Metabolic Network Expansion with Answer Set Programming

Torsten Schaub and Sven Thiele

Universität Potsdam, Institut für Informatik, August-Bebel-Str. 89, D-14482 Potsdam, Germany

Abstract. We propose a qualitative approach to elaborating the biosynthetic capacities of metabolic networks. In fact, large-scale metabolic networks as well as measured datasets suffer from substantial incompleteness. Moreover, traditional formal approaches to biosynthesis require kinetic information, which is rarely available. Our approach builds upon a formal method for analyzing large-scale metabolic networks. Mapping its principles into Answer Set Programming (ASP) allows us to address various biologically relevant problems. In particular, our approach benefits from the intrinsic incompleteness-tolerating capacities of ASP. Our approach is indorsed by recent complexity results, showing that the reconstruction of metabolic networks and related problems are NP-hard.

1 Introduction

The availability of high-throughput methods in biology has resulted in a rapid growth of biological knowledge, gathered in web databases such as KEGG[1] or MetaCyc[2]. Of particular interest are biosynthetic capacities of metabolic networks in view of the design of bioprocesses. However, large-scale metabolic networks as well as measured datasets suffer from substantial incompleteness. Many networks are only partially defined and only few metabolites can be identified without ambiguity. Moreover, traditional formal approaches to biosynthesis (cf. [1–5]) require kinetic information, which is rarely available.

We address this problem and propose a qualitative approach based on Answer Set Programming (ASP;[6]). This approach benefits from the intrinsic incompleteness-tolerating capacities of ASP and allows us to take advantage of its rich modelling language and highly efficient implementations. Our approach is indorsed by recent complexity results, showing that the reconstruction of metabolic networks and related problems are NP-hard [7, 8].

Our approach builds upon a formal method for analyzing large-scale metabolic networks developed in [9, 10]. The basic idea is that a reaction operates only if its reactants are either available as nutrients or can be provided by other metabolic reactions. Starting from some nutrients, referred to as *seeds*, this allows for expanding a metabolic network by successively adding operable reactions and their products. The set of metabolites in the resulting network is called the *scope* of the seeds and represents all metabolites that can principally be synthesized from the seeds by the analyzed metabolic network.

[1] http://www.genome.jp/kegg

[2] http://metacyc.org

Mapping the principles of this approach into ASP allows us to address various biologically relevant problems. A primary problem deals with the completion of genome-scale metabolic networks. When building a metabolic network, as for the recently sequenced green alga *Chlamydomonas reinhardtii*[3], the initial core draft is done by appeal to genomic information. Then, experimental data, in particular, measured metabolites, are taken to define the functionality of the overall network. The above methodology can then be used to check whether a drafted network provides the synthesis routes to comply with the required functionality. If this fails, the draft network can be completed by importing reactions from metabolic reference network stemming from other organisms until the obtained network provides the measured functionality (cf. [11]). Another important problem concerns the determination of seed compounds needed for the synthesis of certain other compounds. As demonstrated in [12], solving this problem is important for indicating (minimal) nutritional requirements for sustaining maintenance or growth of an organism.

Both problems have a combinatorial nature and thus give rise to a multitude of solutions. We address this problem by taking advantage of the various reasoning modes provided by ASP. On the one hand, we use ASP's optimization techniques for finding cardinality or subset minimal solutions, respectively. On the other hand, we exploit consequence aggregation for finding metabolites common to all (optimal) solutions or at least one of them, respectively. Moreover, the aforementioned problems are often subject to additional constraint, aiming at the avoidance of side products or producing target products by staying clear from certain seeds, respectively. Finally, the elaboration tolerance of ASP greatly supports the process of drafting metabolic networks involving continuous validation and increasing functionalities stemming from measured data.

2 Background

Biological problem definition. Following [8], a *metabolic network* is commonly represented as a directed bipartite graph $G = (R \cup M, E)$, where R and M are sets of nodes standing for *reactions* and *metabolites*, respectively. Given a such metabolic network G, we sometimes refer to its components by $R(G)$, $M(G)$, and $E(G)$. Whenever $(m, r) \in E$ for $m \in M$ and $r \in R$, the metabolite m is called a *reactant* of reaction r; for $(r, m) \in E$, metabolite m is called a *product* of r. More formally, for $(R \cup M, E)$ and $r \in R$ define $reac(r) = \{m \in M \mid (m, r) \in E\}$ and $prod(r) = \{m \in M \mid (r, m) \in E\}$.

The aforementioned biological concept of a *scope* can be expressed in terms of reachability. Given a metabolic network $(R \cup M, E)$ and a set $M' \subseteq M$ of *seed* metabolites, a reaction $r \in R$ is *reachable* from M', if $reac(r) \subseteq M'$, that is, if all its reactants are reachable. Moreover, a metabolite $m \in M$ is *reachable* from M', either if $m \in M'$ or if $m \in prod(r)$ for some reaction $r \in R$ being *reachable* from M'. Finally, the *scope* of M', written $\Sigma_{(R \cup M, E)}(M')$ or simply $\Sigma(M')$, is the closure of M' under reachability from M'. Note that the scope of a set of metabolites can be computed in polynomial time.

[3] http://www.goforsys.de

Now, we can make precise the aforementioned biological problems. In the *metabolic network completion*, we are given a metabolic network $(R \cup M, E)$ along with two sets $S, T \subseteq M$ of (seed and target) metabolites, and a reference network $(R' \cup M', E')$. The goal is to find a set of reactions $R'' \subseteq R' \setminus R$ such that $T \subseteq \Sigma_G(S)$ where

$$G = ((R \cup R'') \cup (M \cup M''), E \cup E'') \ ,$$
$$M'' = \{m \in M' \mid r \in R'', m \in \mathit{reac}(r) \cup \mathit{prod}(r)\} \ , \text{ and}$$
$$E'' = \{(m, r) \in E' \mid r \in R'', m \in \mathit{reac}(r)\} \cup \{(r, m) \in E' \mid r \in R'', m \in \mathit{prod}(r)\} \ .$$

We call R'' the *completion* of $(R \cup M, E)$ from $(R' \cup M', E')$ wrt (S, T). Two optimization variants of this problem are obtained by finding a cardinality or subset minimal set of reactions. Further refinements may also optimize on the distance between seeds and targets or minimize forbidden side products.

Three variants of the *inverse scope problem* can be distinguished [8]. In the basic one, we are given a metabolic network $(R \cup M, E)$ and a set $T \subseteq M$ of (target) metabolites. The goal is to find a set of (seed) metabolites $S \subseteq M$ such that $T \subseteq \Sigma(S)$. The two optimization variants of this problem aim at finding a cardinality or subset minimal solution. The second problem restricts the domain of the available seed metabolites. In addition to $(R \cup M, E)$ and $T \subseteq M$, we are given a set of (forbidden) metabolites $F \subseteq M$. Then, the goal is to find a set of (seed) metabolites $S \subseteq (M \setminus F)$ such that $T \subseteq \Sigma(S)$. Apart from optimizing the required seed metabolites, one may also minimize undesired metabolites rather then excluding them. The third problem adds an additional constraint on the avoidance of side products. In addition to $(R \cup M, E)$ and $T, F \subseteq M$, we are given another set of (forbidden) metabolites $E \subseteq M$. Then, the goal is to find a set of (seed) metabolites $S \subseteq (M \setminus F)$ such that $T \subseteq \Sigma(S)$ and $\Sigma(S) \cap E = \emptyset$. As above, the optimization variants can also take side products into account.

Answer Set Programming. We refer the reader to [6] for a formal introduction to ASP and concentrate in what follows on aspects relevant to our application. A *logic program* is a finite set of *rules* of the form

$$a \leftarrow b_1, \ldots, b_m, \mathit{not}\ c_{m+1}, \ldots, \mathit{not}\ c_n \ , \tag{1}$$

where a, b_i, c_j are *atoms* for $0 < i \leq m < j \leq n$. A *literal* is an atom a or its (default) negation $\mathit{not}\ a$. A rule r as in (1) is called a *fact*, if $l = n = 1$, and an *integrity constraint*, if $l = 0$. We denote predicate and constant symbols by lowercase letters and variables by uppercase letters. A logic program with variables is regarded as the set of all its ground-instantiated rules. Moreover, we take advantage of *choice rules* and *conditional literals* [13]; both of which can be regarded as macros. In a choice rule, the head a in (1) is replaced by a set $\{a_1, \ldots, a_l\}$; it allows us to derive any subset provided the rule's body is satisfied. A conditional literal is of form $a : b$ where a and b are literals (containing common variables); informally, it stands for the sequence of all instantiations of a obtained by restricting the substitution of variables common to a and b to those of b (cf. [13] for details). For instance, given $m(1)$, $m(2)$, and $r(a)$, the choice rule $\{p(R, M) : m(M)\} \leftarrow r(R)$ stands for $\{p(a, 1), p(a, 2)\} \leftarrow r(a)$.

The answer sets of a program P are models of P satisfying a certain stability criterion (cf. [6] for details). An answer set is represented by the set of atoms that are true

in it. Apart from testing the existence of an answer set of a program or enumerating all its answers, the following reasoning modes are supported. For this, define $AS(P)$ as the set of all answer sets of Program P. Then, the cautious and brave consequence of P are defined as $\bigcap_{X \in AS(P)} X$ and $\bigcup_{X \in AS(P)} X$. Notably, both sets are computable through linear many computations of one answer set, rather than computing possibly exponential number of answer sets in $AS(P)$. Another mode of interest is solution projection [14], which computes only the projections of all answer sets on a set P of atoms, that is, $\{X \cap P \mid X \in AS(P)\}$, thereby greatly reducing computational efforts.

Cardinality based optimization is provided in ASP through minimize (or maximize) statements of the form

$$\mathit{minimize}\{b_1 = w_1, \ldots, b_m = w_m, \mathit{not}\ c_{m+1} = w_{m+1}, \ldots, \mathit{not}\ c_n = w_n\}$$

enforcing that only answer sets with minimum value of $\sum_{b_i \in X, 1 \leq i \leq m} w_i + \sum_{c_j \notin X, m+1 \leq j \leq n} w_j$ are computed, where $w_1, \ldots, w_n$ are integers. There can be several minimize and/or maximize statements which order the stable models lexicographically. Subset based minimization and/or maximization is more complex and needs the elevated expressiveness of disjunctive programs, being at the second level of the polynomial hierarchy [15]. This would necessitate more expressive ASP solvers and is thus left for future work.

3 Logic program representations

3.1 Metabolic network completion

We start by representing a metabolic network G_n as a set of facts.

$$\begin{aligned}\mathcal{G}(G_n) = \quad & \{\mathit{reaction}(r,n) \mid r \in R(G_n)\} \\ & \cup \{\mathit{reactant}(m,r) \mid r \in R(G_n), m \in \mathit{reac}(r)\} \\ & \cup \{\mathit{product}(m,r) \mid r \in R(G_n), m \in \mathit{prod}(r)\}\end{aligned}$$

While our draft network provides an incomplete biological model, the seed and target metabolites are obtained from experimental data. The seed metabolites are provided as nutrients in an experiment, the target metabolites are measured as its final outcome. A metabolic draft network G_d along with two sets $S, T \subseteq M$ of seed and target metabolites, and a reference network G_r results in the following set of facts, $\mathcal{C}(G_d, G_r, S, T)$.

$$\mathcal{G}(G_d) \cup \mathcal{G}(G_r) \cup \{\mathit{draft}(d)\} \cup \{\mathit{seed}(s) \mid s \in S\} \cup \{\mathit{target}(t) \mid t \in T\} \qquad (2)$$

The current draft network is identified by the fact $\mathit{draft}(d)$.

Draft Scope. The scope of the seed metabolites in the draft network G_d can be determined by the following rules.

$$\begin{aligned}\mathit{dscope}(M) &\leftarrow \mathit{seed}(M) \\ \mathit{dscope}(M) &\leftarrow \mathit{product}(M,R), \mathit{reaction}(R,N), \mathit{draft}(N), \\ &\qquad \mathit{dscope}(M') : \mathit{reactant}(M',R)\end{aligned} \qquad (3)$$

The first rule declares all seed metabolites $M \in S$ as producible. The second rule defines recursively that a product M of a reaction R is producible, whenever all reactants M' of R are available. Together with the encoding of G_d and S in (2), the set of of rules in (3) results in a single answer set X such that $dscope(m) \in X$ iff $m \in \Sigma_{G_d}(S)$ for $m \in M(G_d)$.

Potential Scope. While drafting a metabolic network of an organism biologists are regularly confronted with experiments that show that a certain metabolite can be measured, although it is not producible by the current draft network. To this end, they incorporate metabolic reactions known from metabolic networks of other organisms.

In analogy to the rules in (3), the (potential) scope of the seed metabolites in the draft network G_d augmented by the reference network G_r can be determined as follows.

$$\begin{array}{l} pscope(M) \leftarrow seed(M) \\ pscope(M) \leftarrow product(M,R), reaction(R,N), \\ \qquad\qquad pscope(M') : reactant(M',R) \end{array} \tag{4}$$

Note that dropping the qualification $draft(N)$ from (3) makes us use all available reactions. As before, given the encoding in (2), the set of rules in (4) induces a single answer set X such that $pscope(m) \in X$ iff $m \in \Sigma_{G_d \cup G_r}(S)$ for $m \in M$ where $G_d \cup G_r$ stands for the pairwise union of G_d and G_r.

While the scope of the draft network in (3) gives a lower limit on the metabolites producible from the seeds by the draft network, the potential scope obtained from the augmented network in (4) constitutes an upper limit. Note that targets outside the potential scope cannot be explained.

Metabolic Network Completion. The goal of metabolic network completion is to extend the draft network with reactions from the reference network, so that the target metabolites can be synthesized by the augmented network from the seeds. The reactions of interest belong to the reference network but not the draft network. The following choice rule captures all candidate reactions.

$$\{xreaction(R) : not\ reaction(R,N) : draft(N)\} \tag{5}$$

The conditional literal $not\ reaction(R,N) : draft(N)$ guarantees that all chosen reactions belong to $R(G_r) \setminus R(G_d)$. In fact, the encoding in (2) and the choice rule in (5) result in a set of answer sets being in a one-to-one correspondence to the subsets of $R(G_r) \setminus R(G_d)$.

The (extended) scope of the seed metabolites in the draft network G_d extended by reactions from G_r is defined as follows.

$$\begin{array}{l} xscope(M) \leftarrow seed(M) \\ xscope(M) \leftarrow product(M,R), reaction(R,N), draft(N), \\ \qquad\qquad xscope(M') : reactant(M',R) \\ xscope(M) \leftarrow product(M,R), xreaction(R), \\ \qquad\qquad xscope(M') : reactant(M',R) \end{array} \tag{6}$$

Finally, we have to make sure that an extended scope is able to produce all target metabolites. This is addressed by the following integrity constraint. [4]

$$\leftarrow target(M), not\ xscope(M) \tag{7}$$

Given the above rules, each of its answer set corresponds to a completion of the draft network and vice versa.

Proposition 1. *Let G_d and G_r be metabolic networks and let S and T be sets of metabolites.*

If X is an answer set of logic program[5] *$\mathcal{C}(G_d, G_r, S, T) \cup \{(5), (6), (7)\}$, then $\{r \mid xreaction(r) \in X\}$ is a completion of G_d from G_r wrt (S, T) and vice versa.*

Refined Metabolic Network Completion. Although the above encoding is formally adequate, it suffers from too many uninteresting completions that makes it fail to scale on large metabolic networks comprising several thousand metabolites. We address this problem by some refinements reducing the set of candidate reactions.

At first, we restrict the choice in (5) to "interesting" reactions.

$$\leftarrow xreaction(R), not\ ireaction(R) \tag{8}$$

The qualification expressed by $ireaction(R)$ requires that a reaction of interest must lead to some target metabolites.

$$\begin{aligned} ireaction(R) &\leftarrow interesting(M), \\ &\qquad product(M, R), reaction(R, N) \\ interesting(M) &\leftarrow target(M), not\ dscope(M) \\ interesting(M) &\leftarrow reactant(M, R), ireaction(R), \\ &\qquad not\ dscope(M) \end{aligned} \tag{9}$$

With the first rule we declare a reaction as interesting if it produces interesting metabolites. The second rule defines all target metabolites that cannot be produced by the draft network as interesting, and the third rule states that metabolites needed by interesting reactions and not producible by the draft network are interesting. This concept provides a significant reduction of the set of candidate reactions in view of the given target metabolites.

Second, we further restrict the choice in (5) to "operable" reactions.

$$\begin{aligned} &\leftarrow xreaction(R), not\ oreaction(R) \\ oreaction(R) &\leftarrow xscope(M) : reactant(M', R), \\ &\qquad reaction(R, N), not\ draft(N) \end{aligned} \tag{10}$$

The integrity constraint enforces that each extending reaction is operable, that is, satisfies $oreaction(R)$. The following rule defines a (candidate) reaction as operable, if all its reactants are producible by the current network extension.

The next result shows that the above refinements preserve soundness.

[4] In practice, this integrity constraint is extended by $pscope(M)$ to ignore non-producible targets.

[5] Recall that a rule with variables stands for the set of all its ground instantiations.

Proposition 2. *Let G_d and G_r be metabolic networks and let S and T be sets of metabolites.*

If X is an answer set of $\mathcal{C}(G_d, G_r, S, T) \cup \{(5), (6), (7), (8), (9), (10)\}$, then $\{r \mid xreaction(r) \in X\}$ is a completion of G_d from G_r wrt (S, T).

Optimal Completions. A further natural way to reduce the number of solutions is to concentrate on network completions containing the fewest number of reactions. In ASP, this can be accomplished by the following minimize statement.

$$minimize\ \{xreaction(R) : ireaction(R) : not\ reaction(R, N)\} \tag{11}$$

Interestingly, our refinements are satisfied by such minimal completions, so that we get a soundness and completeness result under optimization.

Proposition 3. *Let G_d and G_r be metabolic networks and let S and T be sets of metabolites.*

If X is an answer set of $\mathcal{C}(G_d, G_r, S, T) \cup \{(5), (6), (7), (8), (9), (10)\} \cup \{(11)\}$, then $\{r \mid xreaction(r) \in X\}$ is a minimum completion of G_d from G_r wrt (S, T) and vice versa.

Sometimes reactions can be associated with confidence levels, for instance, obtained from the proximity of their host organism to the organism addressed by the draft network. This allows us to prefer among the minimum completions those composed of reactions with higher confidence levels; this is accomplished by adding the following statement.

$$maximize\ \{xreaction(R) = L : ireaction(R) : not\ reaction(R, N) : confidence(R, L)\}$$

Reasoning Modes. Given the above ensemble of rules, the reasoning modes of ASP solvers allow us to answer a variety of additional biologically relevant questions. What target metabolites are producible by the draft network? What new metabolites can be produced by adding reactions from other pathways? What is the minimal number of reactions that must be added to explain a target metabolite? What are the minimum or minimal extended scopes? Which reactions belong to all extended scopes, or even all minimum extended scopes? The latter are accomplished by a combination of optimization and cautious reasoning. We return to these question in Section 4 and show how they are realized. The next section also shows how certain seeds or side-products can either be avoided or minimized.

3.2 Inverse Scope Problem

Given a metabolic network and a set of target metabolites, we are interested in sets of seed metabolites that allow for producing the target metabolites from the network.

Basic Setting. Reactions, targets, and seeds are represented as in Section 3. That is, given a network G_n and sets S, T of metabolites, the inverse scope problem is based on the following set of facts.

$$\mathcal{I}(G_n, S, T) = \mathcal{G}(G_n) \cup \{seed(s) \mid s \in S\} \cup \{target(t) \mid t \in T\} \tag{12}$$

By appeal to the encoding of the basic scope in (3), we can then express our task similar to the completion problem by exchanging the roles of reactions and seed metabolites.

$$\{seed(M) : not\ target(M)\} \tag{13}$$
$$\leftarrow target(M), not\ dscope(M) \tag{14}$$

Similar to (5), the choice construct in (13) captures the seed candidates, while the integrity constraint in (14) makes sure that all target metabolites can by synthesized from the seeds chosen in (13).

The next proposition shows that our encoding is sound and complete.

Proposition 4. *Let G_n be a metabolic network and let S and T be sets of metabolites.*

If X is an answer set of logic program $\mathcal{I}(G_n, S, T) \cup \{(3), (13), (14)\} \cup \{draft(n)\}$, then $T \subseteq \Sigma(\{m \mid seed(m) \in X\})$ and vice versa.

The fact $draft(n)$ is merely added for compatibility with (3).

Refined Setting. As above, some refinements lend themselves for reducing the putative seed metabolites.

$$\leftarrow seed(M), not\ imetabolite(M) \tag{15}$$

A metabolite of interest, viz $imetabolite(M)$, must lead to at least one target metabolite.

$$\begin{aligned} imetabolite(M) &\leftarrow target(M) \\ imetabolite(M) &\leftarrow reactant(M, R), ireaction(R) \\ ireaction(R) &\leftarrow imetabolite(M), product(M, R), reaction(R, N) \end{aligned} \tag{16}$$

The first rule defines target metabolites as interesting. The second one extends this to metabolites being reactants of interesting reactions. Similar to (9), the last rule states that interesting reactions are those that produce interesting metabolites.

Although the last refinement eliminates (uninteresting) solutions, it preserves minimum ones. Hence, cardinality minimum solutions to the inverse seed problem are obtained by simply adding the following optimization statement.

$$minimize\{seed(M) : not\ target(M)\}$$

Avoiding Side or Seed Metabolites. The elaboration biosynthetic capacities of often subject to further restrictions, for instance, avoiding seed metabolites or certain side products. This has led to the definition of the two variants of the inverse scope problem defined in Section 2.

Both problems are easily addressed in ASP, once a metabolite, m, is declared as being forbidden, viz. $forbidden(m)$:

$$\leftarrow seed(M), forbidden(M)$$
$$\leftarrow dscope(M), forbidden(M)$$

While the first constraint eliminates forbidden metabolites from the seeds, the second rules out unwanted side products.

The complete exclusion of certain metabolites is sometimes to restrictive. To this end, one may replace one or both of the previous integrity constraint by appropriate minimization statements:

$$minimize\{seed(M) : forbidden(M)\}$$
$$minimize\{dscope(M) : forbidden(M)\}$$

Recall that the order of the two statements deteremines their precedence among each other.

Reasoning Modes. The inverse scope problem usually leads to numerous solutions. Cautious reasoning allows us to compute the ultimately essential seeds belonging to all solutions. Also, brave reasoning is of interest because often solutions are similar, so that the union of all seeds in a solution form a pool of potentially relevant nutrients. Finally, in view of the numerous, often unrelated combinatorial sources, an important role is played by projective solution enumeration for eliminating redundant solutions.

4 Experiments

For validating our approach, we investigate the metabolic network of *Escherichia coli* (E.coli). This choice is motivated by the fact that E.coli is a well studied organism, whose metabolic network is of moderate size, consisting of 3645 reactions and 1556 metabolites. Our experiments consider furthermore 94 seed metabolites and 28 target metabolites. The targets and seeds were chosen by our biological partners in view of the fact that E.coli is able to grow when glucose is the only carbon source. Hence, its metabolic network must be able to synthesize all necessary precursors for high-level processes, from glucose and inorganic material [16]. That is why the targets contain all 20 amino acids, the nucleotide phosphates ATP, CTP, GTP and UTP as well as the deoxy forms dATP, dGTP, dUTP and dTTP; and the seeds are only glucose and inorganic metabolites. In fact, all considered targets could be produced by the original E.coli network. This setup allows us to control and vary our experiments by producing draft networks through eliminating reactions from E.coli's original network.

All experiments were run with ASP grounder *gringo* (2.0.2) and ASP solver *clasp* (1.2.0) on a Linux PC with a Core2DuoE6400 processor and 2GB memory. The computation time was limited to 600 seconds, timeouts are shown throughout as "-".

4.1 Metabolic network completion

For our experiments on network completion, biologist provided us with draft networks. The draft networks have been created with biological background knowledge, by removing 50, 100 and 200 reactions from the original E.coli network. Also, derived reactions have been removed by the biologists. This means, for example, that for reversible reactions also the inverse reactions were removed, and for reactions that are generalizations, all subsumed special cases were removed as well. The resulting networks failed

to produce 7, 10, and 20 targets, respectively. As reference network, we have chosen the entire MetaCyc database[6] containing 13882 reactions. This set of reactions spans the search space specified in (5) for metabolic network completion.

In the first set of experiments, we proceed in two steps. First, we compute for each draft network and each target, the minimum number of reactions that need to be added to complete the network. Then, we compute all solutions satisfying this optimality criterion. In fact, in view of the large set of candidate reactions in the reference network, this approach turned out to be superior to a single step approach, enumerating all optimal solution through *clasp*'s branch and bound algorithm. Rather, we invoke *clasp* with the option `--restart-on-model` that restarts after each minimum solution. This makes *clasp* converge much faster to an optimum solution. Once this is found, *clasp* is invoked again for enumerating all solutions satisfying the optimality criterion.

Table 1 summarizes our first set of experiments. The columns headed by E.coli-50, E.coli-100, and E.coli-200, respectively, provide results obtained on the aforementioned draft networks obtained by removing 50, 100 and 200, respectively, reactions from the original E.coli network. The first column identifies the chosen target metabolite. Then, for each draft network, the columns labeled t_{opt} show the time in seconds for computing the minimum number of reactions that need to be added to produce the target. The columns labeled *opt* provide the minimum number of reactions. The columns t_{all} show the time in seconds for computing all optimal solutions and the column *#opt* gives the number of optimal solutions.

For targets that could not be produced by the draft network, the results are either shown in boldface or are timeouts. For target metabolites whose production pathways are not disturbed, the computation time is insignificant. We observe six timeouts, while searching for an optimal completion on the E.coli-50 network. These six target metabolites could not be produced by the draft network in general. Interestingly, those metabolites cannot be produced by all three draft networks, giving us the hint that the pathways for this metabolites are very fragile. Comparing the results for E.coli-50 and E.coli-100, we see that for two targets, the experiments on E.coli-50 timeout, while they could be solved in time on E.coli-100. For E.coli-200, we see 10 experiments timeout, 10 computing the optimal value in time, and for 9 experiments *clasp* finishes computing all optimal solutions in time. This suggests that pathways, which can be disturbed by removing few reactions, are very fragile and hard to reconstruct, while more robust pathways, which are only disturbed when removing lots of reactions, are more easily repaired. For target metabolites whose production pathways are not disturbed, the computation time is insignificant

In our second experiment, we investigate the scalability of our approach in view of the size of the reference network, taking into account the entire set of target metabolites. We created subsets of the MetaCyc network, choosing 10 random samples of 5000, 6000, 7000, 8000, and 9000 reactions. We fixed the draft network by removing 200 reactions from the E.coli network and tried to complete its completion relative to the differently large reference networks. Note that the joint explanation of all 28 targets is much more difficult than just explaining a single target. This is because the restrictions to interesting reactions introduced in Section 3 become less effective when aiming

[6] http://metacyc.org

	E.coli-50				E.coli-100				E.coli-200			
T	t_{opt}	*opt*	t_{all}	*#opt*	t_{opt}	*opt*	t_{all}	*#opt*	t_{opt}	*opt*	t_{all}	*#opt*
1	0.14	0	0.17	1	0.19	0	0.16	1	**368.72**	**1**	**2.17**	**7**
2	0.18	0	0.17	1	0.18	0	0.16	1	**368.52**	**1**	**2.22**	**7**
3	0.16	0	0.16	1	0.17	0	0.18	1	**195.34**	**7**	**304.35**	**135**
4	0.20	0	0.17	1	0.21	0	0.18	1	**42.89**	**3**	**20.40**	**35**
5	0.18	0	0.16	1	0.18	0	0.14	1	0.15	0	0.15	1
6	0.16	0	0.16	1	**159.07**	**2**	**2.53**	**6**	**226.41**	**7**	-	-
7	0.16	0	0.18	1	0.21	0	0.16	1	-	-	-	-
8	0.15	0	0.14	1	0.17	0	0.15	1	**46.39**	**3**	**29.59**	**35**
9	0.16	0	0.19	1	0.14	0	0.15	1	0.15	0	0.14	1
10	0.18	0	0.17	1	0.14	0	0.16	1	0.14	0	0.17	1
11	0.18	0	0.18	1	0.15	0	0.15	1	**26.58**	1	**2.18**	**7**
12	0.14	0	0.16	1	0.15	0	0.16	1	-	-	-	-
13	-	-	-	-	**105.15**	**4**	**12.35**	**1**	-	-	-	-
14	0.17	0	0.18	1	0.15	0	0.17	1	0.16	0	0.14	1
15	0.13	0	0.19	1	0.16	0	0.17	1	0.18	0	0.16	1
16	0.15	0	0.16	1	0.16	0	0.18	1	**367.10**	**1**	**2.20**	**7**
17	0.20	0	0.16	1	-	-	-	-	-	-	-	-
18	-	-	-	-	-	-	-	-	-	-	-	-
19	-	-	-	-	**80.63**	**2**	**5.18**	**3**	-	-	-	-
20	-	-	-	-	-	-	-	-	-	-	-	-
21	0.18	0	0.17	1	0.15	0	0.15	1	0.16	0	0.15	1
22	0.16	0	0.17	1	0.19	0	0.16	1	0.14	0	0.15	1
23	0.17	0	0.14	1	0.16	0	0.15	1	**353.70**	**1**	**2.17**	**1**
24	**37.87**	**3**	**21.28**	**4**	**3.92**	**6**	**29.78**	**5**	-	-	-	-
25	-	-	-	-	-	-	-	-	-	-	-	-
26	-	-	-	-	-	-	-	-	-	-	-	-
27	0.15	0	0.17	1	0.14	0	0.18	1	**46.07**	**3**	**37.08**	**35**
28	0.16	0	0.19	1	-	-	-	-	0.16	0	0.13	1

Table 1. computing optimal completions for E. coli networks

at multiple targets. On the other hand, the identification of a minimum completion producing a maximum set of target is a highly significant question in synthetic biology.

As above, our experiments use a multi-step process. In a first step, we use *clasp* to compute for each reference network the minimum number of reactions needed to complete the network. Once we have computed the optimal value, we continue by computing the reactions essential to all 28 targets, that is, the reactions contained in every answer set satisfying the optimality criterion. This is accomplished by computing the cautious consequences using the option `--cautious` of *clasp*. These reactions are essential for the joint production of all target metabolites. Finally, we use *clasp* as before to enumerate all optimal solutions.

The first line gives the size of the investigated reference network. The columns labeled with *i* identify the instance of the reference network. The column t_{opt} gives the computation time for computing the minimum number of reactions needed for a completion. Column t_c shows the time needed to compute the essential reactions, that

5000					
i	t_{opt}	*opt*	t_c	t_{all}	*#opt*
1	0.25	5	0.19	0.21	72
2	0.23	8	0.16	0.19	36
3	0.20	5	0.16	0.20	3
4	0.14	11	0.20	0.17	12
5	0.18	3	0.16	0.14	24
6	0.39	11	0.26	0.26	24
7	0.18	4	0.18	0.15	2
8	0.18	9	0.22	0.23	60
9	0.27	15	0.19	0.20	16
10	0.15	3	0.16	0.14	6

6000					
i	t_{opt}	*opt*	t_c	t_{all}	*#opt*
1	0.24	4	0.20	0.20	6
2	0.28	4	0.23	0.19	3
3	3.46	16	0.46	0.43	50
4	0.17	7	0.21	0.22	6
5	0.19	2	0.21	0.23	4
6	0.54	17	0.26	0.28	28
7	0.23	5	0.21	0.21	8
8	0.29	19	0.18	0.26	55
9	0.43	9	0.23	0.27	24
10	0.18	3	0.16	0.18	30

7000					
i	t_{opt}	*opt*	t_c	t_{all}	*#opt*
1	105.16	27	3.24	3.33	160
2	-				
3	10.82	19	2.15	1.86	48
4	0.38	5	0.36	0.38	168
5	0.83	14	0.46	0.44	27
6	0.58	7	0.42	1.15	10
7	0.30	2	0.27	0.23	3
8	16.12	14	0.38	0.54	88
9	58.00	17	1.39	0.89	300
10	11.20	18	9.40	8.28	80

8000					
i	t_{opt}	*opt*	t_c	t_{all}	*#opt*
1	265.14	15	274.56	251.34	672
2	1.07	7	0.25	0.30	5
3	5.16	13	1.23	1.13	4
4	1.50	8	0.36	0.35	10
5	0.68	13	0.87	0.98	12
6	78.49	20	48.91	49.67	288
7	195.66	8	1.77	1.58	40
8	5.98	15	3.44	3.63	24
9	9.08	11	0.53	0.59	8
10	0.89	11	0.48	0.42	12

9000					
i	t_{opt}	*opt*	t_c	t_{all}	*#opt*
1	12.34	17	7.45	8.95	18
2	-				
3	28.05	12	11.32	13.99	88
4	-				
5	-				
6	410.76	30	3.88	3.79	14
7	271.02	16	11.13	28.61	2976
8	-				
9	-				
10	-				

Table 2. Completion with 5000,...,9000 reactions.

is, all reactions that are in all minimum completion. The columns labeled with t_{all} show the time needed to compute all optimal solutions. The columns labeled with *#opt* show how many optimal solution have been found. All times are given in seconds.

We observe that the problem is easily handled up to a size of 6000 reactions; all such problems can be solved under a second. Starting with 7000 reactions, we start to obtain computational more demanding problems, and finally a lot of timeouts at size 9000. Notably, our experiments are restricted by a timeout of 10 minutes; existing approaches to network completion usually run simulations over the period of a day. Of course, we have to extend the timeout in a production mode as well. Interestingly, the successful runs show that finding the optimal number of solutions takes most of the computation time; an issue we want to address in the future by biological domain-specific heuristics.

4.2 Inverse scope problem

Last but not least, let us evaluate our approach to the inverse scope problem. As above, we consider the complete E.coli network and try to compute for every target the minimum number of seeds needed to produce it. Once accomplished, we enumerate all minimum sets of seeds.

T	t_{opt}	*opt*	t_{all}	*#opt*
1	2.40	1	14.82	6
2	0.45	1	35.76	12
3	0.38	1	16.02	6
4	28.21	1	25.42	4
5	19.41	2	-	-
6	4.30	2	187.06	50
7	1.29	2	166.73	63
8	15.79	1	17.24	4
9	13.45	1	13.98	4
10	0.89	1	17.00	5
11	0.53	1	25.92	9
12	7.28	1	14.78	4
13	4.78	1	9.88	4
14	13.23	1	7.67	4

T	t_{opt}	*opt*	t_{all}	*#opt*
15	0.32	1	29.58	11
16	0.32	1	31.16	11
17	14.05	1	24.46	1
18	0.28	1	19.66	3
19	10.44	2	-	-
20	23.33	1	27.58	5
21	14.23	1	6.90	4
22	0.37	1	49.79	11
23	-	-	-	-
24	-	-	-	-
25	17.19	1	21.36	4
26	0.55	1	33.24	5
27	19.85	1	15.22	4
28	-	-	-	-

Table 3. Computing minimal seeds for E.coli targets.

Again, we first solve the optimality problem and use *clasp* to compute the minimum number of seeds needed to produce the target metabolite; and in a second step we relaunch *clasp* to compute all optimal solutions.

The first column denotes the target metabolite for whose production the seeds are computed. The second column shows the time in seconds for computing the minimum number of seeds. The third one gives the minimum number of seeds. The fourth column shows the time in seconds for computing all optimal solutions, and the fifth one shows the number of optimal solutions.

The results show that most of the targets can be produced by providing one or two seeds only. Interestingly, we found that only groups of three seeds are needed to produce all 28 targets. We also checked with the cautious reasoning mode for essential seeds, belonging to all minimum solution but none were found. We further used *clasp* with option `--brave` to compute the union of all reactions occurring in optimal solutions and found a set of 136 different metabolites, from which all minimum sets of seeds are taken. Since we are only discriminating the targets among the seeds in (13), we were surprised to find many seeds among the reactants of the reactions producing the targets. However, for more meaningful results, we need more biological knowledge, to exclude more metabolites as seeds.

5 Discussion

The easy characterization of reachability is one of the key features of ASP. We have exploited this to provide a simple yet powerful account of metabolic network synthesis, a crucial application in the elaboration and design of bioprocesses. The distinguishing feature of our ASP-based approach lies in the unique combination of ease of modelling and powerful reasoning modes, supported by efficient solver technology. In fact, existing qualitative approaches to network synthesis are based on stochastic simulations

based on hidden Markov models (cf. [11]), taking several hours to obtain results from the relative frequencies of compounds in the simulations. Unlike this, our approach is complete and thus allows for proving rather than estimating the production of metabolites. Moreover, the various reasoning modes, including the enumeration of optimal solutions as well as cautious and brave reasoning with respect to all or optimal solutions only, respectively, are indispensable in a biological application due to the large number of possible solutions. For instance, cautious reasoning relative to optimal solutions makes us discover the essential nutritions for producing a target metabolite. These reasoning modes together with the high-level specification of metabolic networks make our approach attractive to biologists, given that they can easily elaborate and explore their model "in silico" by means of ASP.

From the perspective of ASP, our application fostered the development of new reasoning modes that were implemented within the ASP solver *clasp* (1.2.0).[7] For one thing, *clasp* allows for optimization techniques not available in any other ASP solver. Of particular interest is the `--restart-on-model` option that restarts after finding a solution (instead of backtracking). This led to a significant increase in converging to an optimal solution. To a turn, we then exploit the options `--opt-all` and `--opt-value` for enumerating all optimal models. Even though the latter can also be addressed by adding an appropriate constraint to the underlying ASP program, the options allow us to leave the underlying program untouched. For another thing, *clasp* allows for computing all brave and cautious consequences[8] by means of a linear[9] number of calls to a solver (internally computing one answer set) rather then enumerating the entire set of answer sets. This is accomplished by consecutive refinements of an internal constraint by appeal to the incremental solving techniques introduced in [17]. This feature is also unique to *clasp*, although a-priori given brave and cautious queries can be decided by other ASP solvers, like *dlv* [18], as well.

Although, to the best of our knowledge, our application is novel in the field of ASP in particular and declarative programming in general, there has been an increasing interest in using ASP technology for addressing biological problems over the last years. Among them, we find [19–22].

Future work will mainly deal with the elaboration of biological domain knowledge for a better narrowing of the solution space and the application of our methodology in the construction of the metabolic network of the recently sequenced green alga *Chlamydomonas reinhardtii*.[10]

References

1. Savageau, M.: Biochemical system analysis: a study of function and design in molecular biology. Addison-Wesley (1976)
2. Kompala, D., Ramkrishna, D., Jansen, N., Tsao, G.: Investigation of bacterial-growth on mixed substrates. Biotechnology and Bioengineering **28**(7) (1986) 1044–1055

[7] http://potassco.sourceforge.net

[8] This is accomplished with options `--brave` and `--cautious`.

[9] That is, linear in the number of atoms.

[10] http://www.goforsys.de

3. Bonarius, H., Schmid, G., Tramper, J.: Flux analysis of underdetermined metabolic networks: The quest for the missing constraints. Trends Biotechnology **15** (1997) 308314
4. Schilling, C., Schuster, S., Palsson, B., Heinrich, R.: Metabolic pathway analysis: Basic concepts and scientific applications in the post-genomic era. Biotechnology progress **15** (1999) 296–303
5. Wildermuth, M.: Metabolic control analysis: biological applications and insights. Genome Biology **1**(6) (2000) 1031.1–1031.5
6. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
7. Nikoloski, Z., Grimbs, S., May, P., Selbig, J.: Metabolic networks are np-hard to reconstruct. Journal of Theoretical Biology **254** (2008) 807–816
8. Nikoloski, Z., Grimbs, S., Selbig, J., Ebenhöh, O.: Hardness and approximability of the inverse scope problem. In: Proceedings WABI'08. Springer (2008) 99–112
9. Ebenhöh, O., Handorf, T., Heinrich, R.: Structural analysis of expanding metabolic networks. Genome Informatics **15**(1) (2004) 35–45
10. Handorf, T., Ebenhöh, O., Heinrich, R.: Expanding metabolic networks: Scopes of compounds, robustness, and evolution. Journal of Molecular Evolution **61**(4) (2005) 498–512
11. Christian, N., May, P., Kempa, S., Handorf, T., Ebenhöh, O.: An integrative approach towards completing genome-scale metabolic networks (2008) Submitted for publication.
12. Handorf, T., Ebenhöh, O., Heinrich, R.: An environmental perspective on metabolism. Journal of Theoretical Biology **252**(3) (2008) 498–512
13. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence **138**(1-2) (2002) 181–234
14. Gebser, M., Kaufmann, B., Schaub, T.: Solution enumeration for projected boolean search problems. In: Proceedings CPAIOR'09. Springer (2009) To appear.
15. Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: Propositional case. Annals of Mathematics and Artificial Intelligence **15**(3-4) (1995) 289–323
16. Christian, N., May, P., Kempa, S., Handorf, T., Ebenhöh, O.: (2008) Personal communication.
17. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Thiele, S.: Engineering an incremental ASP solver. [23] 190–205
18. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM TOCL **7**(3) (2006) 499–562
19. Baral, C., Chancellor, K., Tran, N., Tran, N., Joy, A., Berens, M.: A knowledge based approach for representing and reasoning about signaling networks. In: Proceedings ISMB'04/ECCB'04. (2004) 15–22
20. Dworschak, S., Grell, S., Nikiforova, V., Schaub, T., Selbig, J.: Modeling biological networks by action languages via answer set programming. Constraints **13**(1-2) (2008) 21–65
21. Gebser, M., Schaub, T., Thiele, S., Usadel, B., Veber, P.: Detecting inconsistencies in large biological networks with answer set programming. [23] 130–144
22. Erdem, E., Türe, F.: Efficient haplotype inference with answer set programming. In: Proceedings AAAI'08, AAAI Press (2008) 436–441
23. Garcia de la Banda, M., Pontelli, E., eds.: Proceedings ICLP'08. Springer (2008)