A Boolean Model for Enumerating Minimal Siphons and Traps in Petri nets

Faten Nabli, Francois Fages, Thierry Martinez, and Sylvain Soliman

EPI Contraintes, Inria Paris-Rocquencourt, France

Abstract. Petri nets are a simple formalism for modeling concurrent computation. Recently, they have emerged as a promising tool for modeling and analyzing biochemical interaction networks, bridging the gap between purely qualitative and quantitative models. Biological networks can indeed be large and complex, which makes their study difficult and computationally challenging. In this paper, we focus on two structural properties of Petri nets, siphons and traps, that bring us information about the persistence of some molecular species. We present two methods for enumerating all minimal siphons and traps of a Petri net by iterating the resolution of Boolean satisfiability problems executed with either a SAT solver or a CLP(B) program. We compare the performances of these methods with respect to a state-of-the-art algorithm from the Petri net community. On a benchmark with 80 Petri nets from the Petri-web database and 403 Petri nets from curated biological models of the Biomodels database, we show that miniSAT and CLP(B) solvers are overall both faster by two orders of magnitude with respect to the dedicated algorithm. Furthermore, we analyse why these programs perform so well on even very large biological models and show the existence of hard instances in Petri nets with unbounded degrees.

1 Introduction

Petri nets were introduced in the 60’s as a simple formalism for describing and studying information processing systems that are characterized as being concurrent, asynchronous, non-deterministic and possibly distributed [21].

The use of Petri nets for representing biochemical reaction models, by mapping molecular species to places and reactions to transitions, was introduced quite late in [22], together with some Petri net concepts and tools for the analysis of metabolic networks [28]. In [21], a Constraint Logic Program over finite domains (CLP(FD)) is proposed for computing place invariants, which in turn provides structural conservation laws that can be used to reduce the dimension of the Ordinary Differential Equations (ODE) associated to a biochemical reaction model.

In this paper, we consider the Petri net concepts of siphons and traps. A siphon is a set of places that, once it is unmarked, remains so. A trap is a set of places that, once it is marked, can never loose all its tokens. Thus, siphons
and traps have opposing effects on the token distribution in a Petri net. These structural properties provide sufficient conditions for reachability (whether the system can reach a given state) and liveness (freedom of deadlocks) properties. It is proved that in order to be live, it is necessary that each siphon remains marked. Otherwise (i.e. once it is empty), transitions having their input places in a siphon can not be live. One way to keep each siphon marked is to have a marked trap inside it. In fact, this condition is necessary and sufficient for a free-choice net to be live \[2\]. Mixed integer linear programs have been proposed in \[14\] and a state-of-the-art algorithm from the Petri net community has been described later in \[9\] to compute minimal sets of siphons and traps in Petri nets.

In this article, we present a simple Boolean model capturing these notions and two methods for enumerating the set of all minimal siphons and traps of a Petri net. The first method iterates the resolution of the Boolean model executed with a SAT solver while the second proceeds by backtracking with a CLP(B) program.

On a benchmark composed of the 80 Petri nets of Petriweb \[1\] and the 403 curated biological models of the biomodels.net \[2\] repository \[16\], we show that miniSAT and CLP(B) solvers are both faster by two orders of magnitude than the dedicated algorithms and can in fact solve all instances. Furthermore, we analyse why these programs perform so well on even very large biological models and show the existence of hard instances in Petri nets with unbounded degrees.

## 2 Preliminaries

### 2.1 Petri nets

A Petri net graph $PN = (P,T,W)$ is a weighted bipartite directed graph, where $P$ is a finite set of vertices called places, $T$ is a finite set of vertices (disjoint from $P$) called transitions and $W : ((P \times T) \cup (T \times P)) \rightarrow \mathbb{N}$ represents a set of directed arcs weighted by non-negative integers (the weight zero represents the absence of arc). Places are graphically represented by circles and transitions by boxes. Unlabeled edges are implicitly labeled with weight 1. A marking for a Petri net graph is a mapping $m : P \rightarrow \mathbb{N}$ which assigns a number of tokens to each place. A place $p$ is marked by a marking $m$ iff $m(p) > 0$. A subset $S \subseteq P$ is marked by $m$ iff at least one place in $S$ is marked by $m$. A Petri net is a 4-tuple $(P,T,W,m_0)$ where $(P,T,W)$ is a Petri net graph and $m_0$ is an initial marking.

The set of predecessors (resp. successors) of a transition $t \in T$ is the set of places $\bullet t = \{ p \in P \mid W(p,t) > 0 \}$ (resp. $t^* = \{ p \in P \mid W(t,p) > 0 \}$). Similarly, the set of predecessors (resp. successors) of a place $p \in P$ is the set of transitions $\bullet p = \{ t \in T \mid W(t,p) > 0 \}$ (resp. $p^* = \{ t \in T \mid W(p,t) > 0 \}$).

For every two markings $m, m' : P \rightarrow \mathbb{N}$ and every transition $t \in T$, there is a transition step $m \xrightarrow{t} m'$, if for all $p \in P$, $m(p) \geq W(p,t)$ and $m'(p) = m(p) - W(p,t)$.
Fig. 1. Petri net associated to the biochemical reaction model of Example 1 displayed here with an arbitrary marking that enables the transition $t_1$.

$m(p) - W(p, t) + W(t, p)$. This notation extends to sequence of transitions $\sigma = (t_0 \ldots t_n)$ by writing $m \xrightarrow{\sigma} m'$ if $m \xrightarrow{t_0} m_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{n-1}} m_n \xrightarrow{t_n} m'$ for some markings $m_1, \ldots, m_n$.

The classical Petri net view of a reaction model is to associate biochemical species to places and biochemical reactions to transitions.

**Example 1.** The system known as Michaelis-Menten enzymatic reactions can be represented by the Petri net depicted in Figure 1. It consists of three enzymatic reactions that take place in two discrete steps: the first involves reversible formation of a complex $(AE)$ between the enzyme $(E)$ and substrate $(A)$ and the second step involves breakdown of the $(AE)$ to form product $(B)$ and to regenerate the enzyme.

$$A + E \rightleftharpoons AE \rightarrow B + E$$

### 2.2 Siphons and Traps

Let $PN = (P, T, W)$ be a Petri net graph.

**Definition 1.** A **trap** is a non-empty set of places $P' \subseteq P$ whose successors are also predecessors, $P' \subseteq \bullet P'$.

A **siphon** is a non-empty set of places $P' \subseteq P$ whose predecessors are also successors: $P' \subseteq \bullet P'$.

A siphon (resp. a trap) is proper if its predecessor set is strictly included in its successor set, $P' \subseteq \bullet P'$ (resp. $P' \subseteq \bullet P'$).

A siphon (resp. a trap) is minimal if it does not contain any other siphon (resp. trap).

It is worth remarking that a siphon in $PN$ is a trap in the dual Petri net graph, obtained by reversing the direction of all arcs in $PN$. Note also that since predecessors and successors of an union are the union of predecessors (resp. successors), the union of two siphons (resp. traps) is a siphon (resp. a trap).
Fig. 2. Petri net graph of Example 2.

Example 2. In the Petri net graph depicted in Figure 2, \{A, B\} is a minimal proper siphon, since \(^*\{A, B\} = \{r_1, r_2\} \subset \{^*A, B\} = \{r_1, r_2, r_3\}\. \{C, D\} is a minimal proper trap, since \{C, D\} \subset \{^*C, D\} = \{r_3, r_4, r_5\}.

The following propositions show that traps and siphons provide a structural characterization of some particular dynamical properties on markings.

Proposition 1. \[^2\] For every subset \(P' \subseteq P\) of places, \(P'\) is a trap if and only if for any marking \(m \in N_P\) with \(m_p \geq 1\) for some place \(p \in P'\), and any marking \(m' \in N_P\) such that \(m \xrightarrow{\sigma} m'\) for some sequence \(\sigma\) of transitions, there exists a place \(p' \in P'\) such that \(m'_p \geq 1\).

Proposition 2. \[^2\] For every subset \(P' \subseteq P\) of places, \(P'\) is a siphon if and only if for any marking \(m \in N_P\) with \(m_p = 0\) for all \(p \in P'\), and any marking \(m' \in N_P\) such that \(m \xrightarrow{\sigma} m'\) for some sequence \(\sigma\) of transitions, we have \(m'_p = 0\) for all \(p' \in P'\).

Although siphons and traps are stable under union, it is worth noting that minimal siphons do not form a generating set of all siphons. A siphon is called a basis siphon if it can not be represented as a union of other siphons \[^3\]. Obviously, a minimal siphon is also a basis siphon, however, not all basis siphons are minimal. For instance, in Example 2, there are two basis siphons, \{A, B\} and \{A, B, C, D\}, but only the former is minimal, the latter cannot be obtained by union of minimal siphons.

2.3 Application to Deadlock Detection

One reason to consider minimal siphons is that they provide a sufficient condition for the non-existence of deadlocks.

It has been shown indeed that in a deadlocked Petri net (i.e., where no transition can fire), all unmarked places form a siphon \[^4\]. The siphon-based approach for deadlock detection checks if the net contains a proper siphon that can become unmarked by some firing sequence. A proper siphon does not become unmarked if it contains an initially marked trap. If such a siphon is identified, the initial marking is modified by the firing sequence and the check continues for the remaining siphons until a deadlock is identified, or until no further progress can be done. Considering only the set of minimal siphons is sufficient because if any
siphon becomes unmarked during the analysis, then at least one of the minimal siphons must be unmarked.

The relevance of siphons and traps for other liveness properties is summarized in [11].

2.4 Complexity

Deciding whether a Petri net contains a siphon or a trap and exhibiting one if it exists is polynomial [5]. However, the decision problem of the existence of a minimal siphon containing a given place is NP-hard [26].

Furthermore, there can be an exponential number of minimal siphons and traps in a Petri net, as shown by the following:

Example 3. In the Petri net depicted in Figure 3 defined by the transitions: $A_1 + B_1 \rightarrow A_2 + B_2$, $A_2 + B_2 \rightarrow A_3 + B_3$, \ldots, $A_n + B_n \rightarrow A_1 + B_1$, there are $2^n$ minimal siphons and $2^n$ minimal traps, each one including either $A_i$ or $B_i$ but not both of them, for all $i$'s.

2.5 Application to Systems Biology

One example of the relevance of traps and siphons in biology was given in [28] for the analysis of the potato plant that produces starch and accumulates it in the potato tubers during growth, while starch is consumed after the tubers are deposited after the harvest. The starch and several of its precursors then form traps in the reaction net during growth, while starch and possible intermediates of degradation form siphons after the harvest.

The underlying Petri net is shown in Figure 4, where $G_1$ stands for glucose-1-phosphate, $G_u$ is UDP-glucose, $S$ is the starch, $I$ stands for intermediary species and $P_1$ and $P_2$ represent external metabolites [25]. In this model, either the branch producing starch ($t_3$ and $t_4$) or the branch consuming it ($t_5$ and $t_6$) is operative. Two Petri nets are derived from this model: one Petri net where $t_5$...
Fig. 4. Petri net graph modeling the growth metabolism of the potato plant [28].

and $t_6$ are removed (in this Petri net, $t_3$ and $t_4$ are said to be operative) and one Petri net where $t_3$ and $t_4$ are removed.

It can be easily observed that the set $\{G_u, S\}$ is a trap when $t_3$ and $t_4$ are operative: once a token arrives in $S$, no transition can be fired and the token remains there independently of the evolution of the system. Dually, $\{S, I\}$ is a siphon when $t_5$ and $t_6$ are operative: once the last token is consumed from $S$ and $I$, no transition can generate a new token in these places, so they remain empty.

In most cells containing starch, starch and specific predecessors form traps, whereas starch and specific successors form siphons. This provides a very simple explanation for the fact that either the branch producing starch or the branch degrading it is operative. This is realized by complete inhibition of the appropriate enzymes by the gene regulatory network.

Another interesting example, also from [28], deals with the analysis of the role of the triosephosphate isomerase (TPI) in Trypanosoma brucei metabolism by detecting solely siphons and traps. At the beginning, Helfert et al. [2] supposed that glycolysis could proceed without TPI. But unexpected results where all system fluxes (Pyruvate, Glycerol) decrease were found so that the authors built a kinetic model for explaining that phenomenon. Then a purely structural explanation for the necessary presence of TPI in glycolysis and glycerol production was provided in [28] by simply considering the presence of siphons and traps in the model.

3 Boolean Model

In the literature, many algorithms have been proposed to compute minimal siphons and traps of Petri nets. Since a siphon in a Petri net $N$ is a trap of the dual net $N'$, it is enough to focus on siphons, the traps are obtained by duality. Some algorithms are based on linear programming [4], Horn clause satisfaction [3,7] or algebraic approaches [30]. More recent state-of-the-art methods are presented in [5,6].
Here we present two Boolean methods for enumerating minimal siphons. First, siphons can be straightforwardly characterized with a boolean model representing the belonging or not of each place to the siphon. For a Petri net with \( n \) places and \( m \) transitions, a siphon \( S \) is a set of places whose predecessors are also successors. \( S \) can be represented with a vector \( V \) of \( \{0, 1\}^n \) such that for all \( i \in \{1, 2, \ldots, n\} \), \( V_i = 1 \) if and only if \( p_i \in S \). The siphon constraint can then be formulated as

\[
\forall i, V_i = \Rightarrow \bullet p_i \subseteq \left( \bigcup_{V_j=1} \{p_j\} \right)^*
\]

which is equivalent to

\[
\forall i, V_i = \Rightarrow (\forall t \in T, t \in \bullet p_i \Rightarrow t \in \left( \bigcup_{V_j=1} \{p_j\} \right)^*)
\]

which is equivalent to

\[
\forall i, V_i = \Rightarrow (\forall t \in T, t \in \bullet p_i \Rightarrow \exists p_j \in \bullet t, V_j = )
\]

which can be rewritten in clausal form as:

\[
\forall i, V_i = \Rightarrow \bigwedge_{t \in \bullet p_i} \left( \bigvee_{p_j \in \bullet t} V_j = \right)
\]

To exclude the case of the empty set, the following constraint is added:

\[
\bigvee_i V_i =
\]

These clauses are Horn-dual clauses (i.e. clauses with at most one negative literal). They are trivially satisfied by taking all variables true.

Second, the enumeration of all minimal siphons (w.r.t. set inclusion) can be ensured by a search strategy and the addition of new Boolean constraints during search. One strategy is to find siphons in set inclusion order, and to add a new constraint each time a siphon \( S \) is found to disallow any superset of this siphon to be found in the continuation of the search. It is worth remarking that this clause is not the dual of a Horn clause. The whole clauses are thus now non-Horn.

In a previous approach based on Constraint Logic Programming [20], the enumeration by set inclusion order was ensured by labeling a cardinality variable in increasing order. Labeling directly on the Boolean variables, with increasing value selection (first 0, then 1), reveals however much more efficient and in fact easier to enforce. The following proposition shows that this strategy correctly finds siphons in set inclusion order.
Proposition 3. Given a binary tree such that, in each node instantiating a variable \(X\), the left sub-edge posts the constraint \(X = 0\) and the right sub-edge posts the constraint \(X = 1\), then for all distinct leaves \(A\) and \(B\), leaf \(A\) is on the left of leaf \(B\) only if the set represented by \(B\) is not included in the set represented by \(A\) (that is to say, there exists a variable \(X\) such that \(X_B > X_A\), where \(X_A\) and \(X_B\) denote the values instantiated to \(X\) in the paths leading to \(A\) and \(B\) respectively).

Proof. \(A\) and \(B\) have a least common ancestor node instantiating a variable \(X\). If leaf \(A\) is on the left of leaf \(B\), the sub-edge leading to \(A\) is the left one, with the constraint \(X = 0\) and the sub-edge leading to \(B\) is the right one, with the constraint \(X = 1\), therefore \(X_B > X_A\).

In a post-processing phase, the computed set of minimal siphons can be filtered for only keeping the minimal siphons that contain a given set of places, and hence solve the above mentioned NP-hard decision problem. It is worth remarking that posting the inclusion of the selected places first would not ensure that the siphons found are indeed minimal w.r.t. set inclusion.

4 Boolean Algorithms

This section describes two implementations of the above model and search strategy, one using an iterated SAT procedure and the other based on Constraint Logic Programming with Boolean constraints.

4.1 Iterated SAT Algorithm

The Boolean model can be directly interpreted using a SAT solver to check the existence of a siphon or trap. We use sat4j\(^3\), an efficient library of SAT solvers in Java for Boolean satisfaction and optimization. It includes an implementation of the MiniSAT algorithm in Java.

The example of the enzymatic reaction of example 1 is encoded as follows: each line is a space-separated list of variables representing a clause; a positive value means that corresponding variable is under positive form (so 2 means \(V_2\)), and a negative value means the negation of that variable (so \(-3\) means \(-V_3\)). In this example, variables \(-2, -3\) correspond to \(E, A\), \(AE\) and \(B\). In the first iteration, the problem amounts to solve the following encoding of Horn-dual clauses:

\[-2 3
-3 2
- 3
- 3
-4 3\]

\(^3\) http://www.sat4j.org/
The problem is satisfied with the values: −1, 2, 3, −4 which means that \( \{A, AE\} \)
is a minimal siphon.

To ensure minimality, the (non Horn-dual) clause \(-2 -3\) is added and the program iterates another time. The problem is satisfied with \( , -2, 3, -4\), meaning that \( \{E, AE\} \) is also a minimal siphon. A new clause is added stating that either \( E \) or \( AE \) does not belong to the siphon and no more variable assignment can satisfy the problem.

Therefore, this model contains 2 minimal siphons: \( \{A, AE\} \) and \( \{E, AE\} \). The enzyme \( E \) is a catalyst protein for the transformation of the substrate \( E \) in a product \( B \). Such a catalyst increases the rate of the reaction but is conserved in the reaction.

### 4.2 Backtrack Replay CLP(B) Algorithm

The search for siphons can also be implemented with a Constraint Logic Program with Boolean constraints (CLP(B)). We use GNU-Prolog\(^4\)\[8\] for its efficient low-level implementation of Boolean constraint propagators.

The enumeration strategy is a variation of branch-and-bound, where the search is restarted to find a non-superset siphon each time a new siphon is found. We tried two variants of the branch-and-bound: with restart from scratch and by backtracking.

In the branch-and-bound with restart method, it is essential to choose a variable selection strategy which ensures diversity. Indeed, an enumeration method with a fixed variable order accumulates failures by always trying to enumerate the same sets first and these failures are only lately pruned by the non-superset constraints. As a consequence, the developed search tree gets more and more dense after each iteration since the previous forbidden sets are repeatedly tried again. This phenomenon does not exist in SAT solvers thanks to no-good recording. In CLP, this problem can be compensated for however, by using a random selection strategy for variables. This provides a good diversity and performs much better than any uniform heuristics.

However, branch-and-bound by backtracking gives better performance when care is taken for posting the non-superset constraint only once, since reposting it at each backtrack step proved to be inefficient. Our backtrack replay strategy is implemented as follows:

1. Each time a siphon is found, the path leading to this solution is memorized,
2. Then the search is fully backtracked in order to add to the model the new non-superset constraint,
3. And then the memorized path is rolled back to continue the search at the point it was stopped.

Figure 5, generated with CLPGUI\(^5\)\[9\], depicts the search tree that is developed for enumerating the 64 minimal siphons of a biological model of 51 species.

\(^4\)http://www.gprolog.org/
\(^5\)http://contraintes.inria.fr/~fages/CLPGUI
Fig. 5. Search tree developed with the backtrack replay strategy for enumerating the 64 minimal siphons of model 239 of biomodels.net (described in Section 5.1).

and 72 reactions. Each sub-tree immediately connected to the root corresponds to the replay of the path with a minimality constraint added. It is remarkable that with the backtrack replay strategy, very few backtracking steps are necessary to search for all solutions.

5 Evaluation

5.1 Benchmark

Petriweb Our first benchmark of Petri nets is Petriweb [10], a benchmark of 80 Petri nets from the Petri net community. The most difficult instances of this benchmark come from case studies in process refinement, namely problems 454, 47 and 6.

Biomodels.net We also consider the Petri nets associated to biochemical reaction models of the biomodels.net repository of 403 models [16] and some other complex biochemical models. The most difficult models are the following ones:

– Kohn’s map of the mammalian cell cycle control [14], a model of 50 species and 775 reactions;
– Model BIOM000000075 of biomodels.net, a model of 8 species and 4 reactions involved in ErbB signaling;
– Model BIOM0000000205 of biomodels.net, a model of 4 species and 3 reactions involved in the regulation of EGFR endocytosis and EGFR-ERK signaling by endophilin-mediated RhoA-EGFR crosstalk;
– Model BIOM000000023 , a core model of 5 species and 72 reactions representing the glucose-stimulated insulin secretion of pancreatic beta cells.

5.2 Results and Comparison

In this section, we compare the two Boolean methods described in the previous section with the state-of-the-art dedicated algorithm of [6]. This algorithm uses a
recursive problem partitioning procedure to reduce the original search problem to multiple simpler search subproblems. Each subproblem has specific additional place constraints with respect to the original problem. This algorithm can be applied to enumerate minimal siphons, place-minimal siphons, or even siphons that are minimal with respect to a given subset of places.

Table 1 presents the CPU times for enumerating all minimal siphons of the Petri nets in Petriweb and biomodels.net. All times are in milliseconds and have been obtained on a PC with an Intel Core processor 2.20 GHz and 8 GB of memory. For each benchmark, we provide the total number of models, the minimal, maximal and average numbers of siphons and the total computation time for enumerating all of them.

Surprisingly, but happily, on all these practical instances, except one instance detailed below, the SAT and CLP(B) programs solve the minimal siphon enumeration problem, in less than one millisecond in average, with a better performance for the CLP(B) program over the SAT solver, and by two orders of magnitude over the dedicated algorithm.

However, one particular model, number 175 in biomodels.net, was excluded from this table because its computational time is very high. Table 2 presents the performance figures obtained on this model and on the three other hardest instances for which we also provide the number of places and transitions. On these hard instances, the SAT solver is faster than the CLP(B) program by one to two orders of magnitude, and is the only algorithm to solve the problem for model 175, in 137 seconds.

That model 175 represents a quantitative model that relates EGF and HRG stimulation of the ErbB receptors to ERK and AKT activation in MCF-7 breast cancer cells. This is the first model to take into account all four ErbB receptors, simultaneous stimulation with two ligands, and both the ERK and AKT pathways. Previous models of ErbB (e.g. the model developed in [23]) were limited to a single ErbB because of combinatorial complexity. It is well known that

<table>
<thead>
<tr>
<th>Database</th>
<th># model</th>
<th># siphons (min–max (avg.))</th>
<th>siphons size (min–max (avg.))</th>
<th>total time dedicated SAT GNU Prolog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biomodels.net</td>
<td>403</td>
<td>0–64 (4.21)</td>
<td>1–413 (3.10)</td>
<td>19734 611 195</td>
</tr>
<tr>
<td>Petriweb</td>
<td>80</td>
<td>0–11 (2.85)</td>
<td>0–7 (2.03)</td>
<td>2325 156 6</td>
</tr>
</tbody>
</table>

**Table 1.** Performance on the whole benchmark.

<table>
<thead>
<tr>
<th>Model</th>
<th># siphons</th>
<th># places</th>
<th># transitions</th>
<th>dedicated algorithm</th>
<th>sat</th>
<th>GNU Prolog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kohn’s map of cell cycle</td>
<td>81</td>
<td>509</td>
<td>775</td>
<td>28</td>
<td>1</td>
<td>221</td>
</tr>
<tr>
<td>BIOMD000000175</td>
<td>3042</td>
<td>118</td>
<td>194</td>
<td>∞</td>
<td>137000</td>
<td>∞</td>
</tr>
<tr>
<td>BIOMD000000205</td>
<td>32</td>
<td>194</td>
<td>313</td>
<td>21</td>
<td>1</td>
<td>34</td>
</tr>
<tr>
<td>BIOMD000000239</td>
<td>64</td>
<td>51</td>
<td>72</td>
<td>2980</td>
<td>1</td>
<td>22</td>
</tr>
</tbody>
</table>

**Table 2.** Performance on the hardest instances.
Fig. 6. Petri net graphs considered for the reduction of 3-SAT to the existence of a minimal siphon containing place $q_0$.

the ErbB signaling network is highly connected and indeed the underlying Petri net contains the highest number of arcs of the biomodels.net repository.

5.3 Hard instances

MiniSAT and CLP(B) outperform the specialized algorithm by at least one order of magnitude and the computation time is extremely short on our practical examples. Even if the model is quite large, e.g. for Kohn’s map of the cell cycle control with 50 species and 775 reactions, the computation time for enumerating its 8 minimal siphons is astonishingly short: one milisecond only. However, this enumeration of all minimal siphons solves the decision problem of the existence of a minimal siphon containing a given set of places which has been proved NP-hard by reduction of 3-SAT in [27], and the question is: why the existing benchmarks from systems biology and petriweb are so easy?

We can provide some hints of explanation by considering the well-known phase transition phenomenon in 3-SAT. The probability that a random 3-SAT problem is satisfiable has been shown to undergo a sharp phase transition as the ratio $\alpha$ of the number of clauses over the number of variables crosses the critical value of about 4.26 [18,7], going from satisfiability to unsatisfiability with probability one when the number of variables grows to the infinity.
The reduction of three-satisfiability (3-SAT) to the problem of existence of a minimal siphon containing a given place has been shown in [27] with the Petri net structure illustrated in Figure 6. It is worth noticing that in this encoding, the Petri net has a maximum place indegree (for $q_0$) which is linear in the number of clauses, and a maximum place outdegree (for $t_0$) which is linear in the number of variables.

Not surprisingly, this family of Petri nets provides a hard benchmark for enumerating minimal siphons. Table 3 contains experimental results on these Petri nets associated to random 3-SAT problems. The table gives the number of minimal siphons and the time to compute them with a timeout of 60 seconds. The table also provides information concerning both the 3-SAT problem and its corresponding Petri net. For each 3-SAT problem, we provide the number of Boolean variables, the number of clauses and the ratio $\alpha$. For the corresponding Petri net, we provide the number of places, the number of transitions and the density (ratio of the number of transitions over the number of places). The computation time of all minimal siphons as a function of $\alpha$, the density of the initial 3-SAT problem, is represented in Figure 7.

The reason for the timeout obtained for all 3-SAT problems of density below the threshold value 4.26 is that for small values of $\alpha$, the clause is satisfiable with an exponential number of valuations which gives rise to an exponential number of minimal siphons to compute. On the other hand, for values of $\alpha$ above

---

Table 3. Computational results for the enumeration of minimal siphons in Petri nets encoding 3-SAT.

<table>
<thead>
<tr>
<th>model</th>
<th># siphons</th>
<th>Petri net view</th>
<th>3-SAT view</th>
<th>time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># places</td>
<td># transitions</td>
<td>density</td>
<td># variables</td>
</tr>
<tr>
<td>pn0.2.xml</td>
<td>&gt;129567</td>
<td>801</td>
<td>441</td>
<td>0.56</td>
</tr>
<tr>
<td>pn0.6.xml</td>
<td>&gt;32392</td>
<td>801</td>
<td>521</td>
<td>0.65</td>
</tr>
<tr>
<td>pn1.xml</td>
<td>&gt;1075</td>
<td>801</td>
<td>601</td>
<td>0.751</td>
</tr>
<tr>
<td>pn2.xml</td>
<td>&gt;74462</td>
<td>801</td>
<td>801</td>
<td>1</td>
</tr>
<tr>
<td>pn3.xml</td>
<td>&gt;63816</td>
<td>801</td>
<td>1001</td>
<td>1.24</td>
</tr>
<tr>
<td>pn4.xml</td>
<td>&gt;59827</td>
<td>801</td>
<td>1201</td>
<td>1.49</td>
</tr>
<tr>
<td>pn4.2.xml</td>
<td>&gt;41415</td>
<td>801</td>
<td>1241</td>
<td>1.54</td>
</tr>
<tr>
<td>pn4.4.xml</td>
<td>&gt;200</td>
<td>801</td>
<td>1281</td>
<td>1.59</td>
</tr>
<tr>
<td>pn4.6.xml</td>
<td>&gt;200</td>
<td>801</td>
<td>1321</td>
<td>1.64</td>
</tr>
<tr>
<td>pn5.xml</td>
<td>&gt;200</td>
<td>801</td>
<td>1401</td>
<td>1.74</td>
</tr>
<tr>
<td>pn6.xml</td>
<td>&gt;200</td>
<td>801</td>
<td>1601</td>
<td>1.99</td>
</tr>
<tr>
<td>pn7.xml</td>
<td>&gt;200</td>
<td>801</td>
<td>1801</td>
<td>2.24</td>
</tr>
<tr>
<td>pn8.xml</td>
<td>&gt;200</td>
<td>801</td>
<td>2001</td>
<td>2.49</td>
</tr>
<tr>
<td>pn9.xml</td>
<td>&gt;200</td>
<td>801</td>
<td>2201</td>
<td>2.74</td>
</tr>
<tr>
<td>pn10.xml</td>
<td>&gt;200</td>
<td>801</td>
<td>2401</td>
<td>2.99</td>
</tr>
</tbody>
</table>

All benchmarks of this section are available at [http://contraintes.inria.fr/~nabli/indexhardinstances.html](http://contraintes.inria.fr/~nabli/indexhardinstances.html)
Fig. 7. CPU time for computing all minimal siphons in Petri nets encoding 3-SAT problems of density ranging from 0 to 0.1 with a time-out of 2 seconds.

<table>
<thead>
<tr>
<th>nb places</th>
<th>nb transitions</th>
<th>density</th>
<th>nb siphons</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>random-degree-pn1</td>
<td>801</td>
<td>80</td>
<td>0.099875156</td>
<td>&gt;77918</td>
</tr>
<tr>
<td>random-degree-pn2</td>
<td>801</td>
<td>320</td>
<td>0.399500624</td>
<td>&gt;69246</td>
</tr>
<tr>
<td>random-degree-pn3</td>
<td>801</td>
<td>560</td>
<td>0.699126092</td>
<td>&gt;45782</td>
</tr>
<tr>
<td>random-degree-pn4</td>
<td>801</td>
<td>801</td>
<td>1</td>
<td>&gt;28285</td>
</tr>
<tr>
<td>random-degree-pn5</td>
<td>801</td>
<td>1041</td>
<td>1.299625468</td>
<td>0</td>
</tr>
<tr>
<td>random-degree-pn6</td>
<td>801</td>
<td>1281</td>
<td>1.599250936</td>
<td>0</td>
</tr>
<tr>
<td>random-degree-pn7</td>
<td>801</td>
<td>1521</td>
<td>1.898876404</td>
<td>0</td>
</tr>
<tr>
<td>random-degree-pn8</td>
<td>801</td>
<td>1762</td>
<td>2.199750312</td>
<td>0</td>
</tr>
<tr>
<td>random-degree-pn9</td>
<td>801</td>
<td>2242</td>
<td>2.790001248</td>
<td>0</td>
</tr>
<tr>
<td>bounded-degree-pn1</td>
<td>801</td>
<td>80</td>
<td>0.099875</td>
<td>377</td>
</tr>
<tr>
<td>bounded-degree-pn2</td>
<td>801</td>
<td>320</td>
<td>0.399501</td>
<td>250</td>
</tr>
<tr>
<td>bounded-degree-pn3</td>
<td>801</td>
<td>560</td>
<td>0.699126</td>
<td>146</td>
</tr>
<tr>
<td>bounded-degree-pn4</td>
<td>801</td>
<td>801</td>
<td>1</td>
<td>66</td>
</tr>
<tr>
<td>bounded-degree-pn5</td>
<td>801</td>
<td>1041</td>
<td>1.299625</td>
<td>29</td>
</tr>
<tr>
<td>bounded-degree-pn6</td>
<td>801</td>
<td>1281</td>
<td>1.599251</td>
<td>13</td>
</tr>
<tr>
<td>bounded-degree-pn7</td>
<td>801</td>
<td>1521</td>
<td>1.898876</td>
<td>4</td>
</tr>
<tr>
<td>bounded-degree-pn8</td>
<td>801</td>
<td>1762</td>
<td>2.199750312</td>
<td>1</td>
</tr>
<tr>
<td>bounded-degree-pn9</td>
<td>801</td>
<td>2242</td>
<td>2.790001248</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4. Computational results for the enumeration of minimal siphons in random Petri nets with varying density and linear versus bounded degrees.

the threshold, the clause are unsatisfiable and there is indeed no minimal siphon containing $q_0$ (only the 200 minimal siphons without $q_0$ are computed).

Now, Table 3 shows that similar bad performance figures are obtained with randomly generated Petri nets with a number of in and out degrees that is linear in the number of places and transitions, while on random Petri nets with a bounded degree (less than 5), the enumeration of minimal siphons is easy. This is the situation encountered in our practical application. As shown in Table 5, the Petri nets associated to the biochemical reaction models of biomodels.net have small in and out degrees for places and transitions even in very large models. Model 75 mentioned in Section 5.2 appears as an exception combining a large size with a high connectivity on places, with some species that are both the reactants of 32 reactions and the products of 3 reactions.
### Table 5.
In and out degrees for places and transitions in the Petri nets of the biomodels.net benchmark with model 175 apart.

<table>
<thead>
<tr>
<th></th>
<th>minimum</th>
<th>maximum</th>
<th>average</th>
<th>model 175</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of arcs</td>
<td>1</td>
<td>913</td>
<td>92</td>
<td>1125</td>
</tr>
<tr>
<td>Avg-indegree-places</td>
<td>0</td>
<td>8</td>
<td>1.89</td>
<td>5</td>
</tr>
<tr>
<td>Avg-indegree-transitions</td>
<td>0</td>
<td>3</td>
<td>1.06</td>
<td>3</td>
</tr>
<tr>
<td>Max-indegree-places</td>
<td>1</td>
<td>54</td>
<td>5.94</td>
<td>31</td>
</tr>
<tr>
<td>Max-indegree-transitions</td>
<td>1</td>
<td>14</td>
<td>2.72</td>
<td>11</td>
</tr>
<tr>
<td>Avg-outdegree-places</td>
<td>0</td>
<td>8</td>
<td>1.93</td>
<td>5</td>
</tr>
<tr>
<td>Avg-outdegree-transitions</td>
<td>0</td>
<td>3</td>
<td>0.99</td>
<td>3</td>
</tr>
<tr>
<td>Max-outdegree-places</td>
<td>0</td>
<td>36</td>
<td>5.53</td>
<td>32</td>
</tr>
<tr>
<td>Max-outdegree-transitions</td>
<td>1</td>
<td>14</td>
<td>2.87</td>
<td>11</td>
</tr>
</tbody>
</table>

### 6 Conclusion

Siphons and traps in Petri nets define meaningful pools of places that display a specific behavior during the dynamical evolution of a Petri net, or of a system of biochemical reactions whatever kinetic parameters are.

We have described a Boolean model for the problem of enumerating all minimal siphons in a Petri net and have compared two Boolean methods to a state-of-the-art algorithm from the Petri net community [6]. The miniSAT solver and the CLP(B) program both solve our large benchmark of real-size problems and outperform the dedicated algorithm by two orders of magnitude. On the benchmark of 403 biological models in biomodels.net, the Boolean method for enumerating all minimal siphons using miniSAT is very efficient. It also scales very well in the size of the net. The CLP(B) program also solves all but one instances of the benchmark, with a better performance than miniSAT in average, but does not scale-up as well to large size models like Kohn’s map with 50 species and 775 reactions.

The surprising efficiency of the miniSAT and CLP(B) methods for solving the practical instances of this NP-hard problem has been analyzed in connection to the well-known phase transition phenomenon in 3-SAT, and to the fact that the degree of Petri nets associated to even very large models of several hundreds of biochemical species and reactions remains limited to small values in practice. This explains why these Boolean methods perform so well in the practical context of systems biology applications.

These results militate for the analysis of biochemical networks with Petri net concepts and Constraint Programming tools.

**Acknowledgment** This work is supported by the French OSEO project Biointelligence.
References


