provincial Key Laboratory of New Network Standards and Technologies (2013E10012), and the Public Technology Research Plan of Zhejiang Province (LGJ21F030001). Recommended by Associate Editor Jun Zhang. *(Corresponding author: Oussama Karoui.)*

Citation: D. You, O. Karoui, and S. G. Wang, "Computation of minimal siphons in Petri nets using problem partitioning approaches," *IEEE/CAA J. Autom. Sinica*, vol. 9, no. 2, pp. 329–338, Feb. 2022.

The authors are all with the School of Information and Electronic Engineering (Sussex Artificial Intelligence Institute), Zhejiang Gongshang University, Hangzhou 310018, China (e-mail: youdan000@hotmail.com; rtkaroui@gmail.com; wsg5000@hotmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2021.1004326

whereas LPMSE requires getting a solution (i.e., a siphon) to the current problem first and then using the siphon to decompose the current problem. Although GPMSE and LPMSE have good performance, they still have some drawbacks. In more detail, GPMSE can hardly be applied to nets with large size, whereas LPMSE might generate non-minimal siphons.

B. Contributions of This Work

In this study, we propose an improved version of GPMSE that we name improved GPMSE (IGPMSE). IGPMSE enumerates minimal siphons in a PN with less consumption on CPU time and memory space than the original version so that it is readily applicable to large-size nets. Such improvements are realized mainly based on the following ideas:

1) We expand as large as possible a set of places that are forced to be contained in a minimal siphon to be found;

2) We deduce more conditions under which there is no need to further partition a problem, i.e., more conditions to terminate the partition of a problem; and

3) We adopt a depth-first search for the enumeration of minimal siphons rather than the width-first search performed in [32].

We notice that the first two ideas aim to reduce the number of sub-problems generated during the procedure of siphon enumeration, while the third idea aims to reduce memory consumption. Indeed, it is validated by experimental results that IGPMSE behaves better than GPMSE in both computational time and memory consumption.

C. Organization of This Paper

The rest of this paper is organized as follows. Section II recalls basic notions of PNs. Section III introduces siphons and their related properties which are involved in the paper. In Section IV, we first present some preliminary functions and results, then introduce the algorithm of IGPMSE, and finally provide an illustrative example. Section V presents the comparison between IGPMSE and GPMSE. We conclude this paper in Section VI along with our further research work.

II. Preliminaries of Petri Nets

In this section, we recall the basics of Petri nets [33] and siphons.

A Petri net (PN) Σ is a three-tuple (P, T, F), where

- P is the set of places;
- T is the set of transitions; and
- $F \subseteq (P\times T)\cup(T\times P)$ represents directed arcs of the net.

The preset and post-set of a node $z \in P\cup T$ are denoted by ${}^\bullet z$ and $z^\bullet$, defined as

${}^\bullet z = \{z' \in P\cup T \,|(z', z) \in F\}$ and $z^\bullet = \{z' \in P\cup T \,|(z, z') \in F\}$, respectively. Moreover, it is defined that ${}^{\bullet\bullet}z = {}^\bullet({}^\bullet z)$ and $z^{\bullet\bullet} = (z^\bullet)^\bullet$. Furthermore, given a set of nodes $Z \subseteq P\cup T$, it is defined that ${}^\bullet Z = \cup_{z\in Z}{}^\bullet z$, and $Z^\bullet = \cup_{z\in Z} z^\bullet$.

A transition $t \in T$ is said to be a sink transition if $t^\bullet = \emptyset$ and a source transition if ${}^\bullet t = \emptyset$. Similarly, a place $p \in P$ is said to be a sink place if $p^\bullet = \emptyset$ or a source place if ${}^\bullet p = \emptyset$.

Given a PN $\Sigma = (P, T, F)$, a subset of places $P' \subseteq P$ and a subset of transitions $T' \subseteq T$, the subnet of Σ generated by P' and T' is $\Sigma' = (P', T', F')$, where $F' = [(P'\times T')\cup(T'\times P')]\cap F$.

III. Siphons and Their Properties

A non-empty set $S \subseteq P$ is called a siphon if ${}^\bullet S \subseteq S^\bullet$. A siphon is said to be minimal if it does not contain any other siphon.

In the following, we report two properties related to siphons that are the known results in the literature.

Property 1 [32]: Let $\Sigma = (P, T, F)$ be a PN and $\{p\} \subseteq P$ such that ${}^\bullet p = \emptyset$. Then, $\{p\}$ is a minimal siphon.

Property 1 indicates that every source place constitutes a set that is a minimal siphon.

Property 2 [32]: Let $\Sigma = (P, T, F)$ be a PN. The set P is a siphon if ${}^\bullet t \neq \emptyset,\ \forall t \in T$.

By Property 2, the set of all places in a PN is a siphon if the net contains no source transition.

IV. Computation of Minimal Siphons in PN

In this section, we propose a method of minimal-siphon computation in a PN based on problem decomposition, namely, improved GPMSE (IGPMSE). To this end, we first formally define a *problem* that is discussed throughout the paper.

Definition 1: Let $\Sigma = (P, T, F)$ be a PN and $P_\Omega \subseteq P$. We define $\Delta = (\Sigma, P_\Omega)$ a *problem* that finds the set of all minimal siphons containing P_Ω in the net Σ. We use Π_Δ to denote the set of all solutions to the problem $\Delta = (\Sigma, P_\Omega)$, i.e., Π_Δ denotes the set of all minimal siphons containing P_Ω in the net Σ.

We notice that, in the case that $P_\Omega = \emptyset$, $\Delta = (\Sigma, P_\Omega)$ is a problem that finds the set of all minimal siphons in the net Σ, which is exactly what we aim to solve in this paper. We will see later how this problem is solved by decomposing itself into several sub-problems whose solutions together constitute the solution space of the original problem.

In the remainder of this section, we first provide several functions that will be called in our proposed method, i.e., IGPMSE, and prove some results related to these functions, and then we introduce IGPMSE in detail.

A. Preliminary Functions and Results

We first introduce a function named *HandleSourcePlace*. We can see that this function returns a set that stores every set consisting of a single source place in the given net and it obtains a reduced net by eliminating all source places from the net. We have the following result regarding the function. It indicates that the set of minimal siphons in the original net consists of minimal siphons in the reduced net and all sets consisting of a source place.

Function 1 $(\Sigma', \Theta) :=$ *HandleSourcePlace* (Σ)

Input: PN $\Sigma = (P, T, F)$;

Output: PN $\Sigma' = (P', T', F')$ and a set Θ.

1: $\Theta := \emptyset$;

2: **for** $(p \in P$ s.t. ${}^\bullet p = \emptyset)\{$

3: $\quad \Theta := \Theta \cup \{\{p\}\}$;

4: }
5: obtain the net $\Sigma' = (P', T', F')$ by removing all source places and their related arcs from Σ;
6: **output:** Σ' and Θ.

Result 1: Given a PN $\Sigma = (P, T, F)$ and $(\Sigma', \Theta) = HandleSourcePlace(\Sigma)$, it holds that $\Pi_{\Delta 1} = \Pi_{\Delta 2} \cup \Theta$, where $\Delta_1 = (\Sigma, \emptyset)$ and $\Delta_2 = (\Sigma', \emptyset)$.

Proof: It is clear that $\Theta = \{\{p\} | {}^\bullet p = \emptyset\}$. By Property 1, $\Pi_{\Delta 1} \supseteq \Theta$ holds. We can see that Function *HandleSourcePlaces* removes all the source places from Σ but preserves all the transitions connected to places. Thus, any minimal siphon S' in Σ' is a minimal siphon in Σ and any minimal siphon S in Σ is either a set consisting of a single source place, i.e., $S \in \Theta$ or a minimal siphon in Σ'. Thus, the result follows. ■

The following function named *ReduceNetSize* can be used to simplify the net that we consider. In simple words, it first repeatedly deletes source transitions and their output places in the considered net and then repeatedly deletes sink places/transitions. Result 2 indicates that such a simplification procedure performed by *ReduceNetSize* has no impact on the minimal-siphon enumeration result.

Function 2 $\Sigma' = ReduceNetSize\ (\Sigma)$

Input: $\Sigma = (P, T, F)$;
Output: $\Sigma' = (P', T', F')$.
1: **While** ($t \in T$ is a source transition in Σ){
2: obtain the subnet $\Sigma' = (P', T', F')$ of Σ generated by $T' := T \setminus \{t\}$ and $P' := P \setminus t^\bullet$;
3: $\Sigma := \Sigma'$;
4: }
5: **while** ($z \in P \cup T$ is a sink place/transition in Σ) {
6: **if** (z is a place) {
7: obtain the subnet $\Sigma' = (P', T', F')$ of Σ generated by $P' := P \setminus \{z\}$ and $T' := T$;
8: }
9: **else** {
10: obtain the subnet $\Sigma' = (P', T', F')$ of Σ generated by $P' := P$ and $T' := T \setminus \{z\}$;
11: }
12: $\Sigma := \Sigma'$;
13: }
14: denote the resultant net as Σ';
15: **output:** Σ'.

Result 2: Given two PNs Σ and Σ', a set of places P_Ω, and two problems $\Delta_1 = (\Sigma, P_\Omega)$ and $\Delta_2 = (\Sigma', P_\Omega)$, it holds that $\Pi_{\Delta 1} = \Pi_{\Delta 2}$ if $\Sigma' = ReduceNetSize\ (\Sigma)$.

Proof: Places connected to source transitions cannot belong to any siphon because no other places can have these transitions in their postset. Thus, we may repeatedly delete source transitions and their output places and the minimal-siphon enumeration results are the same in the original net and the reduced net. Now, we consider sink transitions. It is trivial to see that whether sink transitions exist will not change the siphon enumeration result. Thus, they can be removed. Finally, we consider sink places. Let p be a sink place. Since $p^\bullet = \emptyset$ and ${}^\bullet p \neq \emptyset$, $\{p\}$ cannot be a siphon. Moreover, let S be a siphon including the sink place p. It holds that ${}^\bullet(S \setminus \{p\}) \subseteq {}^\bullet S$ and $(S \setminus \{p\})^\bullet = S^\bullet$. Since S is a siphon, it is ${}^\bullet S \subseteq S^\bullet$. It thus follows that ${}^\bullet(S \setminus \{p\}) \subseteq (S \setminus \{p\})^\bullet$. It means that S is not a minimal siphon. In other words, any siphon including a sink place cannot be minimal. Thus, we may delete sink places whose removal will not change the minimal-siphon enumeration result. Consequently, the result follows. ■

We infer that functions *HandleSourcePlace* and *ReduceNetSize* can reduce significantly the size of a net and therefore facilitate the enumeration process of minimal siphons.

The next function, named *Expand*, is used to expand the set P_Ω of places that must be included in a minimal siphon. In other words, due to some properties of minimal siphons, we may know that, in the case that some places are required to be included in a minimal siphon, some other places are definitely included. Specifically, as performed in *Expand*, when the set P_Ω is given, there are two cases to expand P_Ω: 1) If there exists an input but not output transition of the set P_Ω such that it has only one input place p', the place p' can be included to expand P_Ω; and 2) If there exists a place p in P_Ω and another place p' out of P_Ω such that p' is the unique output place of p, the place p' can be included to expand P_Ω. Result 3 indicates that such an expanding procedure on P_Ω does not change the minimal-siphon enumeration result. It is worth noting that, typically, the bigger the set P_Ω is, the faster a solution to the corresponding problem may be derived. Thus, the function *Expand* may be used later to fasten the process of searching minimal siphons.

Function 3 $P_\Omega' = Expand(P_\Omega, \Sigma)$

Input: PN $\Sigma = (P, T, F)$ without source places and a set P_Ω of places;
Output: a set P_Ω' of places.
1: $P_\Omega' := P_\Omega$;
2: **while** ($\exists t \in {}^\bullet P_\Omega' \setminus P_\Omega'^\bullet$ s.t. ${}^\bullet t = \{p'\} \vee$
$\exists p \in P_\Omega', \exists p' \in P \setminus P_\Omega'$ s.t. $p'^\bullet = \{p'\}$){
3: $P_\Omega' := P_\Omega' \cup \{p'\}$;
4: }
5: **output**: P_Ω'.

Result 3: Given a PN Σ without source places, two sets of places P_Ω and P_Ω', and two problems $\Delta_1 = (\Sigma, P_\Omega)$ and $\Delta_2 = (\Sigma, P_\Omega')$, it holds that $\Pi_{\Delta 1} = \Pi_{\Delta 2}$ if $P_\Omega' = Expand(P_\Omega, \Sigma)$.

Proof: Since $P_\Omega' \supseteq P_\Omega$, a minimal siphon containing P_Ω' is definitely a minimal siphon containing P_Ω, i.e., $\Pi_{\Delta 1} \supseteq \Pi_{\Delta 2}$ holds. Next, we prove a minimal siphon containing P_Ω is also a minimal siphon containing P_Ω', i.e., $\Pi_{\Delta 1} \subseteq \Pi_{\Delta 2}$. Consider that $t \in {}^\bullet P_\Omega \setminus P_\Omega^\bullet$ such that ${}^\bullet t = \{p'\}$. Let S be a minimal siphon containing P_Ω. We can see that $p' \in S$ since otherwise $t \in {}^\bullet S$ but $t \notin S^\bullet$, which contradicts the fact that S is a siphon. Thus, the set P_Ω can be iteratively expanded by including the place p'. Now, consider that $p \in P_\Omega$, and $p' \in P \setminus P_\Omega$ such that $p'^\bullet = \{p'\}$. Let S be a minimal siphon containing P_Ω. We prove that $p' \in S$. By contradiction, suppose that $p' \notin S$. Let $T' = {}^\bullet p' \wedge p^\bullet$. It holds that $T' \not\subset {}^\bullet S$ and $T' \subset S^\bullet$. Let $P_X = {}^\bullet T' \cap S$. We have $(S \setminus P_X)^\bullet = S^\bullet \setminus T'$. Moreover, it is clear that ${}^\bullet(S \setminus P_X) \subseteq {}^\bullet S$. Since

$T' \not\subset {}^\bullet S$, it follows that ${}^\bullet(S\backslash P_X) \subseteq {}^\bullet S\backslash T'$. Consequently, we have ${}^\bullet(S\backslash P_X) \subseteq (S\backslash P_X)^\bullet$. It means that $S\backslash P_X$ is a siphon, which contradicts the fact that S is a minimal siphon. Therefore, it holds that $p' \in S$. Thus, the set P_Ω can be iteratively expanded by including the place p'. By the above analysis, $\Pi_{\Delta 1} \subseteq \Pi_{\Delta 2}$ holds. Consequently, the result follows. ■

In what follows, we present function *PomegaMiniSiphon* that was exhibited in the work of Cordone *et al.* [32]. Given a PN without source transitions and a set P_Ω of places, *PomegaMiniSiphon* returns a P_Ω-minimal siphon in the net. We notice that the notion of a P_Ω-minimal siphon is not the same as the notion of a minimal siphon containing P_Ω. The formal definition of a P_Ω-minimal siphon is as follows:

Given a siphon S and a set of places $P_\Omega \subseteq P$, S is called P_Ω-minimal if S contains all the places in P_Ω and S does not contain any other siphon that contains all the places in P_Ω.

We can see that a P_Ω-minimal siphon is not necessarily a minimal siphon except the case that $P_\Omega = \emptyset$.

Function 4 S = *PomegaMiniSiphon* (Σ, P_Ω)

Input: PN $\Sigma = (P, T, F)$ without source transitions and a set of places $P_\Omega \subseteq P$;

Output: P_Ω-minimal siphon S.

```
While (P_Ω ≠ P) {
    Σ# := Σ;
    select a place p from P\P_Ω;
    Σ := DeletePlace(Σ, p);
    /*DeletePlace is a function returning a net after deleting
    a given place and its related arcs from a given net.*/
      while (t ∈ T is a source transition in Σ) {
          obtain the subnet Σ' = (P', T', F') of Σ generated by
          T' := T \{t} and P' := P \t•;
          Σ := Σ';
          }
      if (P_Ω ⊄ P){
          P_Ω := P_Ω ∪ {p};
          Σ := Σ#;
          }
}
output: S := P_Ω.
```

The computation of *PomegaMiniSiphon* is detailed as follows. If the given P_Ω is the whole set of places of the considered net, P_Ω itself is a P_Ω-minimal siphon; and otherwise, we find a P_Ω-minimal siphon by the following operations:

We delete each place $p \in P\backslash P_\Omega$ from the considered net and then repeatedly delete resultant source transitions and their output places. If any place in P_Ω is deleted due to the above operations, we restore the net to its original structure before deleting p, expand P_Ω by including p, and then go to the next iteration to delete the next place in $P\backslash P_\Omega$; and otherwise, we directly go to the next iteration to delete the next place in $P\backslash P_\Omega$. We repeat the above operations until P_Ω is the whole set of places of the resultant net and P_Ω now is a solution.

Result 4: Given a PN $\Sigma = (P, T, F)$ without source transitions and a set of places $P_\Omega \subseteq P$, S = *PomegaMiniSiphon* (Σ, P_Ω) is a P_Ω-minimal siphon.

Proof: By Function *PomegaMiniSiphon*, we can see that the subnet of Σ generated by S and ${}^\bullet S \cup S^\bullet$ contains no source transition. Thus, S is a siphon due to Property 2.

By contradiction, suppose that S is not a P_Ω-minimal siphon. Let $S' \subset S$ be a siphon containing P_Ω and $p \in S\backslash S'$. Since $p \in S$, according to Function *PomegaMiniSiphon*, there is a path from p to a place $p' \in P_\Omega$ such that every transition in the path has only one input place. Since $p' \in P_\Omega$, it is clear that $p' \in S'$. It thus follows that $p \in S'$ since otherwise S' cannot be a siphon. This, however, contradicts the fact that $p \notin S'$. Consequently, we may conclude that S is a P_Ω-minimal siphon. ■

We finally present a function *SiphonIsMini*, which checks if a given siphon containing two or more places is minimal. Specifically, we first obtain the subnet Σ_s generated by S and ${}^\bullet S \cap S^\bullet$ and then for each place in S, the following procedure is performed: we delete the place and its related arcs from the considered net and then *ReduceNetSize* is called to handle the resultant net. In the case that *ReduceNetSize* returns a non-empty net, we may conclude that the siphon S is not minimal. i.e., *ans* = Flase; otherwise, after checking all the places in S, we may conclude that the siphon S is minimal. i.e., *ans* = True.

Function 5 *ans* = *SiphonIsMini*(S)

Input: a siphon S with $|S| \geq 2$;

Output: *ans* $\in$ {True, False}. /* *ans* = True means that the siphon S is minimal and *ans* = False means not.*/

```
1:  ans := True;
2:  obtain the subnet Σ_s generated by S and •S ∩ S•;
3:  for (p ∈ S) {
4:      Σ' := DeletePlace(Σ_s, p);
5:      Σ' := ReduceNetSize(Σ');
6:      if (Σ' ≠ ∅){
7:          ans := False;
8:          go to Step 11;
9:      }
10: }
11: output: ans.
```

Result 5: Given a siphon S with $|S| \geq 2$, S is minimal *iff* *SiphonIsMini*(S) = True.

Proof: (=>) By contradiction, suppose that S is not minimal. In other words, there is a siphon $S' \subset S$. Let $p \in S\backslash S'$. Let Σ' = *DeletePlace*(Σ_s, p) and Σ'' = *ReduceNetSize*(Σ'). We can see that S' is a siphon in Σ'. Since *SiphonIsMini*(S) = True, it is $\Sigma'' = \emptyset$. By Result 2, there is no minimal siphon in Σ' since there is no minimal siphon in Σ''. Thus, there is no siphon in Σ', which, however, contradicts that S' is a siphon in Σ'. As a result, S is minimal.

(<=) By contradiction, suppose that *SiphonIsMini* (S) = False. It indicates that $\exists p \in S$ such that *ReduceNetSize* (Σ') $\neq \emptyset$, where Σ' = *DeletePlace* (Σ_s, p). Let $\Sigma'' = (P'', T'', F'')$ = *ReduceNetSize* (Σ'). We can see that there is no source transition in Σ''. By Property 2, P'' is a siphon. Since $P'' \subseteq S$, the siphon S is not minimal, which contradicts the fact that S is minimal. As a result, it is *SiphonIsMini* (S) = True. ■

B. Minimal Siphon Computation Using IGPMSE

Algorithm 1 *IGPMSE*

Input: PN $\Sigma = (P, T, F)$.
Output: the set Θ of all minimal siphons.
1: $(\Sigma, \Theta) :=$ *HandleSourcePlace*(Σ);
2: $\Sigma :=$ *ReduceNetSize*(Σ);
3: **if** $(\Sigma \neq \emptyset)$ {
4: create the root node $(\Sigma, \emptyset)$ of a tree;
5: $S :=$ *PomegaMiniSiphon* $(\Sigma, \emptyset)$;
6: $\Theta := \Theta \cup \{S\}$;
7: *index* := 1;
8: Γ[*index*] := S; /* Γ is a linked list saving minimal siphons that are used for decomposing problems*/
9: Γ[*index*+1] := $\emptyset$;
10: $\Theta :=$ *CreateNewNode*$(\Sigma, \emptyset,$ *index*, $\Theta)$;
}
11: **output:** Θ.

Function 6 $\Theta =$ *CreateNewNode* $(\Sigma, P_\Omega,$ *index*, $\Theta)$

Input: PN $\Sigma = (P, T, F)$, a set P_Ω of places, *index* $\in \{1, 2, \ldots\}$, and a set Θ of minimal siphons;
Output: Updated set Θ of minimal siphons.
1: $S := \Gamma$[*index*]; /* a minimal siphon for decomposing*/
2: **if** $(S = \emptyset)$ {
3: $S :=$ *PomegaMiniSiphon* (Σ, P_Ω);
4: Γ[*index*] := S;
5: Γ[*index*+1] := $\emptyset$;
6: $\Theta := \Theta \cup \{S\}$;
7: }
8: $P_\Omega' := P_\Omega$;
9: **for** $(p \in S \setminus P_\Omega)$ {
10: $\Sigma' :=$ *DeletePlace*(Σ, p);
11: $\Sigma' :=$ *ReduceNetSize*(Σ');
12: **if** $(\Sigma' \neq \emptyset)$ **{**
13: $P_\Omega'' :=$ *Expand*(P_Ω', Σ');
14: create a node (Σ', P_Ω'');
15: add an arc labeled by "p" from node (Σ, P_Ω) to node (Σ', P_Ω'');
16: **if** $(P_\Omega'' \subseteq P')$ **{**
17: **if** (P_Ω'' is a siphon) **{**
18: **if** (*SiphonIsMini*(P_Ω'')) **{**
19: $\Theta := \Theta \cup \{ P_\Omega''\}$;
20: **}**
21: **}**
22: **else**
23: $\Theta :=$ *CreateNewNode*$(\Sigma', P_\Omega'',$ *index* + 1, $\Theta)$;
24: **}**
25: **}**
26: $P_\Omega' := P_\Omega' \cup \{p\}$;
27: **}**
28: **Output:** Θ.

In this section, we propose an improved algorithm of GPMSE, named IGPMSE. The improvements mainly lie in: 1) We expand the set of places required to be included in a minimal siphon to be found; 2) We add a condition that terminates the further decomposition on a problem, i.e., P_Ω is a siphon; and 3) The depth-first search is adopted in IGPMSE whereas GPMSE uses the width-first search. The first two improvements may result in a significant decrease in the number of sub-problems to be solved and the third improvement may save the memory consumption.

We explain Algorithm 1 (IGPMSE) as follows. The set Θ is used to save all the found minimal siphons.

First, functions *HandleSourcePlace* and *ReduceNetSize* are called to find all minimal siphons that consist of a single source place and remove places and transitions that are not needed for the further search of minimal siphons. Next, we create the root node $(\Sigma, \emptyset)$ of a tree and use function *PomegaMiniSiphon* to find a minimal siphon. Note that, we establish a linked list Γ to store minimal siphons that are used for decomposing problems. By calling *CreateNewNode*$(\Sigma, \emptyset,$ *index*, $\Theta)$, we decompose the problem $(\Sigma, \emptyset)$ using the minimal siphon stored in Γ[*index*]. In the following, we explain the function *CreateNewNode*$(\Sigma, P_\Omega,$ *index*, $\Theta)$ in detail.

CreateNewNode$(\Sigma, P_\Omega,$ *index*, $\Theta)$ is a recursive function. It iteratively decomposes problems using the depth-first search. The variable *index* indicates that we use the minimal siphon in Γ[*index*] to decompose the problem (Σ, P_Ω). Note that, as shown in Steps 1–7, if Γ[*index*] is empty, we need to compute a minimal siphon and store it in Γ[*index*]. Now, we start the decomposition of the problem (Σ, P_Ω) using the minimal siphon S in Γ[*index*]. We delete places in $S \backslash P_\Omega$ one after another to generate sub-problems. In order to get a sub-problem easy to be solved, after deleting a place p from the net Σ, we further reduce its net size and expand the set P_Ω to a new set P_Ω'', by calling functions *ReduceNetSize* and *Expand*, respectively. In the case that the resultant net Σ' is not empty, we create a new node (Σ', P_Ω'') in the tree. Regarding the new node, there are two cases to terminate the further decomposition on it: 1) $P_\Omega'' \not\subset P'$; and 2) P_Ω'' is a siphon. In any other cases, we continue the decomposition of the problem (Σ', P_Ω'') by calling *CreateNewNode*$(\Sigma', P_\Omega'',$ *index*+1, $\Theta)$. Note that, in the case that we terminate the decomposition on (Σ', P_Ω''), if P_Ω'' is a siphon, we determine if it is minimal. If so, we should include P_Ω'' in the set Θ. Besides, when we consider generating the next sub-problem of (Σ, P_Ω) by deleting another place in $S \backslash P_\Omega$, the currently deleted place p should be included in the set P_Ω considered in that sub-problem (see Step 26). By repeating the above procedure, we finish the construction of the tree when no node needs to be further decomposed and the final set Θ is the set of all minimal siphons in the handled PN.

We finally notice that problems in the same level of the tree are decomposed using the same minimal siphon. In particular, we use the minimal siphon in $\Gamma[i]$ to decompose all the problems in level i.

In what follows, we prove Algorithm 1 (IGPMSE) computes the set of all minimal siphons in a PN. To this end, we need the following two lemmas.

Lemma 1: Let $\Sigma = (P, T, F)$ be a PN, $P_\Omega \subseteq P$ be a set of places, $\Delta = (\Sigma, P_\Omega)$ be the problem of finding the set Π_Δ of all minimal siphons containing P_Ω of Σ and $S = \{p_1, p_2, \ldots, p_n\}$

be a set of places. We have

$$\Pi_\Delta = \Pi_{\Delta 1} \cup \Pi_{\Delta 2} \cup \cdots \cup \Pi_{\Delta n} \cup \Pi_{\Delta n+1}$$

where

- $\Delta_1 = (\Sigma_1, P_\Omega)$ with $\Sigma_1 = DeletePlace(\Sigma, p_1)$;
- $\Delta_2 = (\Sigma_2, P_\Omega \cup \{p_1\})$ with $\Sigma_2 = DeletePlace(\Sigma, p_2)$;

…

- $\Delta_n = (\Sigma_n, P_\Omega \cup \{p_1, p_2, \ldots, p_{n-1}\})$ with $\Sigma_n = DeletePlace(\Sigma, p_n)$;
- $\Delta_{n+1} = (\Sigma, P_\Omega \cup S)$.

Proof: The problem of finding the set Π_Δ of all minimal siphons containing P_Ω of Σ can be decomposed into $n + 1$ sub-problems, that is

1) Finding the set of all minimal siphons of Σ excluding p_1 but containing P_Ω;

2) Finding the set of all minimal siphons of Σ excluding p_2 but containing $P_\Omega \cup \{p_1\}$;

…

n) Finding the set of all minimal siphons of Σ excluding p_n but containing $P_\Omega \cup \{p_1, p_2, \ldots, p_{n-1}\}$; and

n+1) Finding the set of all minimal siphons of Σ containing $P_\Omega \cup \{p_1, p_2, \ldots, p_n\} = P_\Omega \cup S$.

The solutions to the above $n + 1$ sub-problems constitute the solution to the original problem. Consequently, the lemma holds. ■

Lemma 2: Let $\Sigma = (P, T, F)$ be a PN, P_Ω be a set of places, and $\Delta = (\Sigma, P_\Omega)$ be the problem of finding the set Π_Δ of all minimal siphons containing P_Ω of Σ. It holds that

1) $\Sigma = \emptyset \vee P_\Omega \not\subset P \vee P_\Omega$ is a siphon of Σ but not minimal $\Rightarrow \Pi_\Delta = \emptyset$;

2) $\Sigma \neq \emptyset \wedge P_\Omega \subseteq P \wedge P_\Omega$ is a minimal siphon of $\Sigma \Rightarrow \Pi_\Delta = \{P_\Omega\}$.

Proof: Straightforward from the definition of minimal siphons. ■

Now, we are ready to prove the following result.

Theorem 1: Given a PN Σ as the input, Algorithm 1 (IGPMSE) outputs the set of all minimal siphons in Σ.

Proof: We prove that the set Θ output by Algorithm 1 is the solution Π_Δ to the problem $\Delta = (\Sigma, \emptyset)$, i.e., the set of all minimal siphons in Σ.

First, we compute $(\Sigma, \Theta) = HandleSourcePlace(\Sigma)$ and $\Sigma = ReduceNetSize(\Sigma)$. Let Σ' be the updated net. By Results 1 and 2, it is

$$\Pi_\Delta = \Pi_{\Delta'} \cup \Theta \tag{1}$$

where $\Delta = (\Sigma, \emptyset)$ and $\Delta' = (\Sigma', \emptyset)$. Then, the root node of a tree is created, that is, $\Delta' = (\Sigma', \emptyset)$ and the problem decomposition is performed on Δ':

First, a minimal siphon S is computed, which is used for decomposing the problem Δ'. Suppose that $S = \{p_1, p_2, \ldots, p_n\}$. n son-nodes (i.e., sub-problems) of $\Delta' = (\Sigma', \emptyset)$ are created, namely,

- $\Delta'_1 = (\Sigma'_1, P_{\Omega 1})$ with $\Sigma'_1 = ReduceNetSize(DeletePlace(\Sigma', p_1))$ and $P_{\Omega 1} = Expand(\emptyset, \Sigma'_1)$;
- $\Delta'_2 = (\Sigma'_2, P_{\Omega 2})$ with $\Sigma'_2 = ReduceNetSize(DeletePlace(\Sigma', p_2))$ and $P_{\Omega 2} = Expand(\{p_1\}, \Sigma'_2)$;

…; and

- $\Delta'_n = (\Sigma'_n, P_{\Omega(n)})$ with $\Sigma'_n = ReduceNetSize(DeletePlace(\Sigma', p_n))$ and $P_{\Omega(n)} = Expand(\{p_1, p_2, \ldots, p_{(n-1)}\}, \Sigma'_n)$.

Due to Lemma 1 and Results 2–3, we have

$$\Pi_{\Delta'} = \Pi_{\Delta' 1} \cup \Pi_{\Delta' 2} \cup \cdots \cup \Pi_{\Delta' n} \cup \Pi_{\Delta' n+1}$$

where $\Delta'_{n+1} = (\Sigma', S)$. Since S is a minimal siphon of Σ', it is $\Pi_{\Delta' n+1} = \{S\}$ by Lemma 2. Thus,

$$\Pi_{\Delta'} = \Pi_{\Delta' 1} \cup \Pi_{\Delta' 2} \cup \cdots \cup \Pi_{\Delta' n} \cup \{S\}. \tag{2}$$

By (1) and (2), it follows that

$$\Pi_\Delta = \Pi_{\Delta' 1} \cup \Pi_{\Delta' 2} \cup \cdots \cup \Pi_{\Delta' n} \cup \{S\} \cup \Theta.$$

Note that Θ is updated as $\Theta \cup \{S\}$ by Step 6. Thus, we have

$$\Pi_\Delta = \Pi_{\Delta' 1} \cup \Pi_{\Delta' 2} \cup \cdots \cup \Pi_{\Delta' n} \cup \Theta. \tag{3}$$

Consider a sub-problem $\Delta'_i = (\Sigma'_i, P_{\Omega(i)})$, $i \in \{1, 2, \ldots, n\}$. It can be considered in the following three cases:

1) $\Sigma'_i = \emptyset \vee P_{\Omega(i)} \not\subset P'_i \vee P_{\Omega(i)}$ is a siphon of Σ'_i but not minimal;

2) $\Sigma'_i \neq \emptyset \wedge P_{\Omega(i)} \subseteq P'_i \wedge P_{\Omega(i)}$ is a minimal siphon of Σ'_i;

3) $\Sigma'_i \neq \emptyset \wedge P_{\Omega(i)} \subseteq P'_i \wedge P_{\Omega(i)}$ is not a siphon of Σ'_i.

Without loss of generality, we assume that, in the sub-problems Δ'_1–Δ'_n, there are k sub-problems in Case 1, denoted as $\Delta'_{11}, \Delta'_{12}, \ldots, \Delta'_{1k}$, there are l sub-problems in Case 2, denoted as $\Delta'_{21}, \Delta'_{22}, \ldots, \Delta'_{2l}$, and there are m sub-problems in Case 3, denoted as $\Delta'_{31}, \Delta'_{32}, \ldots, \Delta'_{3m}$. Clearly, it is $k + l + m = n$. By Lemma 2 and (3), we have

$$\Pi_\Delta = \left\{P_{\Omega(21)}\right\} \cup \left\{P_{\Omega(22)}\right\} \cup \cdots \cup \left\{P_{\Omega(2l)}\right\} \cup \Pi_{\Delta'(31)} \cup \Pi_{\Delta'(32)} \cup \cdots \cup \Pi_{\Delta'(3m)} \cup \Theta.$$

Note that Θ is updated by including $P_{\Omega(i)}$ for each sub-problem Δ'_i in Case 2. Thus, it follows that

$$\Pi_\Delta = \Pi_{\Delta'(31)} \cup \Pi_{\Delta'(32)} \cup \cdots \cup \Pi_{\Delta'(3m)} \cup \Theta. \tag{4}$$

In the case that $m = 0$, i.e., none of sub-problems are in Case 3, Θ is the final output set and it obviously holds $\Pi_\Delta = \Theta$. Otherwise, problems $\Delta'_{31}, \Delta'_{32}, \ldots, \Delta'_{3m}$ are further decomposed by repeating the above similar procedure. Note that during the procedure of iterative problem decomposition, the size of the considered net is gradually reduced. It implies that there cannot be infinite sub-problems that need to be further decomposed. Consequently, after the iterative problem decomposition and the update of Θ, the final set $\Theta = \Pi_\Delta$. ■

Finally, we note that the computational complexity of the proposed IGPMSE (Algorithm 1) is actually the same as that of GPMSE. This is because the tree generated by IGPMSE can be the same as the one generated by GPMSE in the worst case. Since GPMSE is of exponential complexity with respect to the net size [32], the computational complexity of IGPMSE is also exponential with respect to the net size. Nevertheless, although they have the same complexity, IGPMSE performs much better than GPMSE in computational time and memory consumption in most cases, which will be seen in Section V.

C. Illustrative Example

In this subsection, we illustrate the proposed IGPMSE by considering the computation of all minimal siphons in the PN Σ shown in Fig. 1.

First, functions *HandleSourcePlace* and *ReduceNetSize* are called to handle the PN Σ. We can see that it contains neither

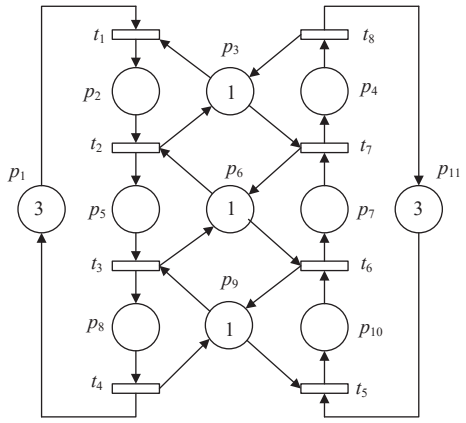


Fig. 1. PN Σ.

source nor sink nodes. Thus, we have $\Theta = \emptyset$ and $\Sigma_1 = \Sigma$ after calling these two functions. (Here we use Σ_1 to denote the resultant net after calling these two functions.) Now, we create the root node $\Delta_1 = (\Sigma_1, \emptyset)$ of a tree, as shown in Fig. 2. For the sake of simplicity, we use the set P_i of places in a net Σ_i to denote the net. For instance, the set of places $P_1 = \{p_1 - p_{11}\}$ denotes the net Σ_1. Next, we get a minimal siphon $S_1 = \{p_8, p_9, p_{10}\}$ by calling *PomegaMiniSiphon* $(\Sigma_1, \emptyset)$ and we have $\Theta = \{S_1\}$, $\Gamma[1] = S_1$ and $\Gamma[2] = \emptyset$.

Now, we start the decomposition of the problem $\Delta_1 = (\Sigma_1, \emptyset)$, which is performed by calling *CreateNewNode*$(\Sigma_1, \emptyset, 1, \{S_1\})$. Since *index* = 1, it means that we use $S_1 = \Gamma[1] = \{p_8, p_9, p_{10}\}$ to decompose the problem. The details are as follows. Note that the depth-first search is adopted to iteratively decompose problems.

First, we consider the case that place p_8 is deleted from the net Σ_1. Function *ReduceNetSize* is then performed to simplify the net. The obtained net is denoted as Σ_2 with $P_2 = \{p_2 - p_7, p_{10}, p_{11}\}$. Accordingly, a new node $\Delta_2 = (\Sigma_2, \emptyset)$ is added to the tree with an arc labeled by "p_8" from node $\Delta_1 = (\Sigma_1, \emptyset)$.

Now, we decompose the problem $\Delta_2 = (\Sigma_2, \emptyset)$ by calling *CreateNewNode*$(\Sigma_2, \emptyset, 2, \{S_1\})$. Since $\Gamma[2]$ is empty, we compute a minimal siphon $S_2 = \{p_5, p_6, p_7\}$ by calling *PomegaMiniSiphon* $(\Sigma_2, \emptyset)$ and use it to decompose the problem $\Delta_2 = (\Sigma_2, \emptyset)$. Note that it is now updated that $\Gamma[2] = S_2$, $\Gamma[3] = \emptyset$ and $\Theta = \{S_1, S_2\}$.

Similarly, we first consider the case that place p_5 is deleted from the net Σ_2, resulting in a new node $\Delta_3 = (\Sigma_3, \emptyset)$ in the tree with an arc labeled by "p_5" from node $\Delta_2 = (\Sigma_2, \emptyset)$, where Σ_3 is a net with the set of places $P_3 = \{p_2 - p_4, p_7, p_{10}, p_{11}\}$.

Again, we decompose the problem $\Delta_3 = (\Sigma_3, \emptyset)$ by calling *CreateNewNode*$(\Sigma_3, \emptyset, 3, \{S_1, S_2\})$. By a similar procedure, we compute a minimal siphon $S_3 = \{p_4, p_7, p_{10}, p_{11}\}$ to decompose the problem $\Delta_3 = (\Sigma_3, \emptyset)$ and it is updated that $\Gamma[3] = S_3$, $\Gamma[4] = \emptyset$ and $\Theta = \{S_1, S_2, S_3\}$.

When we consider the first case that place p_4 is deleted from the net Σ_3, we obtain an empty net. Hence, we consider the second case that place p_7 is deleted from the net Σ_3 and it is required that p_4 is included in minimal siphons to be found. In this case, we derive a new node $\Delta_4 = (\Sigma_4, P_\Omega)$ in the tree with an arc labeled by "p_7" from node $\Delta_3 = (\Sigma_3, \emptyset)$, where Σ_4 is a net with the set of places $P_4 = \{p_2 - p_4\}$ and $P_\Omega = \{p_2 - p_4\}$ derived by expanding the set $\{p_4\}$. It is verified that $P_\Omega = \{p_2 - p_4\}$ is a siphon and thus $\Delta_4 = (\Sigma_4, P_\Omega)$ does not need to be further decomposed. It is verified that $P_\Omega = \{p_2 - p_4\}$ is not a minimal siphon and thus we do not update the set Θ. Then, we consider the third case that place p_{10} is deleted from the net Σ_3 and it is required that p_4 and p_7 are included in minimal siphons to be found. Similarly, a new node $\Delta_5 = (\Sigma_5, P_\Omega)$ is generated in the tree with an arc labeled by "p_{10}" from node $\Delta_3 = (\Sigma_3, \emptyset)$, where Σ_5 is a net with the set of places $P_5 = \{p_2 - p_4\}$ and $P_\Omega = \{p_4, p_7\}$. $\Delta_5 = (\Sigma_5, P_\Omega)$ does not need to be further decomposed because $P_\Omega \not\subset P_5$. Finally, we consider the fourth case that p_{11} is deleted from the net Σ_3 but it is required that p_4, p_7, p_{11} are included in minimal siphons to be found. Similarly, a new node $\Delta_6 = (\Sigma_6, P_\Omega)$ is generated in the tree with an arc labeled by "p_{11}" from node $\Delta_3 = (\Sigma_3, \emptyset)$, where Σ_6 is a net with the set of places $P_6 = \{p_2 - p_4\}$ and $P_\Omega = \{p_4, p_7, p_{10}\}$. $\Delta_6 = (\Sigma_6, P_\Omega)$ does not need to be further decomposed as well because $P_\Omega \not\subset P_6$.

Now, the decomposition of the problem $\Delta_3 = (\Sigma_3, \emptyset)$ is finished. It is $\Theta = CreateNewNode(\Sigma_3, \emptyset, 3, \{S_1, S_2\}) = \{S_1, S_2, S_3\}$. We then go back to consider the second case of decomposing the problem $\Delta_2 = (\Sigma_2, \emptyset)$, i.e., place p_6 is deleted from the net Σ_2 and it is required that p_5 is included in minimal siphons to be found. In this case, a new node Δ_7 is generated in the tree.

By repeating the above similar procedure, we may obtain nodes Δ_8, Δ_9, …, Δ_{17} one by one, resulting in the tree shown in Fig. 2. The final set Θ computed by IGPMSE is $\Theta = \{S_1 - S_7\}$, which means that the set of all minimal siphons of the net in Fig. 1 is $\{S_1 - S_7\}$. The details of $S_1 - S_7$ can be found in Fig. 2.

We notice that we use colored nodes to indicate those problems that do not need to be further decomposed. In particular, red nodes imply that P_Ω is a siphon but not minimal; yellow nodes imply that P_Ω is a minimal siphon; black nodes imply that $P_\Omega \not\subset P$.

V. Comparison

In this section, we compare the performance of the proposed IGPMSE and the original GPMSE in [32]. Note that, although many new methods are proposed after GPMSE for computing minimal siphons in PN, most of them are applicable to a specific class of PN only. Thus, although they might have higher computational efficiency than our proposed method, ours has the advantage of having no restriction on the class of PN. On the other hand, concerning methods in the literature applicable to any class of PN, to our best knowledge, the methods proposed by Cordone *et al.* [32] are still the most efficient methods among them. As a result, we compare the proposed method (IGPMSE) with the GPMSE [32] only in the paper.

First, we make the comparison by applying them both to the PN in Fig. 1. We have already obtained the tree generated by using IGPMSE, as shown in Fig. 2. Now, we construct the tree by using GPMSE, which is presented in Fig. 3. It is clear that the tree generated by IGPMSE is much simpler than that generated by GPMSE. In Table I, we provide more details to

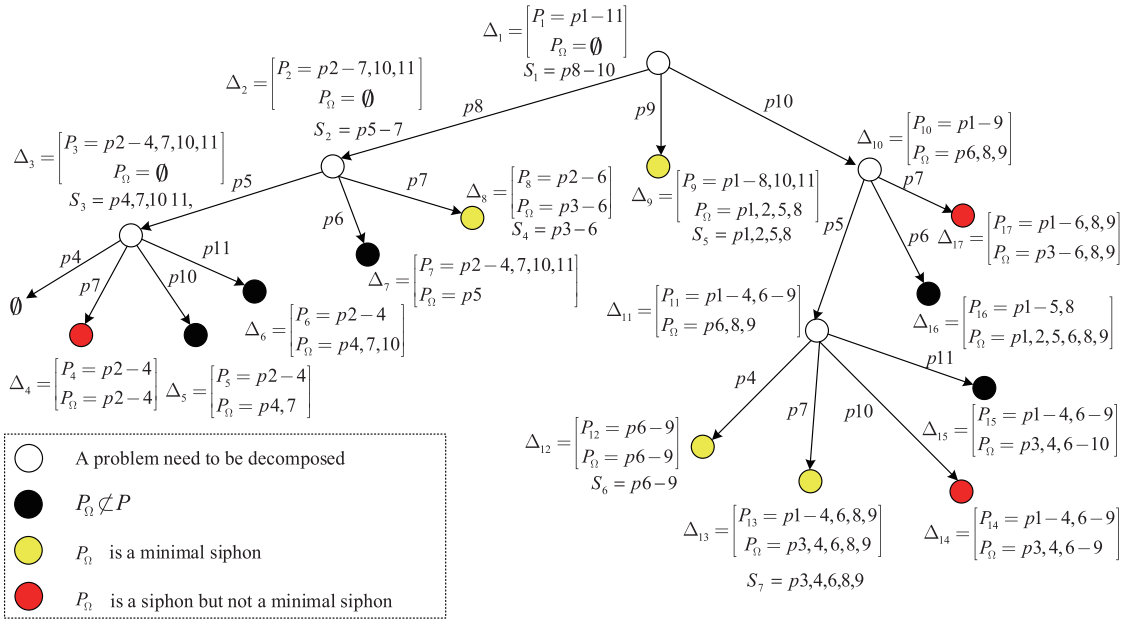


Fig. 2. Tree generated by IGPMSE.

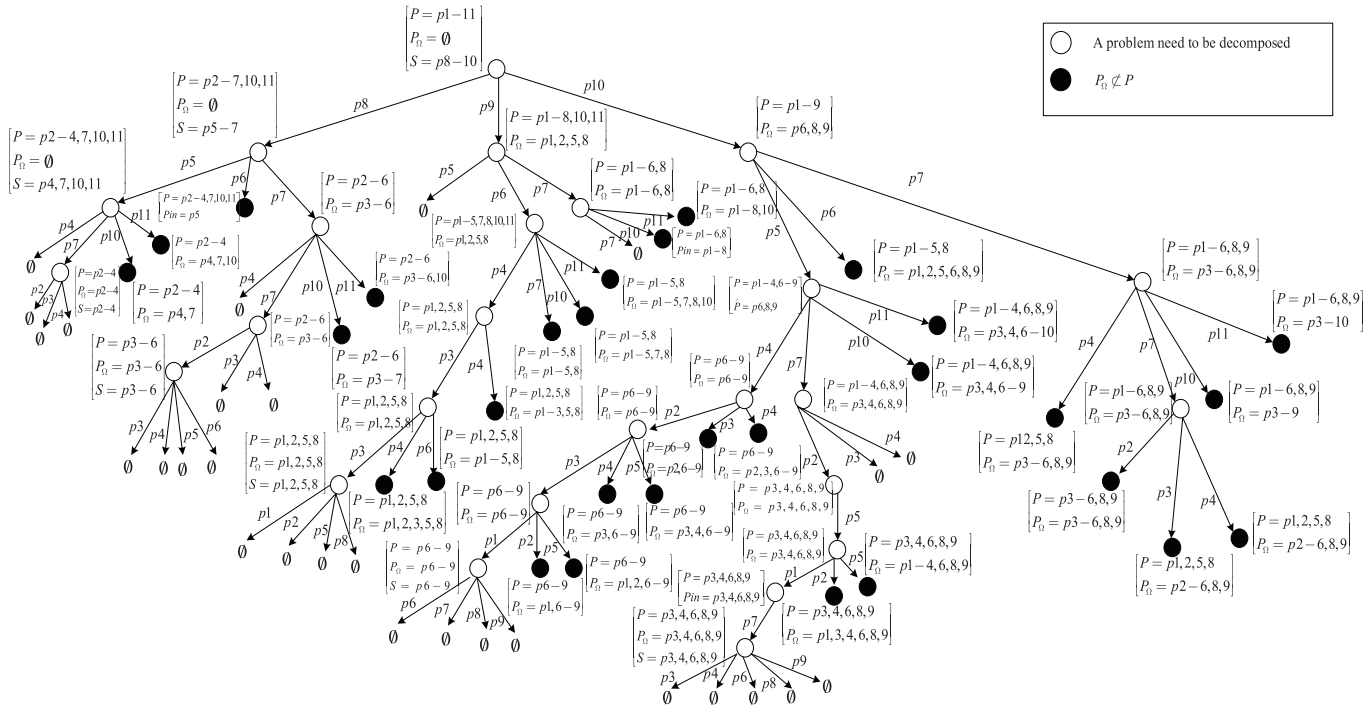


Fig. 3. Tree generated by GPMSE in [32].

compare them. In particular, the number of nodes of the tree is listed in the 2nd column; the number of nodes where a problem needs to be decomposed is listed in the 3rd column; and the last column shows the maximum number of nodes that need to be saved in memory during the construction of the tree (considering the depth-first search). We note that the computational time is closely related to the number of nodes of the tree and the number of nodes that need to be decomposed, whereas the memory consumption depends on the maximum number of nodes that need to be saved during computation. Thus, according to Table I, we may see that IGPMSE performs better than GPMSE when applied to the PN in Fig. 1, both in terms of computational time and memory consumption.

Next, in order to further validate the advantage of IGPMSE over GPMSE, we developed a tool, written in C++, to implement three methods, namely, GPMSE, GPMSE using the depth-first search, and IGPMSE. The readers can download this tool from [34]. Using this tool, we carry out an experiment where we enumerate the minimal siphons in a series of PNs with the net size n (i.e., the number of places and transitions) ranging from 31 to 42. For every net size, we consider 100 nets that are randomly generated with $d_i = d_o = 0.05$, where d_i is the probability of having an arc going from

TABLE I
COMPARISON RESULTS OF GPMSE AND IGPMSE APPLIED FOR THE ENUMERATION OF MINIMAL SIPHONS IN THE PN IN FIG. 1

Method	Number of nodes	Number of decomposed nodes	Maximum number of nodes to be saved in memory
GPMSE	56	26	6
IGPMSE	17	5	3

any place to any transition and d_o is the probability of having an arc going from any transition to any place. The experiment was performed in the environment with Intel Processor at 2.53 GHz and 3 GB memory under Windows 7 operating system.

We found that GPMSE fails to get a result for any of such net sizes (i.e., n = 31, 32, …, 42) due to running out of memory. In contrast, both GPMSE using the depth-first search and IGPMSE can get results. We thus provide the experimental results from these two approaches in Fig. 4, where we show the average CPU time (over the 100 simulations) versus the net size and we use a blue line to denote GPMSE with the depth-first search and a red line to denote IGPMSE. Fig. 4(a) provides the results for all the considered net sizes. We may see that the computational time of GPMSE with the depth-first search grows exponentially and reaches more than 100 seconds when n = 42, whereas our method (IGPMSE) costs less than 10 seconds in all cases. Also, we provide a closer zoom in Fig. 4(b) of those results for n from 34 to 37, which again shows the superiority of IGPMSE over GPMSE with the depth-first search.

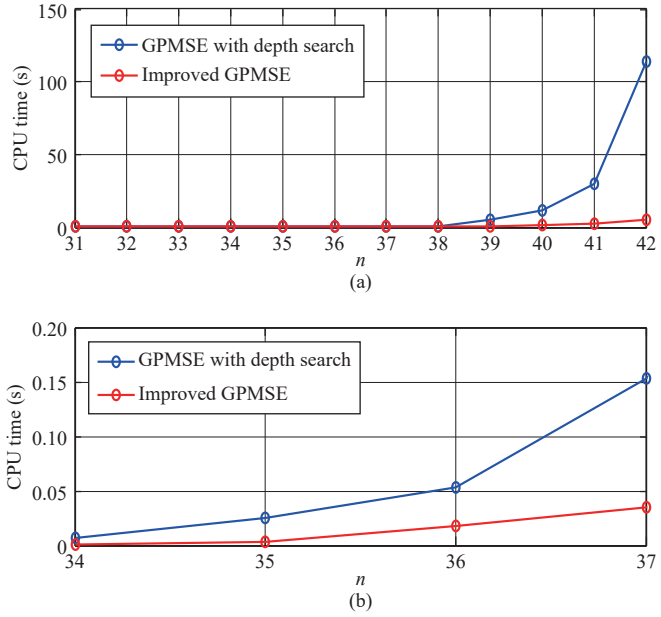


Fig. 4. Experimental results of the application of GPMSE with the depth-first search and IGPMSE. (a) The average CPU time versus the net size n from 31 to 42; (b) A closer zoom of the results in Fig. 4(a) for the net size n from 34 to 37.

From this experiment, we conclude that IGPMSE consumes less memory than GPMSE. Also, we argue that IGPMSE behaves better than GPMSE in computational time, which becomes more evident when the size of the handled net grows.

We conclude this section by mentioning another approach in [35] that computes minimal siphons based on the problem decomposition as well. We note that the approach [35] applies to S^3PR only, whereas the proposed IGPMSE applies to arbitrary PNs.

VI. CONCLUSIONS AND FUTURE WORK

This paper improves an algorithm for the minimal-siphon computation developed in [32], i.e., the algorithm of global partitioning minimal-siphon enumeration (GPMSE). It should be evident from the above discussions that the improved algorithm, namely IGPMSE, provides a tractable decomposition of such a problem. Through the application of several PNs, it is shown that IGPMSE has lower computational complexity and, more importantly, the memory consumption has been reduced significantly by employing the depth-first search. Thus, the proposed IGPMSE constitutes an important step to tackle with minimal-siphon computation in large nets.

Similar improvements can also be made on local partitioning minimal-siphon enumeration (LPMSE) developed in [32], which is left as our future work. In addition, other directions for further research include: 1) Finding more constraints that could further reduce the number of problems in the problem list; and 2) Establishing complete algorithms for supervisory control problems. Furthermore, we may consider the use of parallel computing so that the sub-problems can be solved concurrently.

REFERENCES

[1] A. Giua and M. Silva, "Petri nets and automatic control: A historical perspective," *Annual Reviews in Control*, vol. 45, pp. 223–239, 2018.

[2] K. Y. Xing, M. C. Zhou, F. Wang, H. Liu, and F. Tian, "Resource-transition circuits and siphons for deadlock control of automated manufacturing systems," *IEEE Trans. Syst. Man Cybern. Part A*, vol. 41, no. 1, pp. 74–84, 2011.

[3] X. Wang and H. Hu, "A robust control approach to automated manufacturing systems allowing multi-type and multi-quantity of resources with Petri nets", *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, DOI: 10.1109/TSMC.2018.2852946.

[4] M. Uzam, G. Gelen, and T. L. Saleh, "Think-globally-act-locally approach with weighted arcs to the synthesis of a liveness-enforcing supervisor for generalized Petri nets modeling FMSs," *Inf. Sci.*, vol. 363, pp. 235–260, 2016.

[5] Q. Zeng, C. Liu, H. Duan, and M. Zhou, "Resource conflict checking and resolution controller design for cross-organization emergency response," *IEEE Trans. Systems, Man and Cybernetics: Systems*, vol. 50, no. 10, pp. 3685–3700, 2020.

[6] Q. Zeng, C. Liu, and H. Duan, "Resource conflict detection and removal strategy for nondeterministic emergency response processes using Petri nets," *Enterprise Information Systems*, vol. 10, no. 7, pp. 729–750, 2016.

[7] G. Y. Liu and K. Barkaoui, "A survey of siphons in Petri nets," *Inf. Sci.*, vol. 363, pp. 198–220, 2015.

[8] S. Wang, W. Duo, X. Guo, X. Jiang, D. You, K. Barkaoui and M. Zhou, "Computation of an emptiable minimal siphon in a subclass of Petri nets using mixed-integer programming," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 1, pp. 219–226, 2021.

[9] D. You, S. Wang, and M. Zhou, "Computation of strict minimal siphons in a class of Petri nets based on problem decomposition," *Inf. Sci.*, vol. 409–410, pp. 87–100, Oct. 2017.

[10] M. Gan, S. Wang, Z. Ding, M. Zhou, and W. Wu, "An improved mixed-integer programming method to compute emptiable minimal siphons in

S3PR nets," *IEEE Trans. Control Systems Technology*, vol. 26, no. 6, pp. 2135–2140, 2017.

[11] Y. Hou, K. Barkaoui, "Deadlock analysis and control based on Petri nets: A siphon approach review," *Advances in Mechanical Engineering*, vol. 9, no. 5, pp. 1–30, 2017.

[12] S. G. Wang, C. Y. Wang, M. C. Zhou, and Z. W. Li, "A method to compute strict minimal siphons in S3PR based on loop resource subsets," *IEEE Trans. Syst.*, *Man*, *Cybern.*, *A*, *Syst.*, *Humans*, vol. 42, no. 1, pp. 226–237, 2012.

[13] S. G. Wang, C. Y. Wang, M. C. Zhou, "Controllability conditions of resultant siphons in a class of Petri nets," *IEEE Trans. Syst. Man Cybern. Part A*, vol. 42, no. 5, pp. 1206–1215, 2012.

[14] S. Tanimoto, M. Yamauchi, and T. Watanabe, "Finding minimal siphons in general Petri nets," *IEICE Trans. Fundam.*, vol. E79-A, no. 11, pp. 1817–1824, 1996.

[15] B. Huang, M. Zhou, C. Wang, A. Abusorrah, and Y. Al-Turki, "Deadlock-free supervisor design for robotic manufacturing cells with uncontrollable and unobservable events," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 3, pp. 597–605, 2020.

[16] J. Luo, Z. Liu, S. Wang, and K. Xing, "Robust Deadlock Avoidance Policy for Automated Manufacturing System With Multiple Unreliable Resources," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 3, pp. 812–821, 2020.

[17] O. Karoui, Y. Chen, Z. Li, N. Wu and M. Khalgui, "On hierarchical construction of the state space of an automated manufacturing system modeled with Petri nets," *IEEE Trans. Systems*, *Man*, *and Cybernetics*: *Systems*, vol. 50, no. 10, pp. 3613–3627, Oct. 2020.

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

[19] C. F. Zhong, Z. W. Li, & Barkaoui, K., "Monitor design for siphon control in S4R nets: from structure analysis points of view," *Int. J. Innovative Comput.*, *Inf. Control*, vol. 7, no. 12, pp. 6677–6690, 2011.

[20] 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. 12, no. 6, pp. 4255–4265, 2018.

[21] S. Wang, D. You, and M. Zhou, "A necessary and sufficient condition for a resource subset to generate a strict minimal siphon in S4PR," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 4173–4179, Aug. 2017.

[22] Q. Zhuang, W. Dai, S. Wang, and F. Ning, "Deadlock prevention policy for S4PR nets based on siphon," *IEEE Access*, vol. 6, pp. 50648–50658, 2018.

[23] F. Tricas and J. Ezpeleta, "Computing minimal siphons in Petri net models of resource allocation systems: A parallel solution," *IEEE Trans. Systems*, *Man*, *and Cybernetics*, *Part A*: *Systems and Humans*, vol. 36, no. 3, pp. 532–539, 2006.

[24] A. R. Wang, Z. W. Li, J. Y. Jia, and M. C. Zhou, "An effective algorithm to find elementary siphons in a class of Petri nets," *IEEE Trans. Systems*, *Man and Cybernetics*, *Part A*, vol. 39, no. 4, pp. 912–923, Jul. 2009.

[25] Z. W. Li and M. C. Zhou, "On siphon computation for deadlock control in a class of Petri nets," *IEEE Trans. Syst.*, *Man*, *Cybern.*, *A*, *Syst.*, *Humans*, vol. 38, no. 3, pp. 667–679, 2008.

[26] E. E. Cano, C. A. Rovetto, J. M. Colom, "An algorithm to compute the minimal siphons in S^4PR nets," *Discrete Event Dynamic Systems*, vol. 22, no. 4, pp. 403–428, 2012.

[27] F. Tricas, J. M. Colom, and J. J. Merelo, "Using the incidence matrix in an evolutionary algorithm for computing minimal siphons in Petri net models," In *Proc. 18th Int. Conf. System Theory, Control and Computing, Sinaia, Romania*, 2014, pp. 645–651.

[28] F. Tricas, J. M. Colom, and J. J. Merelo, "Computing minimal siphons in Petri net models of resource allocation systems: An evolutionary approach," In *Proc. Int. Workshop on Petri Nets and Software Engineering* (PNSE'14), Tunis, 2014, Tunisia, pp. 307–322.

[29] L. Piroddi, R. Cordone, and I. Fumagalli, "Combined siphon and marking generation for deadlock prevention in Petri nets," *IEEE Trans. Systems*, *Man*, *and Cybernetics*, *Part A*: *Systems and Humans*, vol. 39, no. 3, pp. 650–661, 2009.

[30] P. H. Starke, "INA: Integrated net analyzer," 1992. [Online]. Available: http://www2.informatik.huberlin.de/~starke/ina.html.

[31] X. Han, Z. Chen, Z. Liu and Q. Zhang, "Calculation of siphons and minimal siphons in Petri nets based on semi-tensor product of matrices," *IEEE Trans. Systems*, *Man*, *and Cybernetics*:*Systems*, vol. 47, no. 3, pp. 531–536, 2017.

[32] R. Cordone, L. Ferrarini, and L. Piroddi, "Enumeration algorithms for minimal siphons in Petri nets based on place constraints," *IEEE Trans. Syst.*, *Man*, *Cybern.*, *A*, *Syst.*, *Humans*, vol. 35, no. 6, pp. 844–854, Nov. 2005.

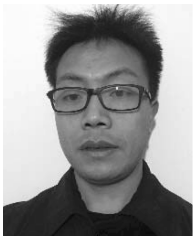
[33] T. Murata, "Petri nets: Properties, analysis and applications", in *Proc IEEE*, 1989, vol. 77, no. 4, pp. 541–580.

[34] "The tools for GPMSE, GPMSE with the depth-first search, and IGPMSE", 2015. [Online]. Available: https://www.dropbox.com/s/7cpho1ka35gpq3i/Tools.zip?dl=0

[35] C. Chen and H. S. Hu, "Liveness-enforcing supervision in AMS-oriented HAMGs: An approach based on new characterization of siphons using Petri nets," *IEEE Trans. Autom. Control*, vol. 63, no. 7, pp. 1987–2002, Jul. 2018.

Dan You (*Member, IEEE*) 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, and the Ph.D. degree from the Department of Electrical and the Electronic Engineering, the University of Cagliari, Italy, in 2021. She is currently a Member of the Discrete-Event System Group, School of Information and Electronic Engineering, Zhejiang Gongshang University. Her research interests include supervisory control of discrete event systems, fault prediction, and deadlock control and siphon computation in Petri nets.

Oussama Karoui (*Associate Member, IEEE*) received the B.S. and M.S. degrees in computer networks and telecommunications from the National Institute of Applied Sciences and Technology, Tunis, Tunisia, in 2012 and 2015, respectively, and the Ph.D. degree in computer technology and application from Macau University of Science and Technology, Macau, China, in 2018. He joined the School of Information and Electronic Engineering (Sussex AI Institute) at Zhejiang Gongshang University, Hangzhou, China, in 2019. His current research interests include modeling, control and scheduling of discrete event systems, intelligent transportation systems, mobile robotics, and machine learning.



Shouguang Wang (*Senior Member, IEEE*) 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. He joined Zhejiang Gongshang University in 2005, 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 System modeling and Control Research Institute, Zhejiang Gongshang University. He is currently an Associate Editor of *IEEE Access* and *IEEE/CAA Journal of Automatica Sinica*. He was a Visiting Professor with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, from 2011 to 2012. He is 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 in Zhejiang Gongshang University from 2011 to 2014. His research interests include supervisory control of discrete event systems and siphon computation in Petri nets.