Proceedings of the 42nd IEEE
Conference on Decision and Control
Maui, Hawaii USA, December 2003

Some Results on the Computation of Minimal Siphons in Petri Nets

R. Cordone
Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano
rcordone@crema.unimi.it

L. Ferrarini, *Senior Member, IEEE*, and L. Piroddi
Dipartimento di Elettronica e Informazione
Politecnico di Milano
ferrarin@elet.polimi.it, piroddi@elet.polimi.it

ABSTRACT

The problem addressed in th e paper is that of computing minimal siphons in standard Petri nets. In particular, some new theoretical results are stated and proved which aim at reducing the original problem to a set of smaller sub-problems. Based on that, a conceptual algorithm is proposed which efficiently computes a set of siphons containing the minimal siphons of a net. An experimental prototypical version of the proposed search algorithm has been developed and a campaign on a large set of random test instances has been carried out to evaluate the effectiveness and efficiency of the proposed method.

1 INTRODUCTION

Siphons and traps are well-known structures in Petri nets [12, 13], related to basic behavioral properties of a Petri net, like reachability, reversibility and liveness [4, 7]. Such properties are extremely useful for the characterization of correct system behavior of large classes of systems, such as FMS, batch processes, communication systems, and any discrete production system in general.

In short, a siphon is a set of places whose input transitions are also output transitions, and symmetrically, a trap is a set of places whose output transitions are also input transitions. It can be easily proved that a siphon remains empty of tokens once it becomes empty. A trap, on the other hand, is a set of places which remains indefinitely marked once a token enters in it.

Actually, due to the symmetric definitions, an algorithm used to find siphons can be used also to find traps. But in spite of the simplicity of their definition, the computation of siphons and traps is far from trivial a problem. In the literature, many algorithms have been proposed: these are based on inequalities [12], logic equations [9, 11], algebraic approaches [10], linear equations with slack variables [8], structural properties [2, 3, 6]. The authors have explored an approach based on a binary programming formulation of the problem, with linear constraints and objective function [5]. A general-purpose MIP solver is employed to find the siphon of minimum cardinality in the net. Then, new constraints are iteratively added to the formulation in order to find all minimal siphons in the net, without generating any non-minimal siphon. Another interesting approach is documented in [7], where a constructive algorithm is proposed based on a depth-first search of different combinations of places. The algorithm builds a branching tree in which the single paths from the root to each leaf correspond to the subsequent addition of places to a subset which is candidate to be a siphon.

In the present work, an innovative algorithm based on a completely different branching approach is proposed. In particular, the algorithm starts with the search for a generic siphon, in order to find a minimal siphon contained in it. Then, based on a theoretical foundation suitably developed, the original problem is decomposed in a set of smaller sub-problems, which are definitely easier to solve. Each sub-problem in turn requires finding a minimal siphon in a "smaller" net, i.e. in the original net where some trivial constraints are imposed on the solutions, such as including or removing some specific places. These constraints can be easily handled by the basic procedure, which can be therefore applied at any iteration.

The structure of the paper is now briefly described. In Section 2, basic definitions on Petri nets and minimal siphons are concisely recalled. In Section 3, the main theoretical results obtained in the characterization of siphons in general and of minimal siphons in particular are illustrated. A conceptual algorithm for finding minimal siphons is presented in Section 4. The algorithm directly exploits the theoretical foundations previously developed, as shown in an illustrative example discussed in Section 5. The very first results obtained with a prototype computation program launched on randomly generated Petri nets are shown in Section 6. Finally, some conclusions are drawn in Section 7.

2 BACKGROUND ON PETRI NETS AND SIPHONS

A Generalities on PNs

The definition of a Petri net is here briefly recalled. More details on Petri nets can be found in [12, 13]. A Petri net (PN) is a bipartite graph, where nodes are classified either as *places* or *transitions*, and directed arcs connect places to transitions and transitions to places. Places are endowed with integer variables called *tokens*, while arcs are labeled with integer numbers called *weights*. More formally, a PN structure (or unmarked PN) can be defined as a triplet (P, T, F), where P is a set of n places and T is a set of m transitions. F represents the *flux relation*, i.e. the relation between places and transitions. F is a sub-set of $P{\times}T \cup T{\times}P$, which means that places can be connected only to transitions, and that transitions can be connected only to places. The flux relation can be given in the form of matrices, namely the input (I), output (O) and incidence

($C = O-I$) matrices. The generic element i_{kj} of the $I_{m\times n}$ matrix represents the weight (0 or 1 for standard PNs) of the arc from place p_k to transition t_j. Conversely, the generic element o_{kj} of matrix $O_{m\times n}$ represents the weight of the arc from transition t_j to place p_k. A *marking* M: $P \rightarrow N$ defines the distribution of tokens in places. M_0: $P \rightarrow N$ is the initial marking. A transition $t_j \in T$ is said to be *enabled* in a marking M if $M \geq I_j$, where I_j is the j-th column of input matrix I. An enabled transition may *fire*, yielding the marking $M^* = M + O_j - I_j = M + C_j$, where O_j, I_j and C_j are the j-th columns of the O, I and C matrices, respectively.

B Siphons

Let S be a set of places in a PN. Then, the pre-set of S, $\bullet S = \{t_j \in T \mid o_{kj} \neq 0, \text{ with } p_k \in S\}$, is the set of input transitions for the places in S, while the post-set of S, $S\bullet = \{t_j \in T \mid i_{kj} \neq 0, \text{ with } p_k \in S\}$, is the set of output transitions for the places in S.

A *siphon* is a set S of places such that $\bullet S \subseteq S\bullet$, while conversely, a *trap* is a set S of places such that $S\bullet \subseteq \bullet S$. The number of siphons of a PN is not known a priori, but experience shows that it grows exponentially with n and m [3]. Luckily, a significant formal analysis does not require the computation of all the siphons in a PN [7, 12, 13]. One only needs to compute the *minimal* siphons [12]. A siphon is said to be a *minimal* siphon if it does not contain any other siphon.

3 SOME THEORETICAL RESULTS ON SIPHONS AND MINIMAL SIPHONS

In the following, some definitions and theoretical results are introduced which can be exploited in the siphon search problem. The first important observation is that if we are interested in the siphons of a Petri net contained in a specified set of places $\tilde{P}$, all the places not in $\tilde{P}$ (and the arcs connected with them) can be discarded from the net, thus greatly simplifying the problem.

Definition 1 — Let $G = (P, T, F)$ be a Petri net and $\tilde{P} \subset P$. The reduction function *red* is defined as follows: $\tilde{G} = red(G, \tilde{P})$, where the reduced net $\tilde{G} = (\tilde{P}, \tilde{T}, \tilde{F})$ is defined by:

i. $\tilde{T} = \{t \in T \mid (\bullet t \cup t\bullet) \cap \tilde{P} \neq \varnothing\}$,

ii. $\tilde{F}(p, t) = F(p, t), \tilde{F}(t, p) = F(t, p), \forall p \in \tilde{P}, \forall t \in \tilde{T}$. ❑

Lemma 1 — Let $G = (P, T, F)$ be a Petri net and $\tilde{P} \subset P$. The set of siphons of G contained in $\tilde{P}$ coincides with the set of siphons of the reduced net $\tilde{G} = red(G, \tilde{P})$. ❑

Proof. Consider a set of places $S' \subseteq \tilde{P}$ in G and let $\tilde{S}'$ be the corresponding set of places in $\tilde{G}$. Now, $\bullet S' = \bullet \tilde{S}'$ and $S'\bullet = \tilde{S}'\bullet$, since all the transitions connected to places in $\tilde{P}$ are preserved by the *red* operator in net $\tilde{G}$, as well as the corresponding arcs. Then the thesis follows. ❑

The siphon search problem can then be tackled by suitable net reduction, exploiting the fact that it is much easier to recognize a siphon in a sub-net with a certain structure, than to spot the same siphon in the original un-reduced net. The following two lemmas identify obvious siphons. A first trivial result concerns places with empty pre-set: clearly, these constitute minimal siphons and can be immediately eliminated from further search.

Lemma 2 — Let $G = (P, T, F)$ be a Petri net and let $\bar{P} \subseteq P$ be such that $\bullet \bar{P} = \varnothing$. Then, the minimal siphons of G are the minimal siphons of the reduced net $\tilde{G} = red(G, P-\bar{P})$, plus the individual places in $\bar{P}$. ❑

Proof. Each place in $\bar{P}$ is clearly a siphon. It is also a minimal siphon since it is a siphon with cardinality 1. All other minimal siphons of G, if any, do not contain any place belonging to $\bar{P}$, and by Lemma 1 can be directly searched in the reduced net $\tilde{G}$. ❑

Another obvious siphon case is when the post-set of the set of places of a net coincides with the whole set of net transitions.

Lemma 3 — Let $G = (P, T, F)$ be a Petri net such that $P\bullet = T$. Then, P is a siphon. ❑

Proof. $\bullet P \subseteq T = P\bullet$. ❑

Places in the post-set of transitions with empty pre-set cannot belong to a siphon and can be eliminated from further search. On the other hand, if there are no transitions with empty pre-set in a Petri net, all the transitions are in the post-set of some place, i.e. the condition $T = P\bullet$ holds and by Lemma 3 the set P is itself a siphon. A smaller siphon can then be found by repeated elimination of places with the characteristics defined in the following Lemma 4.

Lemma 4 — Let $G = (P, T, F)$ be a Petri net and let $\bar{T} \subseteq T$ be such that $\bullet \bar{T} = \varnothing$. Define also $\bar{P} = \bar{T}\bullet$. Then, G has the same siphons of the reduced net $\tilde{G} = red(G, P-\bar{P})$. ❑

Proof. Places connected to a source transition, i.e. places which have an input transition with empty pre-set, cannot belong to any siphon of G, since no places in P can have that transition in their post-set. By Lemma 1, these places can be eliminated from G without losing any siphons. ❑

Given a siphon of a Petri net, it remains to ascertain if it is minimal and, in the negative case, to find a minimal siphon contained in it. The following lemma, derived from the concept of "own transition" [1] and from an equivalent result originally reported in [6], gives a sufficient condition for eliminating inessential places from a given siphon, thus reducing its size.

Lemma 5 — Let $G = (P, T, F)$ be a Petri net and $S \subseteq P$ a siphon of G. If $\exists\ \bar{p} \in S$ such that $\forall t \in \bar{p}\bullet$, either $(\bullet t \cap S) \supset \{\bar{p}\}$ or $t\bullet \cap S = \varnothing$, then $S-\{\bar{p}\}$ is also a siphon of G. ❑

Proof. Place $\bar{p}$ is such that every transition in its post-set either has at least another place of S in its pre-set or is a sink transition with respect to the places in S. This implies that every transition in $S\bullet$ is also in $(S-\{\bar{p}\})\bullet$, except sink transitions. Also, $\bullet(S-\{\bar{p}\}) \subseteq \bullet S$, and clearly cannot contain sink transitions. Therefore, $\bullet(S-\{\bar{p}\}) \subseteq (S-\{\bar{p}\})\bullet$. ❑

Actually, in [6] this property is used to construct minimal siphons by subsequent addition of places, whereas in the present work, it is employed to reduce non minimal siphons.

It is important to notice that a siphon that has no eliminable places according to Lemma 5 is not necessarily minimal. For example, the union of two minimal siphons cannot be further reduced in this way. The concept of "alternating circuit" [1] could be used to verify the minimality of a siphon S. However, in the negative case, we still need to find a minimal siphon contained in S. Therefore, a different approach is adopted, which directly applies the definition of minimality to check exhaustively for smaller siphons according to the following Lemma 6.

Lemma 6 — Let $G = (P, T, F)$ be a Petri net and $S \subseteq P$ a siphon of G. S is a minimal siphon for G iff all the reduced nets $\widetilde{G}_p = red(G, S-\{p\})$, $\forall p \in S$, do not contain siphons. ❑

Proof. By Lemma 1 all siphons of G contained in S are also siphons of $\widetilde{G} = red(G, S)$. These siphons are S itself, plus all the strictly smaller siphons contained in the reduced nets $\widetilde{G}_p = red(G, S-\{p\})$, $\forall p \in S$. ❑

When a minimal siphon S is found, the search for further minimal siphons can exclude all the siphons which contain S. For this purpose, the following decomposition of the siphon search problem in the analysis of suitable sub-nets, can be adopted. In this way the number of nets to be searched is increased, but the individual net size is decreased: in other words, a complicated problem is transformed in several simpler sub-problems.

Lemma 7 — Let $G = (P, T, F)$ be a Petri net and $\widetilde{P} = \{p_1, p_2, ..., p_n\} \subset P$. The set of siphons of G not containing $\widetilde{P}$ is equal to $\bigcup_{i=1}^{n} \Sigma_i$, where Σ_i is the set of siphons of the reduced net $\widetilde{G}_i = red(G, P-\{p_i\})$, $i = 1, ..., n$, containing $\{p_1, p_2, ..., p_{i-1}\}$. ❑

Proof. The set of siphons of G not containing $\widetilde{P}$ can be divided as $\bigcup_{i=1}^{n} \Sigma_i$, where Σ_i is the set of siphons of G, containing $\{p_1, p_2, ..., p_{i-1}\}$ and not p_i. By Lemma 1, the generic Σ_i can be computed as the set of siphons of the reduced net $\widetilde{G}_i = red(G, P-\{p_i\})$, containing $\{p_1, p_2, ..., p_{i-1}\}$. ❑

4 AN ALGORITHM FOR FINDING MINIMAL SIPHONS

In view of the lemmas introduced in the preceding section, a recursive search algorithm can be devised to find all minimal siphons in a Petri net. This algorithm is based on suitable net reduction techniques and problem decomposition to explore the solution space. The general idea can be summarized in the following steps:

Step 1)	Find a generic siphon.
Step 2)	Find a minimal siphon contained in the generic siphon.
Step 3)	Apply Lemma 7 to decompose the problem, and for each sub-problem (corresponding to a specific sub-net and some place constraints) apply the same procedure from Step 1.

Denoting by n and a the numbers of places and arcs, respectively, it can be shown that the search for a generic siphon (Step 1) can be carried out in linear time with respect to a, even if place constraints are imposed. The search for a minimal siphon (Step 2) within a given siphon has a complexity of order $O(na)$. Finally, the number of overall recursive calls (Step 3) turns out to be $O(2^n)$ in the worst case, since each call returns a different siphon, and there are at most $O(2^n)$ siphons in a net. Therefore, the worst-case complexity of the algorithm is $O(an2^n)$. The average complexity is much lower, since the number of recursive calls tends to be equal to the number of minimal siphons.

The algorithm requires two secondary functions, for the computation of a generic siphon, subject to place constraints, and for the computation of a minimal siphon contained in a given siphon, again subject to place constraints.

Let $G = (P, T, F)$ be a Petri net. The following function *FindSiphon*$(G, \widetilde{P})$ can be used to find siphons in G that contain a specific set of places $\widetilde{P}$. More precisely, it outputs an empty set if G has no such siphons, or a generic (not necessarily minimal) siphon if G contains any.

Algorithm 1 — $S = FindSiphon(G, \tilde{P})$

```
Step 1)    IF $\exists p \in P \cap \tilde{P}$ such that $\exists t \in \bullet p$, $t \notin P\bullet$, THEN
           $S = \varnothing$, END
Step 2)    IF $\exists p \in P - \tilde{P}$ such that $\exists t \in \bullet p$, $t \notin P\bullet$, THEN
           GOTO Step 3, ELSE $S = P$, END
Step 3.1)  $G = red(G, P - \{p\})$
Step 3.2)  GOTO Step 1 ❑
```

If a place-elimination in $\tilde{P}$ compatible with Lemma 4 is effected (Step 1 of algorithm 1), there are no possible siphons that contain $\tilde{P}$. Therefore the algorithm ends with empty outcome. When all other place eliminations compatible with Lemma 4 (Steps 2-3) have been performed, there is no place left in the remaining net with a transition in its pre-set which is not in the post-set of at least one place. Therefore, Lemma 3 applies, and the set of remaining places constitutes a siphon for the reduced net. By Lemma 4, this siphon is also a siphon for the original Petri net.

Let $G = (P, T, F)$ be a Petri net, $\tilde{S} \subseteq P$ a siphon of G and $\tilde{P} \subseteq \tilde{S}$ a specific set of places. The following function $FindMinSiphon(G, \tilde{S}, \tilde{P})$ computes one minimal siphon S in G, such that $\tilde{P} \subseteq S \subseteq \tilde{S}$, i.e. such that it is contained in $\tilde{S}$ and contains $\tilde{P}$, if any exists. The function outputs $\tilde{S}$ if G has no smaller siphons satisfying the stated conditions.

Algorithm 2 — $S = FindMinSiphon(G, \tilde{S}, \tilde{P})$

```
Step 1)    IF $\exists p \in (P - \tilde{P}) \cap \tilde{S}$ such that $(\bullet t \cap \tilde{S}) \supset \{p\}$ or
           $t\bullet \cap \tilde{S} = \varnothing$, $\forall t \in p\bullet$, THEN GOTO Step 2,
           ELSE GOTO Step 3
Step 2.1)  $\tilde{S} = \tilde{S} - \{p\}$
Step 2.2)  GOTO Step 1
Step 3)    IF $\tilde{S} \subset P$, THEN $G = red(G, \tilde{S})$
Step 4)    $P_{new} = P - \tilde{P}$
Step 5)    IF $P_{new} = \varnothing$, THEN $S = \tilde{S}$, END
Step 6.1)  $G_p = red(G, P - \{p\})$, $p \in P_{new}$
Step 6.2)  $P_{new} = P_{new} - \{p\})$
Step 6.3)  $S_p = FindSiphon(G_p, \tilde{P})$
Step 6.4)  IF $S_p \neq \varnothing$, THEN $\tilde{S} = S_p$, GOTO Step 3, ELSE
           GOTO Step 5 ❑
```

The following function $FindAllMinSiphons(G, \tilde{P})$, which implements the algorithm introduced informally at the beginning of the section, finds all minimal siphons in G that contain a specific set of places $\tilde{P}$, if any exists. The function outputs an empty set if G has no such siphons.

Algorithm 3 — $\Sigma = FindAllMinSiphons(G, \tilde{P})$

```
Step 1)    $\Sigma = \varnothing$
Step 2)    IF $\tilde{P} = \varnothing$ AND $\exists p \in P$ such that $\bullet p = \varnothing$, THEN
           GOTO Step 3, ELSE GOTO Step 4
Step 3.1)  $S = \{p\}$
Step 3.2)  $\Sigma = \Sigma \cup \{S\}$
Step 3.3)  $G = red(G, P - \{p\})$
Step 3.4)  GOTO Step 2
Step 4)    $\tilde{S} = FindSiphon(G, \tilde{P})$
Step 5)    IF $\tilde{S} = \varnothing$, THEN END
Step 6.1)  $S = FindMinSiphon(G, \tilde{S}, \tilde{P})$
Step 6.2)  $\Sigma = \Sigma \cup \{S\}$
Step 7)    $P_{new} = S - \tilde{P}$, $P_{old} = \varnothing$,
Step 8)    IF $P_{new} = \varnothing$, THEN END
Step 9.1)  $G_p = red(G, P - \{p\})$, $p \in P_{new}$
Step 9.2)  $\Sigma_p = FindAllMinSiphons(G_p, \tilde{P} \cup P_{old})$
Step 9.3)  $\Sigma = \Sigma \cup \Sigma_p$
Step 9.4)  $P_{new} = P_{new} - \{p\}$, $P_{old} = P_{old} \cup \{p\}$
Step 9.5)  GOTO Step 8 ❑
```

By applying $FindAllMinSiphons(G, \varnothing)$ one obtains all minimal siphons in G.

5 AN ILLUSTRATIVE EXAMPLE

To clarify the behavior of the proposed algorithm, let us consider the simple Petri net G in Fig. 1, which has the two minimal siphons $S_1 = \{p_1, p_2, p_3\}$ and $S_2 = \{p_1, p_3, p_4\}$.

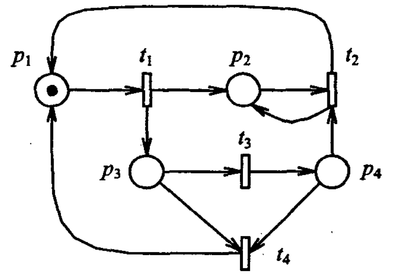

Figure 1. Petri net example

To compute the minimal siphons in the net the search algorithm is launched with no place constraints, $\Sigma = FindAllMinSiphons(G, \varnothing)$, resulting in the following steps:

```
Step 1)    $\Sigma = \varnothing$
Step 2)    $\exists p_i \in P$ such that $\bullet p_i = \varnothing$
Step 4)    $\tilde{S} = FindSiphon(G, \varnothing)$
```

There does not exist any place p_i such that $\exists\ t_j \in \bullet p_i$, $t_j \notin P\bullet$, therefore no place is eliminable according to Lemma 4, and $\tilde{S} = \{p_1, p_2, p_3, p_4\}$ is a siphon by Lemma 3.

Step 6.1) $S = FindMinSiphon(G, \tilde{S}, \varnothing)$

Place p_2 can be eliminated by Lemma 5, resulting in siphon $S = \{p_1, p_3, p_4\}$. No siphon contained in S is found, then S is minimal.

Step 6.2) $\Sigma = \{\{p_1, p_3, p_4\}\}$

The set of siphons containing S is eliminated from the solution space by the problem decomposition according to Lemma 7.

Step 7) $P_{new} = \{p_1, p_3, p_4\}$, $P_{old} = \varnothing$

The first case to examine concerns the reduced net without p_1 and with no place constraints.

Step 9.1) $G_{p1} = red(G, P-\{p_1\})$
Step 9.2) $\Sigma_{p1} = FindAllMinSiphons(G_{p1}, \varnothing)$

The algorithm is recursively applied on G_{p1} but immediately ends at Step 5 with no found siphons.

Step 9.4) $P_{new} = \{p_3, p_4\}$, $P_{old} = \{p_1\}$

The second case to examine concerns the search for siphons containing p_1 in the reduced net without p_3.

Step 9.1) $G_{p3} = red(G, P-\{p_3\})$
Step 9.2) $\Sigma_{p3} = FindAllMinSiphons(G_{p3}, \{p_1\})$

Again, the algorithm stops at Step 5 having not found any siphons.

Step 9.4) $P_{new} = \{p_4\}$, $P_{old} = \{p_1, p_3\}$

The third and last case concerns the search for siphons containing p_1 and p_3 in the reduced net without p_4.

Step 9.1) $G_{p4} = red(G, P-\{p_4\})$
Step 9.2) $\Sigma_{p4} = FindAllMinSiphons(G_{p4}, \{p_1, p_3\})$

At Step 4 the entire set of places $\{p_1, p_2, p_3\}$ of G_{p4} is found to be a siphon. At Step 6.1 the siphon is checked to be minimal (place p_2 cannot be eliminated, and the other are constrained).

Step 9.3) $\Sigma = \{\{p_1, p_2, p_3\}, \{p_1, p_3, p_4\}\}$
Step 9.4) $P_{new} = \varnothing$, $P_{old} = \{p_1, p_3, p_4\}$

The set P_{new} is empty and the algorithm ends.

6 SIMULATION RESULTS

An experimental campaign with a C implementation of the basic algorithm described above has been performed on a test set of 180 randomly generated instances of PNs. The instances have different sizes and topological structure, since these are the most influential factors in determining the number of siphons and the effort required to find them. For the sake of simplicity, four size classes only have been considered, with $n = m = 5$, 10, 15, 20, where n is the number of places and m the number of transitions. Each size class is further divided into 9 sub-classes, whose connectivity has been determined by setting the density of input (d_i) and output (d_o) arcs to 0.25, 0.50, 0.75 in all possible combinations. Each sub-class consists of five instances.

Random PNs exhibit a sharp increase in the number of minimal siphons with respect to size, in contrast to PNs originated by real system modeling. While the latter normally presents a limited number of siphons, the former have already some hundreds with $n = m = 20$. This makes random instances particularly hard to cope with.

The CPU time required to enumerate all the minimal siphons, averaged over the PN instances, is given in Table 1, together with the average number of minimal siphons for each class size. Table 1 also presents a comparison between competing approaches: the proposed algorithm, the MIP approach described in [5] and a constructive method applying the definition of siphon. The first two algorithms are both implemented in C and the second one also employs a commercial MIP solver. The constructive method has been implemented with a MATLAB routine.

Clearly, as the number of recursive calls is bounded from below by the number of minimal siphons, the computational time strictly depends on the latter, and increases strongly with the size of the net. For $n = m = 20$, it is still of a few seconds on a AMD Athlon 4 1.24 GHz, but for $n = m = 25$ the complete minimal siphon computation takes several minutes on some instances.

Table 1. Total CPU time required for the minimal siphon computation

PN size	number of min. siphons	Total CPU time (s)		
		proposed algorithm	MIP approach	constructive algorithm
5	2.53	0.05	0.03	0.02
10	10.98	0.07	0.28	1.79
15	60.04	0.39	5.45	601.70
20	302.44	6.84	303.47	-

Another important parameter is the ratio of the CPU time to the number of siphons $\tau = T_{CPU}/|\Sigma|$; which can be interpreted as the computational time required to find a *single* siphon. It can be shown that τ increases with the PN size and the output density. This is consistent with the theoretical complexity of the minimal siphon computation procedure ($O(na)$) and with the fact that the algorithm focuses on the arcs going out of transitions. In any case, τ is extremely low with the proposed approach (some hundredths of second for the size classes explored).

7 CONCLUSIONS

In the paper, the problem of the computation of minimal siphons in a standard Petri net is addressed. First, some new theoretical results have been specifically developed. Then, a conceptual algorithm is devised, and a very first prototypical program has been generated which conforms to it. The algorithm is based on a strategy which divides the original problem into several but smaller sub-problems, that are definitely easier to solve. The sub-problems, in fact, coincide with the general problem of finding a generic siphon in a given net, with additional constraints including or excluding given places.

Future work will include the development of a full software package for the analysis of large-size PNs. A computational comparison with other siphon generating algorithms among the many different ones available in the literature will be also considered.

ACKNOWLEDGEMENTS.

The authors would like to thank student A. Benigno for her helpful experimental work.

REFERENCES

[1] Barkaoui, K., and B. Lemaire, "An Efficient Graph Theoretical Characterization of Minimal Deadlocks and Traps in Petri Nets", *Proceedings of the 10th International Conference on Application and Theory of Petri Nets*, Bonn, Germany, pp. 1-21, 1989.

[2] Barkaoui, K., and M. Minoux, "A Polynomial-Time Graph Algorithm to Decide Liveness of Some Basic Classes of Bounded Petri Nets", *Application and Theory of Petri Nets*, pp. 62-75, 1992.

[3] Boer, E.R., and T. Murata, "Generating basis siphons and traps of Petri nets using the sign incidence matrix", *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, Vol. 41, n. 4, pp. 266-271, 1994.

[4] Chu, F., and X.-L. Xie, "Deadlock Analysis of Petri Nets Using Siphons and Mathematical Programming", *IEEE Transactions On Robotics And Automation*, Vol. 13, n. 6, pp. 793-804, 1997.

[5] Cordone, R., L. Ferrarini and L. Piroddi, "Characterization of Minimal and Basis Siphons with Predicate Logic and Binary Programming", IEEE Int. Conf. On Computer Aided Control System Design, Glasgow, Scotland, pp. 193-198, 2002.

[6] Der Jeng, M., and M.Y. Peng, "Generating minimal siphons and traps for Petri nets", *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 4, pp. 2996-2999, 1996.

[7] Der Jeng, M., and M.Y. Peng, "Petri nets liveness analysis by minimal siphons", *Proceedings of 6th International Conference on Emerging Technologies and Factory Automation, ETFA '97*, pp. 315-320, 1997.

[8] Ezpeleta, J., and J.M. Couvreur, "A new technique for finding a generating family of siphons, traps and st-components. Application to colored Petri nets", *Proc. of 12th Conference on Applications and Theory of Petri Nets*, pp. 126-147, 1991.

[9] Kinuyama, M., and T. Murata, "Generating siphons and traps by Petri net representation of logic equations", *Proceedings of 2th Conference of the Net Theory SIG-IECE*, pp. 93-100, 1986.

[10] Lautenbach, K., "Linear algebraic calculation of deadlocks and traps", in *Concurrency and Nets – Advances in Petri Nets*, Voss, Genrich and Rozenberg, Eds., New York: Springer-Verlag, pp. 315-336, 1987.

[11] Minoux, M., and K. Barkaoui, "Deadlocks and traps in Petri nets as Horn-satisfiability solutions and some related polynomially solvable problems", *Discrete Applied Mathematics*, Vol. 29, pp. 195-210, 1990.

[12] Murata, T., "Petri nets: properties, analysis and application", *Proceedings of the IEEE*, Vol. 77, n. 4, pp. 541-580, 1989.

[13] Reisig, W., *Petri Nets: an Introduction*, Springer Verlag, 1982.