BIOINFORMATICS ORIGINAL PAPER

Vol. 28 no. 19 2012, pages 2474–2483
doi:10.1093/bioinformatics/bts423

Systems biology

Advance Access publication July 10, 2012

Algorithms and complexity of enumerating minimal precursor sets in genome-wide metabolic networks

Vicente Acuña[1,2,3,†,*], Paulo Vieira Milreu[1,3,†,*], Ludovic Cottret[4], Alberto Marchetti-Spaccamela[5], Leen Stougie[6] and Marie-France Sagot[1,3,*]

[1]Université de Lyon, F-69000 Lyon, Université Lyon 1, CNRS, UMR5558, Laboratoire de Biométrie et Biologie Evolutive, F-69622 Villeurbanne, France, [2]Mathomics, Center for Genome Regulation (Fondap 15090007) and Center for Mathematical Modeling (UMI 2807 CNRS), University of Chile, Santiago, Chile, [3]INRIA Rhône-Alpes, 38330 Montbonnot Saint-Martin, France, [4]Laboratoire d'Ingénierie des Systèmes Biologiques et des Procédés (LISBP), UMR CNRS 5504, INRA 792, Toulouse, 31000, France, [5]Dip. di Informatica e Sistemistica, University of Rome La Sapienza, 00184, Rome, Italy, and [6]Department of Economics and Business Administration, VU University, 1087 HV Amsterdam and Centrum voor Wiskunde en Informatica (CWI), 1098 XG Amsterdam, The Netherlands

Associate Editor: Trey Ideker

ABSTRACT

Motivation: In the context of studying whole metabolic networks and their interaction with the environment, the following question arises: given a set of target metabolites T and a set of possible external source metabolites $\mathcal{S}$, which are the minimal subsets of $\mathcal{S}$ that are able to produce all the metabolites in T. Such subsets are called the minimal precursor sets of T. The problem is then whether we can enumerate all of them efficiently.

Results: We propose a new characterization of precursor sets as the inputs of reaction sets called *factories* and an efficient algorithm to decide if a set of sources is precursor set of T. We show proofs of hardness for the problems of finding a precursor set of minimum size and of enumerating all minimal precursor sets T. We propose two new algorithms which, despite the hardness of the enumeration problem, allow to enumerate all minimal precursor sets in networks with up to 1000 reactions.

Availability: Source code and datasets used in our benchmarks are freely available for download at http://sites.google.com/site/pitufosoftware/download.

Contact: vicente77@gmail.com, pvmilreu@gmail.com or marie-france.sagot@inria.fr

Received on February 16, 2012; revised on June 6, 2012; accepted on June 29, 2012

1 INTRODUCTION

We recently introduced the concept of a *minimal precursor set* which corresponds to a set of metabolites that an organism may obtain from its environment and that enables it to produce a set of metabolic targets of interest (see Cottret *et al.* (2008) for the initial definition of this concept). In this model, we propose a definition of precursor set which does not consider the stoichiometry of reactions. Indeed, the values given for such stoichiometry may often not be accurate. For instance, 51% of the reactions in the Kyoto Encyclopedia of Genes and Genomes (KEGG) were considered to be unbalanced in 2004 (Feist *et al.*, 2009) and solving these cases may become challenging for more complex reactions (Thiele and Palsson, 2010). To overcome this problem, we propose instead a model based only on the topology of a metabolic network, that is that considers the set of substrates and products of each reaction without considering the amounts involved. The collection of precursor sets defined should therefore be considered as potential/candidate solutions which could be confirmed or discarded *a posteriori* by other sources of information.

The method we developed in Cottret *et al.* (2008) to enumerate all minimal precursor sets for a given set of targets was then applied in Cottret *et al.* (2010a) to a relatively complex symbiotic system. In this case, the environment was represented by an insect. *Homalodisca coagulata*, which hosts within its cells two bacteria, respectively, *Baumannia cicadellinicola* and *Sulcia muelleri*. The identification of the precursor sets for the sets of metabolites each bacterium gives to the symbiotic system (host and co-resident endocytobiont), enabled to refine the analysis that had been done previously (McCutcheon and Moran, 2007) of the complementarity between the metabolisms of the two bacteria and their host. It also suggested that both *B. cicadellinicola* and *S. muelleri* might be completely independent of the metabolites output by the co-resident endocytobiont to produce the carbon backbone of the metabolites provided to the symbiotic system.

The algorithms in Cottret *et al.* (2010a) and Cottret *et al.* (2008) suffered of a memory problem due to the necessity to construct a huge tree—called the 'replacement tree'. Moreover, the enumeration procedure followed using such a tree was not the most efficient way either to enumerate all minimal precursor sets. For small networks (<250 nodes), the previous method runs in a acceptable time, but for bigger networks it usually runs out of memory.

In this article, we present new algorithms for enumerating all minimal precursor sets that address both memory requirements and time efficiency (Section 4). We also provide full proofs for the complexity results that were just indicated in Cottret *et al.* (2008) (Section 3). We use for this a simpler characterization of a

*To whom correspondence should be addressed.
†The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.

precursor set that makes the concept, and the subsequent proofs, formally easier to grasp (Section 2). Finally, we show by extensive tests that the new algorithms are indeed able to deal with much larger networks, up to 1000 reactions (Section 5).

2 DEFINITIONS AND CHARACTERIZATIONS

A metabolic network is modelled as a *directed hypergraph* $G = (\mathcal{C}, \mathcal{R})$ with $\mathcal{C}$ the set of *vertices* corresponding to metabolites (also called compounds) and $\mathcal{R}$ the set of *hyperarcs* corresponding to reactions. A directed hyperarc of a reaction $r \in \mathcal{R}$ is an ordered pair of metabolite sets $r = (Subs(r), Prod(r))$, where $Subs(r)$ is the set of substrates of r and $Prod(r)$ is the set of products of r. Reactions are supposed to be irreversible: each originally reversible reaction is replaced by two irreversible reactions of opposite direction.

We consider also a set of *sources* $\mathcal{S} \subseteq \mathcal{C}$ representing the metabolites that are *potentially* available in infinite external supply. Sources used as substrates of reactions produce other metabolites, thereby increasing the set of available ones. In addition, the set $T \subseteq \mathcal{C}$ denotes the *target set*, that is a set of metabolites that it is interesting to produce. Given a source set $\mathcal{S}$ and a target set T of metabolites, the aim is to find subsets of $\mathcal{S}$ which are able to produce all metabolites of T. We need now to formally define the meaning of: *being able to produce the target*.

When stoichiometric information is missing or not (fully) reliable, two definitions have been proposed to model this concept that can give different solutions to a particular instance.

Before comparing the two approaches, we introduce some notation. Let M be a set of metabolites of $\mathcal{C}$. We define $Reac(M)$ as the set of reactions that can be fired when the metabolites in M are present. In other words, $Reac(M) = \{r \in \mathcal{R} \mid Subs(r) \subseteq M\}$. For a given set of reactions $R \subseteq \mathcal{R}$, we define the sets $Subs(R) = \cup_{r \in R} Subs(r)$ and $Prod(R) = \cup_{r \in R} Prod(r)$.

2.1 Sequencial production of the target

The *forward propagation* of M, denoted by $Fwd(M)$, is the set of metabolites successively produced from M using the reactions of the network. Formally, $Fwd(M)$ is the result of the recursion $M_{i+1} = M \cup Prod(Reac(M_i))$ starting from $M_0 = M$ and until a fixed point is reached. For instance, in the network of Figure 1, if $M_0 = \{a, b, c\}$ then $M_1 = \{a, b, c, e\}$, $M_2 = \{a, b, c, e, h\}$ and so on until the fixed point $\{a, b, c, e, h, i, t\}$ is reached. Thus, $Fwd(\{a, b, c\}) = \{a, b, c, e, h, i, t\}$.

Romero and Karp (2001) considered a subset X of the sources $\mathcal{S}$ as a precursor set of a target T, when $T \subseteq Fwd(X)$. For instance, the set of sources $X = \{a, b, c\}$ is a precursor set of the target set $T = \{t\}$ since $Fwd(\{a, b, c\})$ contains t. This *iterative* way to calculate what is available from X may however not be enough to model some real cases. Indeed, the network could have cycles whose metabolites need to be consumed and produced *all at the same time*. For instance, in Figure 1 reactions r_1 and r_2 form a cycle that consumes metabolites c and d to produce f and g. However, $Fwd(\{c, d\}) = \{c, d\}$, that is it contains neither f nor g.

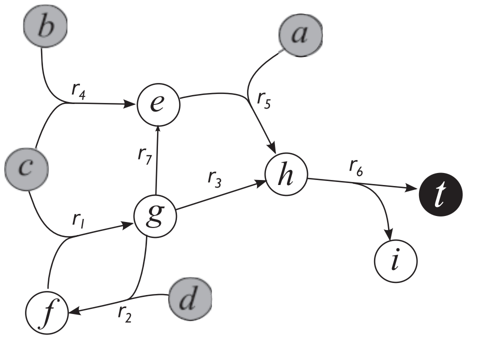

Fig. 1. A metabolic network. Nodes represent metabolites and hyperarcs represent reactions. Grey nodes are sources while the black node is the target.

2.2 Including cycles in the target production

In our work, we used the model proposed in Cottret *et al.* (2008) which defines a precursor set using a different approach. Instead of starting the propagation from a subset of the sources, the authors allow from the beginning the inclusion of other metabolites (called *internal supply*) provided that such metabolites are produced by some reaction in a future step of the forward propagation (besides production of the target). Formally, the authors define $Fwd_Z(M)$, the forward propagation of M with Z (as internal supply), as the result of the recursion $M_{i+1} = M \cup Prod(Reac(M_i \cup Z))$ starting from $M_0 = M$ and until a fixed point is reached. A subset X of the sources is a precursor set of T if T and Z are both included in $Fwd_Z(X)$. For instance, in the network of Figure 1, $Fwd_{\{f\}}(\{c, d\}) = \{c, d, f, g, e, h, i, t\}$. Thus, it produces t but also *re-produces* f to maintain the cycle working.

DEFINITION 1. *A set of sources $X \subseteq \mathcal{S}$ is a* precursor set of $T \subseteq \mathcal{C}$ *if there exists a set $Z \subseteq \mathcal{C}$ such that $T \cup Z \subseteq Fwd_Z(X)$. In this case, we say that Z is an* internal supply *of the precursor set X.*

Of course, the internal supply may be not unique for a given precursor set. In Figure 1, both $Z = \{f\}$ and $Z = \{g\}$ are internal supplies for the precursor set $X = \{c, d\}$. Observe also that any set of metabolites which is a precursor set by the Romero and Karp definition will continue being a precursor set for this definition just considering $Z = \emptyset$.

Suppose now that the target is a set of metabolites whose production we want to *avoid*. In this case, we can define the notion of a *precursor cut set* or simply *cut set*, that is a subset X of sources such that, if they are not present, then the target cannot be produced by any combination of the remaining sources. This concept has a biological application, for instance, in the case where we want a bacterium to avoid producing some given metabolite while providing it with a maximal set of resources that enables it to continue doing its other specific tasks. As an example, in Figure 1, the set $\{a, d\}$ is a cut set of $\{t\}$.

DEFINITION 2. *A set of sources $X \subseteq \mathcal{S}$ is a* cut set *of $T \subseteq \mathcal{C}$ if and only if the set $\mathcal{S} \setminus X$ is* not *a precursor set of T.*

If the target contains more than one metabolite, a cut will avoid the production of the *whole* target set but could still produce *some* of their elements (a strict subset of T). If we want to block *each* element of the target, we can modify slightly the network in the following way: given $T = \{t_1, \ldots, t_\ell\}$, we can define a new target metabolite t_{target} and reactions $r_{t_1}, \ldots, r_{t_\ell}$ with $Subs(r_{t_i}) = \{t_i\}$ and $Prod(r_{t_i}) = \{t_{\text{target}}\}$. Clearly, a cut set of the new target $T' = \{t_{\text{target}}\}$ must block the production of each metabolite in T.

2.3 New characterization using factories

We give now a simpler and more natural way to grasp the characterization of a precursor set X of T by considering, instead of metabolites, a set of reactions $F \subseteq \mathcal{R}$ that *connects* X with T. Clearly, the reactions must verify these two properties:

1. (Feasibility of reactions) Each substrate of the reactions in F is contained in X or is produced by some reaction in F;
2. (Production of target) Each metabolite in the target T (which is not in X) is produced by some reaction in F.

These two conditions can be summarized in one: $T \cup Subs(F) \subseteq X \cup Prod(F)$. In this case, we say that F is a *factory from X to T*.

Theorem 1. *A set of sources $X \subseteq \mathcal{S}$ is a* precursor set of $T \subseteq \mathcal{C}$ *if and only if there exists a factory from X to T.*

Proof. If X is a precursor set of T, there exists $Z \subseteq \mathcal{C}$ such that $T \cup Z \subseteq Fwd_Z(X) = X \cup Prod(Reac(Fwd_Z(X)))$. The set of reactions $F = Reac(Fwd_Z(X))$ is such that $T \subseteq X \cup Prod(F)$ and $Subs(F) \subseteq Fwd_Z(X) \subseteq X \cup Prod(F)$. Therefore, F is a factory from X to T. Inversely, let F be a set of reactions such that $T \cup Subs(F) \subseteq X \cup Prod(F)$. Defining $Z = Subs(F)$, we have $Fwd_Z(X) = X \cup Prod\ (Reac(Fwd_Z(X) \cup Subs(F)))$ which clearly contains $X \cup Prod(F)$. Therefore, $T \cup Z \subseteq Fwd_Z(X)$. □

In the example of Figure 1, the set $\{c, d\}$ is a precursor set of $\{t\}$, since the set of reactions $F = \{r_1, r_2, r_3, r_6\}$ is such that $\{t\} \cup \{c, d, f, g, h\} \subseteq \{c, d\} \cup \{f, g, h, i, t\}$.

3 COMPLEXITY RESULTS

Given a metabolic network $G = (\mathcal{C}, \mathcal{R})$ with $\mathcal{S} \subseteq \mathcal{C}$ a set of sources and $T \subseteq \mathcal{C}$ a set of target metabolites, we address the theoretical complexity of the following three problems:

MinimalPS(T): find a minimal precursor set $X \subseteq \mathcal{S}$ of T.
MinSizePS(T): find a minimum size precursor set $X \subseteq \mathcal{S}$ of T.
AllPS(T): enumerate all minimal precursor sets $X \subseteq \mathcal{S}$ of T.

We also consider the analogous problems where what is searched are precursor *cut* sets: MinimalPCS, MinSizePCS and AllPCS.

3.1 Finding a minimal precursor set and a minimal cut set

Given a set $X \subseteq \mathcal{S}$ of sources, we can compute the maximal set of reactions $F_{\max}$ that satisfy the first condition of the factory definition (feasibility of reactions). Thus, to decide whether X is a precursor set of a given T, we can compute $F_{\max}$ of X and check whether T is included in the products of $F_{\max}$. To obtain this set, we use the following recursion: starting from the whole set of reactions $F_0 = \mathcal{R}$, compute the set $F_{i+1} = Reac(X \cup Prod(F_i))$ until a fixed point is reached. Defining $K = max_{r \in \mathcal{R}}$ $(|Subs(r)| + |Prod(r)|)$, we have the following result.

Theorem 2. *Given a subset $X \subseteq \mathcal{S}$ of sources and a target set $T \subseteq \mathcal{C}$, we can decide in polynomial time $O(|\mathcal{C}||\mathcal{R}| + |\mathcal{R}|^2 K)$ whether X is a precursor set of T.*

Proof. We show the maximality of $F_{\max}$. Let F$'$ be another set of reactions such that $Subs(F') \subseteq X \cup Prod(F')$. Clearly, if $F' \subseteq F_i$ then $F' \subseteq Reac(X \cup Prod(F')) \subseteq Reac(X \cup Prod(F_i)) = F_{i+1}$, Since we start with $F_0 = \mathcal{R}$, we conclude that $F' \subseteq F_{\max}$.

The algorithm iterates at most $|\mathcal{R}| - |F_{\max}|$ times. Computing F_i takes $O(|\mathcal{C}| + |\mathcal{R}|K)$ time. Therefore, the running time of the whole procedure is $O(|\mathcal{C}||\mathcal{R}| + |\mathcal{R}|^2 K)$. □

This method provides also a way to find a *minimal* precursor set of T. Starting from $X = \mathcal{S}$, we successively check if removing a metabolite of X the target is still produced, maintaining in X only those that are needed to produce T. We obtain a minimal precursor set in $|\mathcal{S}|$ iterations. A similar procedure is also valid to find a minimal cut set starting from $X' = \emptyset$ and adding sources while the target is not produced. The set $X = \mathcal{S} \setminus X'$ is a minimal cut set.

Corollary 3. *Both* MinimalPS(T) *and* MinimalPCS(T) *can be solved in polynomial time $O(|\mathcal{C}||\mathcal{R}||\mathcal{S}| + |\mathcal{R}|^2|\mathcal{S}|K)$.*

3.2 Minimum size precursor set and cut set

Although finding one minimal precursor set of a target is easy, obtaining a (minimal) precursor set of *minimum size* is NP-hard. This result is proved by a reduction from the NP-complete problem HittingSet (Garey and Johnson, 1979): given a finite set of elements U and a collection of subsets $\mathcal{I} = \{I_1, \ldots, I_n\}$ of U, find a minimum cardinality subset of elements $H \subseteq U$ such that H intersects all the subsets in $\mathcal{I}$.

Theorem 4. *The problem* MinSizePS*(T) is NP-hard.*

Proof. We show hardness by proving completeness of the decision version where we ask if a precursor set of size at most k exists. Theorem 2 implies that this decision version is in NP.

We make a polynomial time reduction from the decision version of HittingSet, asking if there exists a hitting set of size at most k. Consider $H, \mathcal{I}$ and k a hitting set instance with $\mathcal{I} = \{I_1, \ldots, I_n\}$. For each element h in H, we create a vertex h in $\mathcal{C}$, and for each set I_j in $\mathcal{I}$, we create a vertex I_j in $\mathcal{C}$ (Fig. 2). We create an extra vertex t in $\mathcal{C}$. For each $h \in I_j$, we create in $\mathcal{R}$ an arc r_{hj} going from h to I_j. Moreover, we create the hyperarc r_t having $Subs(r_t) = \{I_1, \ldots, I_n\}$ and $Prod(r_t) = \{t\}$. We define t to be the only target metabolite, and we define the vertices corresponding to the elements of H as the sources $\mathcal{S}$ of G. □

Observe that in the above reduction, there is a one-to-one relation between hitting sets and precursor sets, and a related pair is of the same size. This implies that MinSizePS is as hard to approximate in polynomial time as HittingSet, which is known to be APX-hard (Ausiello *et al.*, 1999). Namely, no polynomial time algorithm for MinSizePS can have approximation ratio $o(\log n)$ unless P = NP (Raz and Safra, 1997).

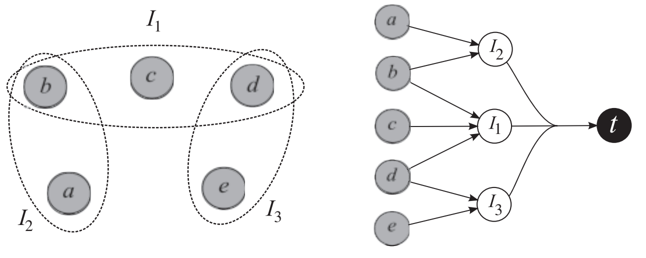

Fig. 2. Reduction of an instance of the hitting set problem. Each hitting set of $\mathcal{I} = \{I_1, I_2, I_3\}$ corresponds to a precursor set of $\{t\}$ (and vice versa).

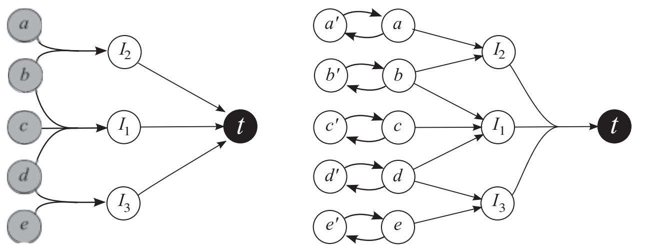

Fig. 3. Modification of the hitting set reduction to the proof of hardness of MinSizePCS (left) and to the proof of Proposition 5 (right).

A similar proof shows NP-hardness for the problem of finding a minimum size cut set. We consider in this case the same reduction but with two modifications (Fig. 3, left): (i) replace the hyperarc r_t (from $\{I_1, \dots, I_n\}$ to t) by n separate reactions, from each I_j to t, for $j \in \{1, \dots, n\}$ and (ii) replace, for each I_j, the set of reactions producing I_j by a single reaction r_j producing I_j from the whole set of elements of I_j. In this case, each hitting set corresponds to a cut set. Therefore, MinSizePCS(T) is NP-hard and APX-hard.

Related hardness results are as follows:

Proposition 5. *Given a precursor set X of T, the following two problems are NP-hard:*

1. *Find a minimum cardinality set of metabolites Z such that Z is an internal supply of X.*
2. *Find a minimum cardinality set of reactions F such that F is a factory from X* to T.

Proof. We modify the reduction presented in the proof of Theorem 4 as follows (Fig. 3, right): for each element h in H, we create another extra vertex h' in $\mathcal{C}$, and two reactions $r_{hh'}$ and $r_{h'h}$ from h to h' and from h' to h, respectively, the set of sources $\mathcal{S}$ is empty and the remaining of the construction stays the same. It is easy to see that the only minimal precursor set of T is the empty set. Using similar arguments as in the previous reduction, we have that any possible set Z corresponds to a hitting set. Analogously, any possible factory F corresponds also to a hitting set. □

3.3 Enumerating all minimal precursor sets and cut sets

We showed that MinimalPS($G, \mathcal{S}, T$) can be solved in polynomial time. Nevertheless, if we are interested in finding *all* minimal precursor sets of T, the number of solutions can grow

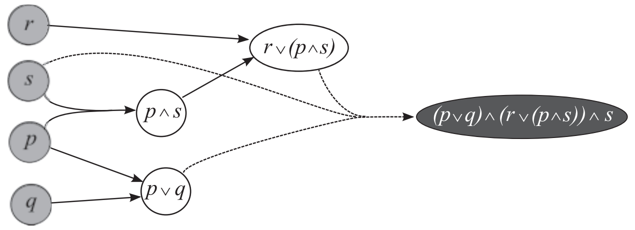

Fig. 4. Graphical representation of the reduction presented in Theorem 6.

exponentially. We are therefore interested in knowing whether AllPS can be solved in polynomial *total* time, that is, polynomial in the size of the input and output (Johnson *et al.*, 1988).

Given a boolean $\wedge, \vee-$formula f (that is with no negation), a *prime implicant* is a minimal set of variables such that if they are all true then f is true (for instance, $\{p, s\}$ is a prime implicant of $f = (p \vee q) \wedge (r \vee (p \wedge s)) \wedge s$). Enumerating the set of all prime implicants of f cannot be done in polynomial total time unless P = NP (Gurvich and Khachiyan, 1999). We show that this problem can be reduced to AllPS.

Theorem 6. *The enumeration problem* AllPS *cannot be solved in polynomial total time unless $P = NP$.*

Proof. Let f be an $\wedge, \vee$-formula. The set $\mathcal{C}$ of metabolites corresponds to the set of variables plus one metabolite for each conjunction and disjunction inside the formula (Fig. 4). The sources are the metabolites corresponding to each single variable. The set of hyperarcs is as follows: for each metabolite representing a conjunction c in f, there is a single hyperarc from the clauses of c to the metabolite c, and for each metabolite representing a disjunction d, there are arcs from each term of d to the metabolite d. The target set is a singleton containing the metabolite representing f. Clearly, a minimal precursor set of T corresponds to a prime implicant of f and vice versa. □

Observe that the reduction holds even in the case of networks without *cycles* (for any reasonable definition of cycle). This result is also valid if we consider the enumeration of all minimal precursor *cut* sets of T. Indeed, in the reduction of Theorem 6, a minimal cut set corresponds exactly to a prime *implicate* of the boolean function f, that is to a minimal set of variables such that if all are false then f is false. As for prime implicants, enumerating the set of prime implicates cannot be done in polynomial total time unless P=NP (Gurvich and Khachiyan, 1999). Thus, the enumeration problem AllPCS cannot be solved in polynomial total time unless P = NP.

3.4 Simultaneous enumeration of precursor sets and cut sets

Although enumeration of minimal precursor sets and enumeration of minimal cut sets are both hard problems, we now show that the enumeration of both problems *simultaneously* can be done in quasi-polynomial total time (that is in time $N^{O((\log N)^c)}$ for some c fixed and N the size of the input and output). Indeed, we can represent any set of sources $X \subseteq \mathcal{S}$ as a vector of $\{0, 1\}^{|\mathcal{S}|}$. We denote by $\mathbb{P}_T$ the collection of all minimal precursor sets of

T, and by $\mathbb{C}_T$ the collection of all minimal cut sets of T. We define the function F: $\{0, 1\}^{|\mathcal{S}|} \rightarrow \{0, 1\}$ as $F(X) = 1$ if X is a precursor set of T and $F(X) = 0$ otherwise. It is easy to see that F is a monotone boolean function (although it is not explicitly expressed as a conjunction and disjunction of literals) whose prime implicants are exactly $\mathbb{P}_T$ and whose prime implicates are exactly $\mathbb{C}_T$.

In Gurvich and Khachiyan (1999), the authors show an incremental method to enumerate both prime implicants and prime implicates of a monotone boolean function *at the same time*. Roughly, given a collection $\mathbb{P}' \cup \mathbb{C}'$ of solutions already found (with $\mathbb{P}' \subseteq \mathbb{P}_T$ and $\mathbb{C}' \subseteq \mathbb{C}_T$), the method finds a set $X \subseteq \mathcal{S}$ such that X is not superset of any minimal precursor set in $\mathbb{P}'$ and $\mathcal{S} \setminus X$ is not superset of any minimal cut set in $\mathbb{C}'$. Since either X is a precursor set or $\mathcal{S} \setminus X$ is a cut set, we have found a new solution not in $\mathbb{P}' \cup \mathbb{C}'$.

The algorithm finds this new solution in $O(n(\tau + n)) + m^{O(\log m)}$, where n is the number of variables, m the number of solutions already found and τ is the time to compute the value $F(X)$. By Theorem 2, τ is $O(|\mathcal{C}||\mathcal{R}| + |\mathcal{R}|^2 K)$. Therefore, given m solutions in $\mathbb{P}_T \cup \mathbb{C}_T$, we can obtain a new solution in time $O(|\mathcal{S}||\mathcal{C}|^2 + |\mathcal{S}||\mathcal{C}||\mathcal{R}|K + |\mathcal{S}|^2) + m^{O(\log m)}$.

COROLLARY 7. *The collections $\mathbb{P}_T$ and $\mathbb{C}_T$ can be jointly enumerated in quasi-polynomial incremental (and hence total) time.*

Observe that applying the same method to enumerate only one collection (i.e. discarding the solutions of the other) can be very inefficient. Some instances can have exponentially more cut sets than precursor sets (that is $|\mathbb{P}_T| << |\mathbb{C}_T|$), and thus obtaining all precursor sets can take more than quasi-polynomial time compared with $|\mathbb{P}_T|$. Analogously, there are instances where $|\mathbb{C}_T| << |\mathbb{P}_T|$. In the next section, we present algorithms to enumerate all precursors sets by taking advantage of the network topology.

4 PRECURSOR SETS ENUMERATION

We present two new algorithms that compute the collection of all minimal precursor sets of a target set T. To facilitate the exposition, we suppose that the metabolic network studied has the following properties: (i) each source $x \in \mathcal{S}$ is not produced by any reaction and (ii) each reaction belongs to at least one factory from the sources to the target. It is not difficult to see that by applying the following steps, we transform any network in order to satisfy these conditions without changing the collection of precursor sets of a target T:

1. Sources are not products: rename as x' each x in $\mathcal{S}$ that is the product of at least one reaction. Then, add a new reaction with substrate a new metabolite labelled x and product x'. The set of sources continues to be $\mathcal{S}$.
2. All reactions in a factory: compute the maximal factory (see proof of Theorem 2) and remove all reactions in the complement. Remove all unconnected metabolites.

4.1 Backtracking from the target to the sources

The general approach to enumerate $\mathbb{P}_T$ (the collection of all minimal precursor sets of T) proceeds by backtracking: starting from the target, the method performs a kind of depth-first search on the hypergraph using reactions in opposite direction. In this way, the factories that produce T starting from any minimal source sets are covered. Since we are considering hyperarcs, each factory is composed by more than one path of the depth-first search from the target to the sources. We thus obtain the whole set of solutions only at the end of the algorithm, when all paths have been travelled.

A similar idea was already presented in Cottret *et al.* (2008). The algorithm PITUFO was proposed to enumerate all minimal precursor sets by building a *replacement tree* which represented all paths obtained by going from T to the sources by using the reactions in reverse order. In a second step, this tree was compacted from the sources to the target until it reaches depth 2, on which the solutions were easily recognizable. The main problem of this method is the huge amount of memory needed to build the replacement tree, which made the algorithm useful only for small networks.

4.2 Decomposing into subproblems

We decompose the problem into *subproblems* where each subproblem has M as target set. Starting from $M := T$, two kinds of subproblem decompositions are successively applied:

1. Target decomposition: given a target $M = \{m_1, \ldots, m_k\}$ and X a precursor set of M, then X can be written as $X = \cup_{i=1}^{k} X_i$, where each X_i is a precursor set of $M_i = \{m_i\}$. Thus, we can enumerate $\mathbb{P}_M$ by enumerating $\mathbb{P}_{\{m\}}$ (the minimal precursor sets of $\{m\}$) for each metabolite $m \in M$ and taking all the corresponding unions of solution sets (one set from each collection).
2. Reaction decomposition: if the target is a singleton $M = \{m\}$ and $r_1, \ldots, r_\ell$ is the set of all reactions producing m (which is not empty if m is not a source), then X is a precursor set of M if and only if X is a precursor set of some $M_i = Subs(r_i)$ with $i \in \{1, \ldots, \ell\}$. Thus, to enumerate $\mathbb{P}_{\{m\}}$, we can enumerate $\mathbb{P}_{Subs(r)}$ for all the reactions r that produce m, and then take the union of the collections.

In both decompositions, solutions are obtained after discarding the possible non-minimal sets obtained. Successively alternating these decompositions, the aim is to have subproblems where the target is a singleton *source* $\{s\}$, which has the set $\{s\}$ itself as the only precursor set, that is $\mathbb{P}_{\{s\}} = \{\{s\}\}$.

4.3 Including available metabolites in the input

The difficulty in implementing the above described approach is given by the presence of cycles in the network. Indeed, cycles can make the algorithm enter into an endless loop. For this reason, we must include explicitly in the input of the subproblems the set of *available metabolites*, that is the metabolites already analysed in previous steps of the algorithm. In this way, we can avoid continuing the search for precursor sets of these metabolites.

Thus, given a set A of available metabolites, we conveniently consider the following generalization of the precursor set definition.

DEFINITION 3. *Given a set of sources $\mathcal{S}$, a target M and a set A of available metabolites, we say that a set $X \subseteq \mathcal{S}$ is a* precursor set of M when A is available *if there is a factory from $X \cup A$ to M.*

Observe that, if $\mathbb{P}_M(A)$ denotes the collection of all precursor sets of M when A is available, then $\mathbb{P}_T(\emptyset)$ is exactly the collection of all minimal precursor sets of T. Thus, starting from T and having $A = \emptyset$ available, we successively apply the target and reaction decompositions increasing, at each step, the set of available metabolites. We finally need to solve the subproblems $\mathbb{P}_{\{m\}}(A)$ which are composed of one of these two *base* cases:

(a) m is available: if m is in A, then $\mathbb{P}_{\{m\}}(A)$ contains only the empty set as element, i.e. $\mathbb{P}_{\{m\}}(A) = \{\emptyset\}$.

(b) m is not available but is a source: if $m \in \mathcal{S} \setminus A$, then $\mathbb{P}_{\{m\}}(A) = \{\{m\}\}$.

4.4 Increasing the set of available metabolites

We show how the set of available metabolites can be increased in each subproblem decomposition. Observe that increasing the set A of available metabolites is not only necessary to avoid cycling. The bigger is this set, the shorter are the factories that produce the given target when A is available, that is we can arrive faster to the base case (a). Thus, in each decomposition, we try to maximize the set of available metabolites that can be added without changing the solution of the original problem.

As mentioned before, to enumerate the collection $\mathbb{P}_M(A)$, we can enumerate the collections $\mathbb{P}_{\{m_i\}}(A)$ for each $m_i \in M$ and compute all possible unions of its elements (one from each collection). In fact, in the enumeration of the solutions of $\mathbb{P}_{\{m_i\}}(A)$, we can include as available any other metabolite in M different from m_i, that is $\mathbb{P}_{\{m_i\}}(A \cup (M \setminus \{m_i\}))$. Indeed, we know that these metabolites will be produced by the precursor sets given by the other *parallel* subproblems called. In the next lemma, for a given collection of sets $\mathbb{X}$, *minimal*[$\mathbb{X}$] is the collection of all sets of $\mathbb{X}$ that are not supersets of any other set of $\mathbb{X}$.

LEMMA 8. *Given the sets $M = \{m_1, \ldots, m_\ell\} \subseteq \mathcal{C}$ and $A \subseteq \mathcal{C}$, we have the following relation:*

$$\mathbb{P}_M(A) = minimal\left[\left\{\bigcup_{i=1}^{\ell} X_i \text{s.t.} X_i \in \mathbb{P}_{\{m_i\}}\Big(A \cup (M \setminus \{m_i\})\Big)\right\}\right].$$

PROOF. Given $X \in \mathbb{P}_M(A)$, there is a factory F such that $M \cup Subs(F) \subseteq X \cup A \cup Prod(F)$. Therefore, $\{m_i\} \cup Subs(F) \subseteq X \cup A \cup Prod(F)$ for all $m_i \in M$. Adding $(M \setminus \{m_i\})$ to the right side, we conclude that F is a factory from $X \cup (M \setminus \{m_i\}) \cup A$ to $\{m_i\}$ for all $m_i \in M$.

Conversely, given for all $m_i \in M$ the sets $X_i \in \mathbb{P}_{\{m_i\}}((M \setminus \{m_i\}) \cup A)$, there exist sets F_i such that $\{m_i\} \cup Subs(F_i) \subseteq X_i \cup (M \setminus \{m_i\}) \cup A \cup Prod(F_i)$ for all $m_i \in M$. This implies that $m_i \in X_i \cup A \cup Prod(F_i)$, and also implies that $M \cup Subs(F) \subseteq X \cup M \cup A \cup Prod(F)$, where $F = \cup_i F_i$ and $X = \cup_i X_i$. These two relations in turn imply $M \cup Subs(F) \subseteq X \cup A \cup Prod(F)$. □

In the case of *reaction decomposition*, computation of $\mathbb{P}_{\{m\}}(A)$ (when we are not in one of the base cases) requires to compute $\mathbb{P}_{Subs(r)}(A)$ for any reaction r producing m. Clearly, since r produces m, we can include m as available in the subproblems, that is $\mathbb{P}_{Subs(r)}(A \cup \{m\})$ (which avoids getting into an endless loop). Furthermore, we can also include any other product of r, that is $\mathbb{P}_{Subs(r)}(A \cup Prod(r))$.

LEMMA 9. *Given $m \in \mathcal{C}$ and $A \subseteq \mathcal{C}$, if $m \notin \mathcal{S} \cup A$, then we have the following relation:*

$$\mathbb{P}_{\{m\}}(A) = minimal\Big[\bigcup_{\forall r \text{ producing } m} \mathbb{P}_{Subs(r)}\ \big(A \cup Prod(r)\big)\Big].$$

PROOF. Consider $X \in \mathcal{S}$ and $F \subseteq \mathcal{R}$ such that $Subs(r) \cup Subs(F) \subseteq A \cup Prod(r) \cup X \cup Prod(F)$. Since $m \in Prod(r)$, we have $\{m\} \cup Subs(\{r\} \cup F) \subseteq A \cup X \cup Prod(\{r\} \cup F)$. Hence, $\{r\} \cup F$ is a factory from $A \cup X$ to $\{m\}$. Conversely, consider and $F \subseteq \mathcal{R}$ a factory from $X \cup A$ to the target $\{m\}$. Then, F must contain a reaction r that produces m. Therefore, $Subs(r) \cup Subs(F) = Subs(F) \subseteq A \cup X \cup Prod(F) \subseteq A \cup Prod(r) \cup X \cup Prod(F)$. F is also a factory from $X \cup A \cup Prod(r)$ to $Subs(r)$.

Hence, the collection of sets X having a factory from $X \cup A$ to the target $\{m\}$ is the same as the collection of sets of X having a factory from $X \cup Prod(r) \cup A$ to $Subs(r)$ for any r producing m. By considering minimality on each side, we conclude the proof. □

4.5 Pruning solutions by minimality

While performing reaction decomposition, there might exist reactions that can be *a priori* discarded because they do not give any *minimal* solution. Indeed, if r and r' produce m, and furthermore if the set of substrates of r that are not in A is a subset of the set of substrates of r' that are not in A, then for any solution to the subproblem defined on $Subs(r')$, we have a solution smaller or equal on $Subs(r)$. In other words, any solution given by r' is not minimal or is included in the solutions given by r. Therefore, we can avoid computing $\mathbb{P}_{Subs(r')}(Prod(r') \cup A)$ without losing minimal precursor sets.

LEMMA 10. *Let r and r' be two reactions producing m such that $Subs(r) \setminus A \subseteq Subs(r') \setminus A$. Then for any solution $X' \in \mathbb{P}(Subs(r'), Prod(r') \cup A)$, there is a solution $X \in \mathbb{P}(Subs(r), Prod(r) \cup A)$ such that $X \subseteq X'$.*

PROOF. Since $X' \in \mathbb{P}_{Subs(r')}(Prod(r') \cup A)$, there exists X' such that $Subs(r') \cup Subs(F) \subseteq Prod(r') \cup A \cup X' \cup Prod(F)$. Therefore, $Subs(F \cup \{r'\}) \cup A \subseteq A \cup X' \cup Prod(F \cup \{r'\})$.

By hypothesis, $Subs(r) \subseteq Subs(r') \cup A$, and then we can add $Subs(r)$ to the left side of the previous equation: $Subs(r) \cup Subs(F \cup \{r'\}) \cup A \subseteq A \cup X' \cup Prod(F \cup \{r'\})$. By removing the union of A on the left and adding the union of $Prod(r)$ on the right, we obtain $Subs(r) \cup Subs(F \cup \{r'\}) \subseteq A \cup Prod(r) \cup X' \cup Prod(F \cup \{r'\})$. In other words, $F \cup \{r'\}$ is a factory from $A \cup Prod(r) \cup X'$ to $Subs(r)$. We conclude that there exists $X \subseteq X$ such that $X \in \mathbb{P}_{Subs(r)}(Prod(r) \cup A)$. □

Thus, if we compute the solutions of $\mathbb{P}_{\{m\}}(A)$ by using Lemma 9, we can first compute $Subs(r) \setminus A$ for all reactions r producing m and not consider those reactions where this set is not minimal.

Algorithm TRD: target–reaction decomposition

Our first algorithm, called TRD, consists in successively applying target and reaction decompositions using the procedures *TDecomp* and *RDecomp* below until reaching the base cases. The first method uses the subroutine *CrossUnions*($\mathbb{U}$, $\mathbb{P}_m$) that computes the collection of all unions of one set of $\mathbb{U}$ and one set of $\mathbb{P}_m$. Running *TDecomp*(M, A), we obtain exactly all the minimal precursor sets of M when A is available. Algorithm TRD is therefore given by the execution of *TDecomp*(T,∅).

```
TDecomp($M \subseteq \mathcal{C}$, $A \subseteq \mathcal{C}$):
  $\mathbb{U} := [\{\}]$;
  For each metabolite $m \in M$ do
    If $m$ is in $A$ then $\mathbb{P}_m := [\{\}]$
    else if $m$ is in $\mathcal{S}$ then $\mathbb{P}_m := [\{m\}]$
    else
      $\mathbb{P}_m :=$ RDecomp($m$, $A \cup (M \setminus \{m\})$);
    $\mathbb{U} := CrossUnions(\mathbb{U}, \mathbb{P}_m)$;
  Return the collection $minimal(\mathbb{U})$.

RDecomp($m \in \mathcal{C}$, $A \subseteq \mathcal{C}$):
  $\mathbb{P} := [\,]$;
  For each reaction $r$ producing $m$ with $Subs(r) \setminus A$ minimal do
    $\mathbb{U}_r :=$ TDecomp($Subs(r)$, $A \cup Prod(r)$);
    $\mathbb{P} := \mathbb{P} \cup \mathbb{U}_r$;
  Return the collection $minimal(\mathbb{P})$.
```

4.6 Including factories as pseudo-reactions

A main limitation of TRD is the following. Method *RDecomp*(m, A) outputs all precursor sets of $\{m\}$ when A is available. This means that, if X is a set included in the output, we know that there exists a factory from $X \cup A$ to $\{m\}$. If *RDecomp* is called again as a subproblem on the same metabolite m with a similar available set A', then most of the successive decompositions will be repeated again until the base cases are reached. In this sense, the algorithm has no memory about the factories previously computed.

We propose a new algorithm that, each time that a decomposition is finished, includes this information in the network by adding *pseudo-reactions* representing the previously computed factories. For instance, if in the network there is a factory from $X \cup A$ to $\{m\}$ given by reactions r_1 and r_2, then we include a pseudo-reaction $\bar{r}_{1+2}$ with $Subs(\bar{r}_{1+2}) = X \cup A$ and $Prod(\bar{r}_{1+2}) = \{m\}$. Clearly, this operation is *safe*: the precursor sets of T do not change.

Moreover, we do not want to lose the information about the remaining of the metabolites produced by the factory, which are used to increment the set A of available metabolites. For this reason, we associate to each pseudo-reaction $\bar{r}$ a set $Int(\bar{r})$ of *internal available metabolites* which contains any metabolite produced by the reactions represented by $\bar{r}$. Thus, if we use this reaction in a future decomposition, we can consider this set as available. If we define $Int(r) = Prod(r)$ for any original reaction r of the network, then we do not need to distinguish between reactions and pseudo-reactions. In the previous example, we have then that $Int(\bar{r}_{1+2}) = Int(r_1) \cup Int(r_2)$.

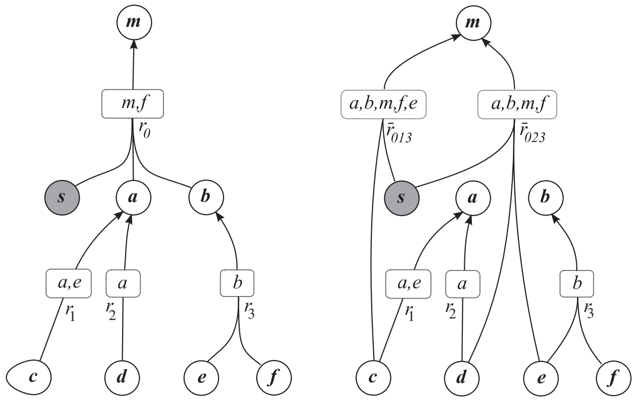

Fig. 5. Example of the application of *Replace* to reaction r_0. Left*:* reaction r_0 has internal production m and f (enclosed in a rectangle). The substrates of r_0 are s (which is a source), a and b. The collection $R_{\min}(r_0)$ contains the minimal sets of reactions that produce a and b, that is $R_{\min}(r_0) = [\{r_1, r_3\}, \{r_2, r_3\}]$. Right*:* we replace r_0 by new reactions corresponding to the merge of r_0 to each set of reactions of $R_{\min}(r_0)$. Thus, reaction r_0 is replaced by reactions $\bar{r}_{013}$ and $\bar{r}_{023}$. Notice that the substrates of $\bar{r}_{013}$ do not include substrates of r_3 since they are internally produced by r_1 and r_0.

4.7 Reaction replacement

Adding new pseudo-reactions to the network decreases the number of reactions of the factories from X to $\{m\}$. However, to really decrease the time, we need to ensure that the algorithm will not consider again the original factory. Otherwise, if m is revisited, the algorithm would analyse both the original factory and the new one containing the new added reactions. To avoid this, we delete the original reaction producing m but while guaranteeing that the collection of minimal precursor sets of T is maintained.

Suppose that we want to delete a reaction r producing m. Notice that any factory from X to m that contains r must also contain at least one reaction producing each substrate of r (except the sources). Thus, if r is merged with each set of such reactions then r can be removed without modifying the minimal precursor sets.

More formally, given a reaction r, we say that a set of reactions R is a *predecessor reaction set* of r, if R produces all the substrates of r that are not sources, that is $Prod(R) \supseteq Subs(r) \setminus \mathcal{S}$. Let $\mathcal{R}_{min}(r)$ be the collection of all minimal predecessor reaction sets of r. Clearly, any factory containing r must also contain a set $R \in \mathcal{R}_{min}(r)$. The following method *Replace*(r) removes reaction r and adds pseudo-reactions corresponding to the merge of r with every reaction set $R \in \mathcal{R}_{min}(r)$ (Fig. 5).

```
Replace($r \in \mathcal{R}$)
  Compute $\mathcal{R}_{min}(r) = min\{R \subseteq \mathcal{R} \mid Prod(R) \supseteq Subs(r) \setminus \mathcal{S}\}$;
  For each set $R \in \mathcal{R}_{min}$ do
    Add a new reaction $\bar{r}_R$ to the network with
      $Prod(\bar{r}_R) := Prod(r)$,
      $Int(\bar{r}_R) := Int(R) \cup Int(r)$;
      $Subs(\bar{r}_R) := (Subs(R) \cup Subs(r)) \setminus Int(\bar{r}_R)$,
  Remove $r$ from the network
```

LEMMA 11. *Let $r \in \mathcal{R}$ be a reaction of the network G and let G′ be the network that results from applying the procedure* Replace(r). *Then, X is a precursor set of T in G if and only if X is a precursor set of T in G′.*

PROOF. The factories in G which do not include r are factories also in G'. Let $F \subseteq \mathcal{R}$ be a factory from X to T in G which contains r. Clearly, F must contain a set $R \in \mathcal{R}_{\min}(r)$. Thus, the set $F' = F \cup \{\bar{r}_R\} \setminus \{r\}$ is a factory from X to T in G'. Conversely, if F' is a factory from X to T in G' containing the set $R_{\text{new}} = \{\bar{r}_{R_1}, \ldots, \bar{r}_{R_k}\}$ of new reactions added by *Replace(r)*, then $F = (F' \setminus R_{\text{new}}) \cup \{r\} \cup \bigcup_{i=1}^{k} R_i$ is a factory from X to T in G. □

Algorithm NS: network shortcutting

We define a new algorithm ns to compute all precursor sets of a target T based on reaction replacement. The following preprocessing of the network is required: a new metabolite t and a new reaction r_t are created, such that $Subs(r_t) = T$ and $Prod(r_t) = \{t\}$. Clearly, the minimal precursor sets of $\{t\}$ are exactly the minimal precursor sets of T.

Starting from $r := r_t$, NS traverses the network in the same way that TRD does. However, instead of computing the minimal solutions, NS goes deep in the recursion until finding a reaction r satisfying the following two conditions: (a) *not all* substrates of r are in the base cases and (b) *all* substrates of all reactions in the next level of the recursion are in the base cases. When such a reaction is found, then it is replaced by new reactions.

Successively removing and adding reactions in this way, we decrease the size of the factories from $\mathcal{S}$ to $\{t\}$. Finally, the last reaction removed is r_t which is replaced by new reactions producing t and having only sources as substrates. The substrates of each reaction correspond exactly to a minimal precursor set of $\{t\}$.

Running NS(r_t, ∅) we obtain a network where the minimal precursor sets are exactly the substrate sets of all the reactions that produce t. The network can also contain many other reactions, but they are not even connected to t.

```
NS($r_0 \in \mathcal{R}$, $A \subseteq \mathcal{C}$):
  $M := Subs(r_0)$;
  If $M$ contains a metabolite not in $A \cup \mathcal{S}$ then
    For each metabolite $m \in M \setminus (A \cup \mathcal{S})$ do
      $NewA := A \cup (M \setminus \{m\})$;
      For each $r$ producing $m$ with $Subs(r) \setminus NewA$ minimal do
        NS($r$, $NewA \cup Int(r)$);
    $Replace(r_0)$;
```

5 PERFORMANCE ANALYSIS

Extensive tests were performed in order to measure the performance of the different algorithms on real metabolic networks. These algorithms were compared for several different singleton target sets (for instance, amino-acids, metabolites related to the synthesis of the cell wall, DNA, RNA, membranes, etc.) in seven networks of different sizes and topologies downloaded from MetExplore (Cottret *et al.*, 2010b). Ubiquitous metabolites were filtered out and the split reactions using pairs of co-factors option was chosen.

We adopted an automatic process to define the set of sources based on the topology of the network. A metabolite m is considered a source if it satisfies one of these two conditions: (a) m is not the product of any reaction or (b) m is involved in only two reactions corresponding to both directions of an originally reversible reaction (i.e. m is substrate in one and product in the other). The target sets were chosen based on their role: amino-acids, metabolites related to the synthesis of the cell wall or DNA, etc.

5.1 Removing bad cycles

There are some cycles that we know *a priori* that are not realistic since they are able to produce compounds outside the cycle without the need of any input. In particular, the two directions of an originally reversible reaction form a cycle which can produce its metabolites without using any external source. These *bad cycles* must be avoided in factories since they may create fake solutions in which an empty set of metabolites produces the target.

In order to avoid bad cycles in factories, we preprocess the input network breaking this kind of cycles by removing some reactions. Specifically, starting from a set M of metabolites containing only the target and an empty set R of reactions, we include in R a randomly chosen reaction producing a metabolite of M unless its inclusion generates a bad cycle. The substrates of the added reactions are included in M. Successively repeating this process we obtain a network with no bad cycles. Notice that this process corresponds to a heuristic whose result depends on the order in which reactions are chosen to be included in R.

5.2 Benchmarks

Table 1 presents an extract of the results for PITUFO, the algorithm described in Cottret *et al.* (2008) and the two different algorithms described in this article (TRD and NS) with and without the test of minimality. The targets presented are those for which finding the minimal precursor sets required more time for the new algorithms with minimality check. The table shows, for each network, the size of the sets of metabolites and reactions, and for each target, the size of the preprocessed network, the number of precursor sets found and the time in seconds that each algorithm spent.

All algorithms have been implemented in Java and the running times were collected using a cluster for the computation and setting a limit of 1 GB of RAM memory for each process. Although PITUFO may be fast for small networks, its use is limited since, as the size of the networks grows, the method takes a long time to finish, and for some targets, it does not finish in the given time limit of 24 h. This already justifies the new methods presented in this work, since they do not present the same behaviour for larger networks.

Concerning the minimality check, we may observe that it is not necessarily true that it improves the running time. In some cases, doing the check may even lead to worse results (example *Yeast*, target FADH2, NS method), while in others it may have a strong positive impact on the execution time of the algorithm which becomes 700 times faster (example, *Escherichia coli*, target L-aspartate, TRD method).

Table 1. Runtime (in seconds) for computing minimal precursor sets of three singleton targets in seven different networks using five methods: PITUFO, TRD without and with the minimality pruning (respectively All and Min) and NS without and with the minimality pruning (respectively All and Min)

Network ($\|\mathcal{C}\|/\|\mathcal{R}\|$)	PITUFO	TRD		NS	
Target ($\|\mathcal{C}\|/\|\mathcal{R}\|$ after preprocess)		All	Min	All	Min
S. muelleri (75/65)					
L-Arginine (33/22)	0.017	0.062	0.02	0.015	0.018
L-Isoleucine (32/21)	0.008	0.069	0.02	0.015	0.016
L-Lysine (31/20)	0.014	0.084	0.019	0.021	0.016
Carsonella Ruddii (114/126)					
L-Leucine (86/56)	0.005	0.106	0.046	0.035	0.047
L-Isoleucine (83/49)	0.055	0.105	0.032	0.036	0.040
L-Valine (83/49)	0.037	0.091	0.030	0.028	0.035
B. cicadellinicola (236/229)					
Octapremyl diphos, (149/160)	0.726	0.283	0.209	0.221	0.195
Tetrahydrofolate (148/149)	0.337	0.227	0.170	0.237	0.179
Heme-O (150/161)	1.164	0.319	0.208	0.217	0.172
B. aphidicola (396/338)					
Pyruvate (219/87)	0.082	0.131	0.105	0.105	0.104
dGTP (206/76)	0.099	0.138	0.126	0.118	0.101
UTP (219/87)	0.113	0.117	0.099	0.148	0.104
Yeast (703/1010)					
FADH2 (444/314)	*	14.39	5.55	7.27	14.55
L-Histidine (415/269)	*	5.55	4.80	5.02	6.62
L-Aspartate (410/ 274)	176.40	4.53	4.65	4.82	4.66
Human (997/1225)					
L-Alanine (710/359)	5058.27	5.15	3.34	10.76	10.78
Seriapterine (698/329)	*	3.19	2.96	6.85	2.88
L-Cysteina (150/161)	5579.85	3.32	3.32	4.22	3.17
E. coli (1010/1164)					
L-Aspartate (714/507)	*	2139.01	3.32	10.57	47.72
L-Metionine (737/545)	*	632.20	13.62	14.08	14.17
Glycine (706/503)	*	553.21	11.55	11.01	13.90

All methods were applied to the same preprocessed network on each target. In the cases marked '*', the algorithm did not finish within 24 h. For each target, the size of metabolites and reactions after bad cycle deletion is indicated.

Notice also that as the size and complexity of the networks increase, the number of different minimal precursor sets found increases also, and it does this at a rate faster than the increase of the time needed to compute them.

5.3 Computing solutions for several preprocessed networks

As mentioned before, the network free of bad cycles that is given by the heuristic proposed depends on the order in which reactions are added to R. Thus, different orders can generate different minimal precursor sets. To recover as many solutions as possible, we can repeat the search for precursors on several different results of the preprocessing part. In order to analyse the effect of this heuristic on the algorithms, we successively repeated this random process while computing, at each repetition, the number of new precursor sets obtained. The process stops

Table 2. Computation of minimal precursor sets of the *E. coli* network for three targets, using several different preprocessed networks

Target	Convergency		Prec. sets	Iteration reaching X% of Prec. sets			
	Iteration	Time (s)		25%	50%	80%	95%
L-Aspartate	71	3128	267	1	6	49	57
L-Metionine	94	2738	399	1	5	46	80
Glycine	73	2693	242	1	4	19	56

Each iteration corresponds to repeating the heuristical random preprocessing and computing the minimal precursor sets using trd with minimality. For each target, we show the iteration where the convergence is reached, the time required, the total number of different minimal precursor sets (at the convergence), and the iterations in which a given percentage of this total number of solutions is recovered.

when no new precursor set is recovered in 10 consecutive repetitions. Analysing the results for three different targets of *E. coli*, this convergence criterium was reached in <100 iterations. In the three cases, >50% of the solutions were recovered in the first six repetitions and >80% in the first 50 iterations (Table 2).

6 CONCLUSION

Despite the proved hardness of enumerating all precursor sets of a given target, the algorithms presented in this article can find all solutions for networks of up to around a 1000 reactions. If we restrict ourselves to the benchmark built for this article, the TRD method with minimality check is the one that presented the best behaviour on average. However, the methods vary widely depending on the target chosen. Our benchmark does not allow us to conclude which algorithm has a better performance between TRD with a minimality test and NS with and without minimality test for bigger networks. This justifies the utility of each method individually and leaves an open space for further improvements.

Funding: This work was partially funded by the following projects: the French project ANR MIRI BLAN08-1335497, the French project ANR NEMO ANR-08-BLAN-0304-01, the ERC Advanced Grant Sisyphe held by Marie-France Sagot, the INRIA Associated Team SIMBIOSI, the INRIA International Partnership AMICI, the Dutch NWO-CLS MEMESA project and the Chilean FONDAP 15090007 project.

Conflict of Interest: none declared.

REFERENCES

Ausiello,G. *et al.* (1999) *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, Berlin.

Cottret,L. *et al.* (2008) Enumerating precursor sets of target metabolites in a metabolic network. In *Proceedings of the 8th International Workshop on Algorithms in Bioinformatics (WABI)*. Karlsruhe, Germany, September 15–17, 2008, Lecture Notes in Computer Science, 5151, Springer Verlag, Berlin, pp. 233–244.

Cottret,L. *et al.* (2010a) A Graph-based analysis of the metabolic exchanges between two co-resident intracellular symbionts, *Baumannia cicadellinicola* and *Sulcia muelleri*, with their insect host, *Homalodisca coagulata. PLoS Comput. Biol.*, **6**, e1000904.

Cottret,L. *et al.* (2010b) Metexplore: a web server to link metabolomic experiments and genome-scale metabolic networks. *Nucleic Acids Res.*, **38**, 132–137.

Feist,A.M. *et al.* (2009) Reconstruction of biochemical networks in microorganisms. *Nat. Rev. Microbiol.*, **7**, 129–43.

Garey,M. and Johnson,D. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Series of books in the mathematical sciences. W.H. Freeman, New York.

Gurvich,V. and Khachiyan,L. (1999) On generating the irredundant conjunctive and disjunctive normal forms of monotone boolean functions. *Discrete Appl. Math.*, **96–97**, 363–373.

Johnson,D.S. *et al.* (1988) On generating all maximal independent sets. *Inform. Process. Lett.*, **27**, 119–123.

McCutcheon,J.P. and Moran,N.A. (2007) Parallel genomic evolution and metabolic interdependence in an ancient symbiosis. *Proc. Natl Acad. Sci. USA*, **104**, 19392–19397.

Raz,R. and Safra,S. (1997) A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*. STOC'97, pp. 475–484.

Romero,P. and Karp,P.D. (2001) Nutrition-related analysis of pathway/genome databases. In *Pacific Symposium on Biocomputing'01*. pp. 470–482.

Thiele,I. and Palsson,B.O. (2010) A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nat. Protoc.*, **5**, 1750–2799.