Hybrid Metabolic Network Completion

Clémence Frioux, Torsten Schaub, Sebastian Schellhorn, Anne Siegel, Philipp Wanko

Hybrid Metabolic Network Completion

C. Frioux[1,2], T. Schaub[1,3], S. Schellhorn[3], A. Siegel[2], and P. Wanko[3]

[1] Inria, Rennes, France
[2] IRISA, Université de Rennes 1, France
[3] Universität Potsdam, Germany

Abstract. Metabolic networks play a crucial role in biology since they capture all chemical reactions in an organism. While there are networks of high quality for many model organisms, networks for less studied organisms are often of poor quality and suffer from incompleteness. To this end, we introduced in previous work an ASP-based approach to metabolic network completion. Although this qualitative approach allows for restoring moderately degraded networks, it fails to restore highly degraded ones. This is because it ignores quantitative constraints capturing reaction rates. To address this problem, we propose a hybrid approach to metabolic network completion that integrates our qualitative ASP approach with quantitative means for capturing reaction rates. We begin by formally reconciling existing stoichiometric and topological approaches to network completion in a unified formalism. With it, we develop a hybrid ASP encoding and rely upon the theory reasoning capacities of the ASP system *clingo* for solving the resulting logic program with linear constraints over reals. We empirically evaluate our approach by means of the metabolic network of *Escherichia coli*. Our analysis shows that our novel approach yields greatly superior results than obtainable from purely qualitative or quantitative approaches.

1 Introduction

Among all biological processes occurring in a cell, metabolic networks are in charge of transforming input nutrients into both energy and output nutrients necessary for the functioning of other cells. In other words, they capture all chemical reactions occurring in an organism. In biology, such networks are crucial from a fundamental and technological point of view to estimate and control the capability of organisms to produce certain products. Metabolic networks of high quality exist for many model organisms. In addition, recent technological advances enable their semi-automatic generation for many less studied organisms. However, the resulting metabolic networks are of poor quality, due to error-prone, genome-based construction processes and a lack of (human) resources. As a consequence, they usually suffer from substantial incompleteness. The common fix is to fill the gaps by completing a draft network by borrowing chemical pathways from reference networks of well studied organisms until the augmented network provides the measured functionality.

In previous work [19], we introduced a logical approach to *metabolic network completion* by drawing on the work in [10]. We formulated the problem as a qualitative combinatorial (optimization) problem and solved it with Answer Set Programming (ASP [2]). The basic idea is that reactions apply only if all their reactants are available,

either as nutrients or provided by other metabolic reactions. Starting from given nutrients, referred to as *seeds*, this allows for extending 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. In metabolic network completion, we query a database of metabolic reactions looking for (minimal) sets of reactions that can restore an observed bio-synthetic behavior. This is usually expressed by requiring that certain *target* metabolites are in the scope of some given seeds. For instance, in the follow-up work in [4, 15], we successfully applied our ASP-based approach to the reconstruction of the metabolic network of the macro-algae *Ectocarpus siliculosus*, using the collection of reference networks at `http://metacyc.org`.

Although we evidenced in [16] that our ASP-based method effectively restores the bio-synthetic capabilities of moderately degraded networks, it fails to restore the ones of highly degraded metabolic networks. The main reason for this is that our purely qualitative approach misses quantitative constraints accounting for the law of mass conservation, a major hypothesis about metabolic networks. This law stipulates that each internal metabolite of a network must balance its production rate with its consumption rate. Such rates are given by the weighted sums of all reaction rates consuming or producing a metabolite, respectively. This calculation is captured by the *stoichiometry*[4] of the involved reactions. Hence, the qualitative ASP-based approach fails to tell apart solution candidates with correct and incorrect stoichiometry and therefore reports inaccurate results for highly degraded networks.

We address this by proposing a hybrid approach to metabolic network completion that integrates our qualitative ASP approach with quantitative techniques from *Flux Balance Analysis* (FBA[5] [12]), the dominating quantitative approach for capturing reaction rates in metabolic networks. We accomplish this by taking advantage of recently developed theory reasoning capacities for the ASP system *clingo* [7]. More precisely, we use an extension of *clingo* with linear constraints over reals, as dealt with in Linear Programming (LP [5]). This extension provides us with an extended ASP modeling language as well as a generic interface to alternative LP solvers, viz. *cplex* and *lpsolve*, for dealing with linear constraints. We empirically evaluate our approach by means of the metabolic network of *Escherichia coli*. Our analysis shows that our novel approach yields superior results than obtainable from purely qualitative or quantitative approaches. Moreover, our hybrid application provides a first evaluation of the theory extensions of the ASP system *clingo* with linear constraints over reals in a non-trivial setting.

2 Metabolic Network Completion

We represent a *metabolic network* as a labeled directed bipartite graph $G = (R \cup M, E, s)$, where R and M are sets of nodes standing for *reactions* and *metabolites*, respectively. When $(m, r) \in E$ or $(r, m) \in E$ for $m \in M$ and $r \in R$, the metabolite m is called a *reactant* or *product* of reaction r, respectively. More formally, for any $r \in R$, define $rcts(r) = \{m \in M \mid (m, r) \in E\}$ and $prds(r) = \{m \in M \mid (r, m) \in E\}$. The

[4] See also `https://en.wikipedia.org/wiki/Stoichiometry`.
[5] See also `https://en.wikipedia.org/wiki/Flux_balance_analysis`.

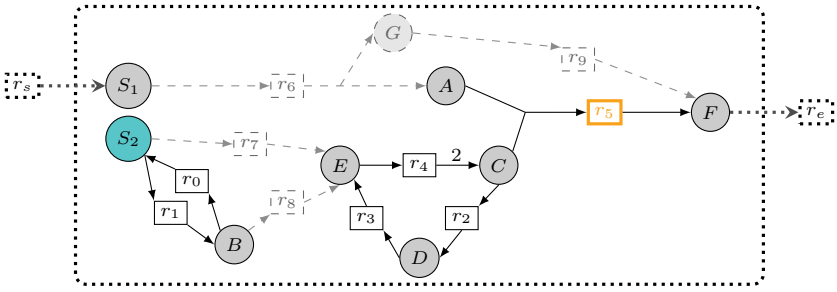

Fig. 1. Example of a metabolic network

edge labeling $s : E \to \mathbb{R}$ gives the stoichiometric coefficients of a reaction's reactants and products, respectively. Finally, the activity rate of reactions is bound by lower and upper bounds, denoted by $lb_r \in \mathbb{R}_0^+$ and $ub_r \in \mathbb{R}_0^+$ for $r \in R$, respectively. Whenever clear from the context, we refer to metabolic networks with G (or G', etc) and denote the associated reactions and metabolites with M and R (or M', R' etc), respectively.

We distinguish a set $S \subseteq M$ of metabolites as initiation *seeds*, that is, compounds initially present due to experimental evidence. Another set of metabolites is assumed to be activated by default. These *boundary metabolites* are defined as: $S_b(G) = \{m \in M \mid r \in R, m \in prds(r), rcts(r) = \emptyset\}$. For simplicity, we assume that all boundary compounds are seeds: $S_b(G) \subseteq S$. Note that concepts like reachability and activity in network completion are independent of this assumption.

For illustration, consider the metabolic network in Fig. 1 and ignore the shaded part. The network consists of 8 reactions, r_s, r_e and r_0 to r_5, and 8 metabolites, $A, \ldots, F$, S_1, S_2. Here, $S = \{S_1, S_2\}$, S_1 being the only boundary compound of the network. Consider reaction $r_4 : E \to 2C$ transforming one unit of E into two units of C. We have $rcts(r_4) = \{E\}$, $prds(r_4) = \{C\}$, along with $s(E, r_4) = 1$and $s(r_4, C) = 2$.

In biology, several concepts have been introduced to model the activation of reaction fluxes in metabolic networks, or to synthesize metabolic compounds. To model this, we introduce a function $active$ that given a metabolic network G takes a set of seeds $S \subseteq M$ and returns a set of activated reactions $active_G(S) \subseteq R$. With it, *metabolic network completion* is about ensuring that a set of target reactions is activated from seed compounds in S by possibly extending the metabolic network with reactions from a reference network (cf. shaded part in Fig. 1).

Formally, given a metabolic network $G = (R \cup M, E, s)$, a set $S \subseteq M$ of seed metabolites such that $S_b(G) \subseteq S$, a set $R_T \subseteq R$ of target reactions, and a reference network $(R' \cup M', E', s')$, the *metabolic network completion problem* is to find a set $R'' \subseteq R' \setminus R$ of reactions of minimal size such that $R_T \subseteq active_{G''}(S)$ where[6]

$$G'' = ((R \cup R'') \cup (M \cup M''), E \cup E'', s'') \,, \tag{1}$$

$$M'' = \{m \in M' \mid r \in R'', m \in rcts(r) \cup prds(r)\} \,, \tag{2}$$

$$E'' = E' \cap ((M'' \times R'') \cup (R'' \times M'')), \text{ and} \tag{3}$$

[6] Since s, s' have disjoint domains we view them as relations and compose them by union.

$$s'' = s \cup s' \,. \tag{4}$$

We call R'' a *completion* of $(R \cup M, E, s)$ from $(R' \cup M', E', s')$ wrt S and R_T.

Our concept of activation allows us to capture different biological paradigms. Accordingly, different formulations of metabolic network completion can be characterized: the stoichiometric, the relaxed stoichiometric, the topological, and the hybrid one. We elaborate upon their formal characterizations in the following sections.

Stoichiometric Metabolic Network Completion. The first activation semantics has been introduced in the context of Flux Balance Analysis capturing reaction flux distributions of metabolic networks at steady state. In this paradigm, each reaction r is associated with a *metabolic flux value*, expressed as a real variable v_r confined by the minimum and maximum rates:

$$lb_r \leq v_r \leq ub_r \qquad \text{for } r \in R \tag{5}$$

Flux distributions are formalized in terms of a system of equations relying on the stoichiometric coefficients of reactions. Reaction rates are governed by the *law of mass conservation* under a steady state assumption, that is, the input and output rates of reactions consuming and producing a metabolite are balanced:

$$\textstyle\sum_{r \in R} s(r,m) \cdot v_r + \sum_{r \in R} -s(m,r) \cdot v_r = 0 \qquad \text{for } m \in M \tag{6}$$

Given a target reaction $r_T \in R_T$, a metabolic network $G = (R \cup M, E, s)$ and a set of seeds S, *stoichiometric activation* is defined as follows:

$$r_T \in \mathit{active}^s_G(S) \text{ iff } v_{r_T} > 0 \text{ and (5) and (6) hold for } M \text{ and } R.$$

Note that the condition $v_{r_T} > 0$ strengthens the flux condition for $r_T \in R$ in the second part. More generally, observe that activated target reactions are not directly related to the network's seeds S. However, the activation of targets highly depends on the boundary metabolites in $S_b\,(G)$ for which (6) is always satisfied and thus initiates the fluxes.

To solve metabolic network completion with flux-balance activated reactions, Linear Programming can be used to maximize the flux rate v_{r_T} provided that the linear constraints are satisfied. This problem turns out to be hard to solve in practice and existing approaches scale poorly to real-life applications (cf. [13]).

This motivated the use of approximate methods. The relaxed problem is obtained by weakening the mass-balance equation (6) as follows:

$$\textstyle\sum_{r \in R} s(r,m) \cdot v_r + \sum_{r \in R} -s(m,r) \cdot v_r \geq 0 \qquad \text{for } m \in M \tag{7}$$

This lets us define the concept of *relaxed stoichiometric activation*:

$$r_T \in \mathit{active}^r_G(S) \text{ iff } v_{r_T} > 0 \text{ and (5) and (7) hold for } M \text{ and } R.$$

The resulting problem can now be efficiently solved with Linear Programming [18]. Note however that for strict steady-state modeling an *a posteriori* verification of solutions is needed to warrant the exact mass-balance equation (6).

In our draft network G, consisting of all bold nodes and edges depicted in Fig. 1 (viz. reactions r_s, r_e and r_0 to r_5 and metabolites $A, \ldots, F$, S_1 and S_2 and r_5 the single target

reaction) and the reference network G', consisting of the shaded part of Fig 1, (viz. reactions r_6 to r_9 and metabolite G) a strict stoichiometry-based completion aims to obtain a solution with $r_5 \in active^s_{G''}(\{S_1, S_2\})$ where v_{r_5} is maximal. This can be achieved by adding the completion $R''_1 = \{r_6, r_9\}$. The cycle made of compounds E, C, D is already balanced and notably self-activated. Such self-activation of cyclic pathways is an inherent problem of purely stoichiometric approaches to network completion. This is a drawback of the semantics since the effective activation of the cycle requires the additional (and unchecked) condition that at least one of the compounds was present as the initial state of the system [16]. The instance of Equation (6) controlling the reaction rates related to metabolite C is $2 \cdot v_{r_4} - v_{r_2} - v_{r_5} = 0$.

Existing systems addressing strict stoichiometric network completion either cannot guarantee optimal solutions [11] or do not support a focus on specific target reactions [21]. Other approaches either partially relax the problem [22] or solve the relaxed problem based on Equation (7), like the popular system *gapfill* [18].

Topological Metabolic Network Completion. A qualitative approach to metabolic network completion relies on the topology of networks for capturing the activation of reactions. Given a metabolic network G, a reaction $r \in R$ is *activated* from a set of seeds S if all reactants in $rcts(r)$ are reachable from S. Moreover, a metabolite $m \in M$ is *reachable* from S if $m \in S$ or if $m \in prds(r)$ for some reaction $r \in R$ where all $m' \in rcts(r)$ are reachable from S. The *scope* of S, written $\Sigma_G(S)$, is the closure of metabolites reachable from S. In this setting, *topological activation* of reactions from a set of seeds S is defined as follows:

$$r_T \in active^t_G(S) \text{ iff } rcts(r_T) \subseteq \Sigma_G(S).$$

Note that this semantics avoids self-activated cycles by imposing an external entry to all cycles. The resulting network completion problem can be expressed as a combinatorial optimization problem and effectively solved with ASP [19].

For illustration, consider again the draft and reference networks G and G' in Fig. 1. We get $\Sigma_G(\{S_1, S_2\}) = \{S_1, S_2, B\}$, indicating that target reaction r_5 is not activated from the seeds with the draft network because A and C are not reachable. This changes once the network is completed. Valid minimal completions are $R''_2 = \{r_6, r_7\}$ and $R''_3 = \{r_6, r_8\}$ because $r_5 \in active^t_{G''_i}(\{S_1, S_2\})$ since $\{A, C\} \subseteq \Sigma_{G''_i}(\{S_1, S_2\})$ for all extended networks G''_i obtained from completions R''_i of G for $i \in \{2, 3\}$. Relevant elements from the reference network are given in dashed gray.

Hybrid Metabolic Network Completion. The idea of hybrid metabolic network completion is to combine the two previous activation semantics: the topological one accounts for a well-founded initiation of the system from the seeds and the stoichiometric one warrants its mass-balance. We thus aim at network completions that are both topologically functional and flux balanced (without suffering from self-activated cycles). More precisely, a reaction $r_T \in R_T$ is *hybridly activated* from a set S of seeds in a network G, if both criteria apply:

$$r_T \in active^h_G(S) \text{ iff } r_T \in active^s_G(S) \text{ and } r_T \in active^t_G(S)$$

Applying this to our example in Fig. 1, we get the (minimal) hybrid solutions $R''_4 = \{r_6, r_7, r_9\}$ and $R''_5 = \{r_6, r_8, r_9\}$. Both (topologically) initiate paths of reactions from

the seeds to the target, ie. $r_5 \in active^t_{G_i''}(\{S_1, S_2\})$ since $\{A, C\} \subseteq \Sigma_{G_i''}(\{S_1, S_2\})$ for both extended networks G_i'' obtained from completions R_i'' of G for $i \in \{4, 5\}$. Both solutions are as well stoichiometrically valid and balance the amount of every metabolite, hence we also have $r_5 \in active^s_{G_i''}(\{S_1, S_2\})$.

3 Answer Set Programming with Linear Constraints

For encoding our hybrid problem, we rely upon the theory reasoning capacities of the ASP system *clingo* that allows us to extend ASP with linear constraints over reals (as addressed in Linear Programming). We confine ourselves below to features relevant to our application and refer the interested reader for details to [7].

As usual, a *logic program* consists of *rules* of the form

$$\texttt{a}_0 \texttt{ :- a}_1\texttt{,...,a}_m\texttt{,not a}_{m+1}\texttt{,...,not a}_n$$

where each $\texttt{a}_i$ is either a *(regular) atom* of form $\texttt{p(t}_1\texttt{,...,t}_k\texttt{)}$ where all $\texttt{t}_i$ are terms or a *linear constraint atom* of form[7] '$\texttt{\&sum\{a}_1\texttt{*x}_1\texttt{;...;a}_l\texttt{*x}_l\texttt{\} <= k}$' that stands for the linear constraint $a_1 \cdot x_1 + \cdots + a_l \cdot x_l \leq k$. All $\texttt{a}_i$ and $\texttt{k}$ are finite sequences of digits with at most one dot[8] and represent real-valued coefficients a_i and k. Similarly all $\texttt{x}_i$ stand for the real-valued variables x_i. As usual, `not` denotes (default) *negation*. A rule is called a *fact* if $n = 0$.

Semantically, a logic program induces a set of *stable models*, being distinguished models of the program determined by stable models semantics [9]. Such a stable model X is an *LC-stable model* of a logic program P,[9] if there is an assignment of reals to all real-valued variables occurring in P that (i) satisfies all linear constraints associated with linear constraint atoms in P being in X and (ii) falsifies all linear constraints associated with linear constraint atoms in P being not in X. For instance, the (non-ground) logic program containing the fact '`a("1.5").`' along with the rule '`&sum{R*x} <= 7 :- a(R).`' has the stable model

$\{$`a("1.5")`, `&sum{"1.5"*x}<=7`$\}$.

This model is LC-stable since there is an assignment, e.g. $\{x \mapsto 4.2\}$, that satisfies the associated linear constraint '$1.5 * x \leq 7$'. We regard the stable model along with a satisfying real-valued assignment as a solution to a logic program containing linear constraint atoms.

To ease the use of ASP in practice, several extensions have been developed. First of all, rules with variables are viewed as shorthands for the set of their ground instances. Further language constructs include *conditional literals* and *cardinality constraints* [20]. The former are of the form $\texttt{a:b}_1\texttt{,...,b}_m$, the latter can be written as $\texttt{s\{d}_1\texttt{;...;d}_n\texttt{\}t}$, where $\texttt{a}$ and $\texttt{b}_i$ are possibly default-negated (regular) literals and each $\texttt{d}_j$ is a conditional literal; $\texttt{s}$ and $\texttt{t}$ provide optional lower and upper bounds on the number of satisfied literals in the cardinality constraint. We refer to $\texttt{b}_1\texttt{,...,b}_m$ as a *condition*. The practical value of both constructs becomes apparent when used with variables. For instance, a

[7] In *clingo*, theory atoms are preceded by '`&`'.

[8] In the input language of *clingo*, such sequences must be quoted to avoid clashes.

[9] This corresponds to the definition of T-stable models using a *strict* interpretation of theory atoms [7], and letting T be the theory of linear constraints over reals.

conditional literal like `a(X):b(X)` in a rule's antecedent expands to the conjunction of all instances of `a(X)` for which the corresponding instance of `b(X)` holds. Similarly, `2{a(X):b(X)}4` is true whenever at least two and at most four instances of `a(X)` (subject to `b(X)`) are true. Finally, objective functions minimizing the sum of weights w_i subject to condition c_i are expressed as `#minimize{`w_1`:`c_1`;...;`w_n`:`c_n`}`.

In the same way, the syntax of linear constraints offers several convenience features. As above, elements in linear constraint atoms can be conditioned, viz.

'`&sum{`a_1`*`x_1`:`c_1`;...;`a_l`*`x_l`:`c_n`} <= y`'

where each c_i is a condition. Moreover, the theory language for linear constraints offers a domain declaration for real variables, '`&dom{lb..ub} = x`' expressing that all values of `x` must lie between `lb` and `ub`. And finally the maximization (or minimization) of an objective function can be expressed with `&maximize{`a_1`*`x_1`:`c_1`;...;`a_l`*`x_l`:`c_n`}` (by `minimize`). The full theory grammar for linear constraints over reals is available at `https://potassco.org`.

4 Solving Hybrid Metabolic Network Completion

In this section, we present our hybrid approach to metabolic network completion. We start with a factual representation of problem instances. A metabolic network G with a typing function $t : M \cup R \to \{\texttt{d},\texttt{r},\texttt{s},\texttt{t}\}$, indicating the origin of the respective entities, is represented as follows:

$$\begin{aligned} F(G,t) = \quad & \{\texttt{metabolite(}m\texttt{,}t(m)\texttt{)} \mid m \in M\} \\ \cup\ & \{\texttt{reaction(}r\texttt{,}t(r)\texttt{)} \mid r \in R\} \\ \cup\ & \{\texttt{bounds(}r\texttt{,}lb_r\texttt{,}ub_r\texttt{)} \mid r \in R\} \cup \{\texttt{objective(}r\texttt{,}t(r)\texttt{)} \mid r \in R\} \\ \cup\ & \{\texttt{reversible(r)} \mid r \in R, \mathit{rcts}(r) \cap \mathit{prds}(r) \neq \emptyset\} \\ \cup\ & \{\texttt{rct(}m\texttt{,}s(m,r)\texttt{,r,}t(r)\texttt{)} \mid r \in R, m \in \mathit{rcts}(r)\} \\ \cup\ & \{\texttt{prd(}m\texttt{,}s(r,m)\texttt{,r,}t(r)\texttt{)} \mid r \in R, m \in \mathit{prds}(r)\} \end{aligned}$$

While most predicates should be self-explanatory, we mention that `reversible` identifies bidirectional reactions. Only one direction is explicitly represented in our fact format. The four types `d`, `r`, `s`, and `t` tell us whether an entity stems from the **d**raft or **r**eference network, or belongs to the **s**eeds or **t**argets.

In a metabolic network completion problem, we consider a draft network $G = (R \cup M, E, s)$, a set S of seed metabolites, a set R_T of target reactions, and a reference network $G' = (R' \cup M', E', s')$. An instance of this problem is represented by the set of facts $F(G,t) \cup F(G',t')$. In it, a key role is played by the typing functions that differentiate the various components:

$$t(n) = \begin{cases} \texttt{d}, \text{ if } n \in (M \setminus (T \cup S)) \cup (R \setminus (R_{S_b} \cup R_T)) \\ \texttt{s}, \text{ if } n \in S \cup R_{S_b} \\ \texttt{t}, \text{ if } n \in T \cup R_T \end{cases} \qquad \text{and} \quad t'(n) = \texttt{r},$$

where $T = \{m \in \mathit{rcts}(r) \mid r \in R_T\}$ is the set of target metabolites and $R_{S_b} = \{r \in R \mid m \in S_b(G), m \in \mathit{prds}(r)\}$ is the set of reactions related to boundary seeds.

```
1  edge(R,M,N,T) :- reaction(R,T), rct(M,_,R,T), prd(N,_,R,T).
2  edge(R,M,N,T) :- reaction(R,T), rct(N,_,R,T), prd(M,_,R,T), reversible(R).

4  scope(M,d) :- metabolite(M,s).
5  scope(M,d) :- edge(R,_,M,T), T!=r, scope(N,d):edge(R,N,_,T'), N!=M, T'!=r.

7  scope(M,x) :- scope(M,d).
8  scope(M,x) :- edge(R,_,M,_), scope(N,x):edge(R,N,_,_), N!=M.

10 { completion(R) : edge(R,M,N,r), scope(N,x), scope(M,x) }.

12 scope(M,c) :- scope(M,d).
13 scope(M,c) :- edge(R,_,M,T), T!=r, scope(N,c):edge(R,N,_,T'), T'!=r, N!=M.
14 scope(M,c) :- completion(R), edge(R,_,M,r), scope(N,c):edge(R,N,_,r), N!=M.

16 :- metabolite(M,t), not scope(M,c).

18 &dom{L..U} = R :- bounds(R,L,U).

20 &sum{ IS*IR : prd(M,IS,IR,T), T!=r;  IS'*IR' : prd(M,IS',IR',r), completion(IR');
21       -OS*OR : rct(M,OS,OR,T), T!=r; -OS'*OR' : rct(M,OS',OR',r), completion(OR')
22      } = "0" :- metabolite(M,_).

24 &sum{ R } > "0" :- reaction(R,t).

26 &maximize{   R : objective(R,t) }.
27 #minimize{ 1,R : completion(R)  }.
```

Listing 1. Encoding of hybrid metabolic network completion

Our encoding of hybrid metabolic network completion is given in Listing 1. Roughly, the first 10 lines lead to a set of candidate reactions for completing the draft network. Their topological validity is checked in lines 12–16 with regular ASP, the stoichiometric one in lines 18–24 in terms of linear constraints. (Lines 1–16 constitute a revision of the encoding in [19].) The last two lines pose a hybrid optimization problem, first minimizing the size of the completion and then maximizing the flux of the target reactions.

In more detail, we begin by defining the auxiliary predicate `edge`/4 representing directed edges between metabolites connected by a reaction. With it, we calculate in Line 4 and 5 the scope $\Sigma_G(S)$ of the **d**raft network G from the seed metabolites in S; it is captured by all instances of `scope(M,d)`. This scope is then e**x**tended in Line 7/8 via the reference network G' to delineate all possibly producible metabolites. We draw on this in Line 10 when choosing the reactions R'' of the completion (cf. Section 2) by restricting their choice to reactions from the reference network whose reactants are producible. This amounts to a topological search space reduction.

The reactions in R'' are then used in lines 12–14 to compute the scope $\Sigma_{G''}(S)$ of the **c**ompleted network. And R'' constitutes a topologically valid completion if all targets in T are producible by the expanded draft network G'': Line 16 checks whether $T \subseteq \Sigma_{G''}(S)$ holds, which is equivalent to $R_T \subseteq \mathit{active}^t_{G''}(S)$. Similarly, R'' is checked for stoichiometric validity in lines 18–24. For simplicity, we associate reactions with their rate and let their identifiers take real values. Accordingly, Line 18 accounts for (5) by imposing lower and upper bounds on each reaction rate. The mass-balance equation (6) is enforced for each metabolite `M` in lines 20–22; it checks whether the sum of products of stoichiometric coefficients and reaction rates equals zero, viz. `IS*IR`, `-OS*OR`, `IS'*IR'`, and `-OS'*OR'`. Reactions `IR`, `OR` and `IR'`, `OR'` belong to the

	F(BB)		F(USC)		F(BB+USC)		VERIFIED		
DEGRADATION	#SOLS	#OPTS	#SOLS	#OPTS	#SOLS	#OPTS	F(BB+USC)	M	G
10% (900)	**900**	**900**	892	892	900	900	**900**	660	56
20% (900)	**830**	669	793	**769**	867	814	**867**	225	52
30% (900)	**718**	88	461	**344**	780	382	**780**	61	0
all (2700)	**2448**	1657	2146	**2005**	2547	2096	**2547**	946	108

Table 1. Comparison of qualitative results.

	F(BB)		F(USC)	
CONFIGURATION	T	TO	T	TO
DEFAULT	377	121	**190**	46
CORE-50	358	109	230	75
CORE-0	**350**	**96**	233	76
PROP-50	363	112	226	69
PROP-100	386	105	360	139
HEURISTIC	542	178	252	**28**

Table 2. Comparison of system options.

	F(VBS)		VERIFIED		
DEGRADATION	#SOLS	#OPTS	F(VBS)	M	G
10% (900)	900	900	**900**	660	56
20% (900)	896	855	**896**	225	52
30% (900)	848	575	**848**	61	0
40% (900)	681	68	**681**	29	0
all (3600)	3325	2398	**3325**	975	108

Table 3. Results using best system options.

draft and reference network, respectively, and correspond to $R \cup R''$. Finally, by enforcing $r_T > 0$ for $r_T \in R_T$ in Line 24, we make sure that $R_T \subseteq active^s_{G''}(S)$.

In all, our encoding ensures that the set R'' of reactions chosen in Line 10 induces an augmented network G'' in which all targets are activated both topologically as well as stoichiometrically, and is optimal wrt the hybrid optimization criteria.

5 System and Experiments

In this section, we introduce *fluto*, our new system for hybrid metabolic network completion, and empirically evaluate its performance. The system relies on the hybrid encoding described in Section 4 along with the hybrid solving capacities of *clingo* [7] for implementing the combination of ASP and LP. We use *clingo* 5.2.0incorporating as LP solvers either *cplex* 12.7.0.0 or *lpsolve* 5.5.2.5 via their respective Python interfaces. We describe the details of the underlying solving techniques in a separate paper and focus below on application-specific aspects.

The output of *fluto* consists of two parts. First, the completion R'', given by instances of predicate `completion`, and second, an assignment of floats to (metabolic flux variables v_r for) all $r \in R \cup R''$. In our example, we get

$R'' = \{\texttt{completion}(r_6), \texttt{completion}(r_8), \texttt{completion}(r_9)\}$

and $\{r_s = 49999.5, r_9 = 49999.5, r_3 = 49999.5, r_2 = 49999.5, r_e = 99999.0, r_6 = 49999.5, r_5 = 49999.5, r_4 = 49999.5\}$. Variables assigned 0 are omitted. Note the flux value $r_8 = 0$ even though $r_8 \in R''$. This is to avoid the self-activation of cycle C, D and E. By choosing r_8, we ensure that the cycle has been externally initiated at some point but activation of r_8 is not necessary at the current steady state.

We analyze (i) the quality of *fluto*'s approach to metabolic network completion and (ii) the impact of different system configurations. To have a realistic setting, we use degradations of a functioning metabolic network of *Escherichia coli* [17] comprising 1075 reactions. The network was randomly degraded by 10, 20, and 30 percent, creating

10 networks for each degradation by removing reactions until the target reactions were inactive according to *Flux Variability Analysis* [3]. 90 target reactions with varied reactants were randomly chosen for each network, yielding 2700 problem instances in total. The reference network consists of reactions of the original metabolic network.

We ran each benchmark on a Xeon E5520 2.4 GHz processor under Linux limiting RAM to 20 GB. At first, we investigate two alternative optimization strategies for computing completions of minimum size. The first one, *branch-and-bound* (BB), iteratively produces solutions of better quality until the optimum is found and the other, *unsatisfiable core* (USC), relies on successively identifying and relaxing unsatisfiable cores until an optimal solution is obtained. Note that we are not only interested in optimal solutions but if unavailable also solutions activating target reactions without trivially restoring the whole reference network. In *clingo*, BB naturally produces these solutions in contrast to USC. Therefore, we use USC with stratification [1], which provides at least some suboptimal solutions. Each obtained best solution was checked with *cobrapy* 0.3.2 [6], a renowned system implementing an FBA-based gold standard (for verification only).

Table 1 gives the number of solutions (#SOLS) and optima (#OPTS) obtained by *fluto* (F) in its default setting within 20 minutes for BB, USC and the best of both (BB+USC), individually for each DEGRADATION and over**all**. For 94.3% of the instances *fluto*(BB+USC) found a solution within the time limit and 82.3% of them were optimal. We observe that BB provides overall more useful solutions but USC acquires more optima, which was to be expected by the nature of the optimization techniques. Additionally, each technique finds solutions to problem instances where the other exceeds the time limit, underlining the merit of using both in tandem. Column VERIFIED compares the quality of solutions provided by *fluto*, *meneco* 1.4.3 (M) [16] and *gapfill*[10](G) [18]. Both *meneco* and *gapfill* are systems for metabolic network completion. While *meneco* pursues the topological approach, *gapfill* applies the relaxed stoichiometric variant using Equation (7). The numbers represent how many problem instances had verified solutions for each system.[11] All solutions found by *fluto* could be verified by *cobrapy*. In detail, *fluto* found a smallest set of reactions completing the draft network for 77.6%, a suboptimal solution for 16.7%, and no solution for 5.6% of the problem instances. In comparison, for *meneco* 35.0%, and for *gapfill* merely 4.0% of its solutions passed verification. The ignorance of *meneco*regarding stoichiometry leads to possibly unbalanced networks, which particularly outcrops for higher degradation. The simplified view of *gapfill*in terms of stoichiometry misguides the search for possible completions and eventually leads to unbalanced networks. Moreover, *gapfill*'s ignorance of network topology results in self-activated cycles. By exploiting both topology and stoichiometry, *fluto* avoids such cycles and scales much better with increasing reference network size and degradation of the network.

The configuration space of *fluto* is huge. In addition to its own parameters, the ones of *clingo* and the respective LP solver amplify the number of options. We thus focus on distinguished features revealing an impact in our experiments. First, the *fluto* option CORE-n invokes the irreducible inconsistent set algorithm [14] whenever $n\%$ of atoms

[10] Update of 2011-09-23 see http://www.maranasgroup.com/software.htm

[11] The results for *meneco* and *gapfill* are taken from previous work [16], where they were run to completion with *no* time limit.

are decided. This algorithm extracts a minimal set of conflicting linear constraints for a given conflict. Second, PROP-n controls the frequency of LP propagation: the consistency of linear constraints is only checked if $n\%$ of atoms are decided. Finally, HEURISTIC allows for using *clingo*'s domain-specific heuristics. Such heuristics are expressed in the input language with the directive `#heuristic`. In *fluto*, we use the statement

```
#heuristic completion(R) : interesting(R). [1,true]
```

to make the solver first decide interesting reactions and assign them true. A reaction of G' is `interesting` if it is on a direct path in $G' \cup G$ from a seed metabolite to a target metabolite. The DEFAULT is to use CORE-100, PROP-0, disable HEURISTIC, and use LP solver *cplex*. This allows us to detect conflicts among the linear constraints as soon as possible and only perform expensive conflict analysis on the full assignment.

For our experiments, we selected at random three networks with at least 20 instances for which BB and USC could find the optimum in 100 to 600 seconds. With the resulting 270 medium to hard instances, we compared DEFAULT as baseline, $n \in \{0, 50, 100\}$ for CORE-n and PROP-n, respectively, and HEURISTIC, limiting time to 600 seconds. Table 2 gives the overall average time in seconds (T) and number of timeouts (TO). The first column reflects the CONFIGURATIONs. We focus on the impact of distinguished parameters wrt the default setting and leave a more exhaustive exploration to future work.[12] Overall, USC performs best as regards average time, and USC and HEURISTIC yield the least number of timeouts. BB works well with frequent conflict analysis (CORE-0), while it weakens USC's performance. On the other hand, unlike BB, USC favors frequent theory propagation (DEFAULT). BB learns weaker constraints while optimizing only pertaining to the best known bound, thus the improvement step is less constraint compared to USC. Due to this, conflicts are more likely to appear later on and be of less quality, enhancing the potential of conflict analysis and hampering the usefulness of frequent LP propagation. USC on the other hand, aims at quickly identifying unsatisfiable partial assignments and learning structural constraints building upon each other, which is enhanced by frequent conflict detection. Thus, higher quality conflicts are likely detected earlier where conflict minimization has less potential and produces overhead. HEURISTIC reduces performance for BB. Even though the bound of the initial solution might be lower, the solver derives no additional information from this bound, and the heuristic hurts the unsatisfiability proof at the end. USC works surprisingly well with HEURISTIC. Since all heuristically modified variables are part of the optimization, the first USC optimization step disregards the heuristics entirely because no reactions from the reference network are selected. Afterward, the learned unsatisfiable core is relaxed by choosing heuristically modified reactions first. This might lead to unsatisfiable cores with higher quality since they arguably include relevant reactions. Iterating this process, the solver appears to learn shortcuts, fixing sets of important reactions that have to be included in solutions, thus reducing the complexity of the remaining search. Note that instead of modifying all atoms in the optimization statement which was shown to be unsuccessful in [8], we specifically select topologically relevant reactions.

Finally, we take the best configurations and examine how *fluto* scales on harder instances. To this end, we use configurations with bold rows in Table 2. We rerun the first experiment after adding 900 instances degraded by 40% (Table 3). F(VBS) denotes

[12] Also, we do not present results of *lpsolve* since it produced inferior results.

the virtual best results, meaning for each problem instance the best known solution among the three configurations was verified. For 20% and 30% degradation, we obtain additional 29 and 68 solutions and 41 and 193 optima, respectively. Overall, we find solutions for 92.4% out of the 3600 instances and 72.1% of them are optimal. The number of solutions decreases slightly and the number of optima more drastically with higher degradation. While again 100% of *fluto*'s solutions could be verified, only 27.1% and 3% are obtained for *meneco* and *gapfill*, respectively.

6 Discussion

We presented the first hybrid approach to metabolic network completion by combining topological and stoichiometric constraints in a uniform setting. To this end, we elaborated a formal framework capturing different semantics for the activation of reactions. Based upon these formal foundations, we developed a hybrid ASP encoding reconciling disparate approaches to network completion. The resulting system, *fluto*, thus combines the advantages of both approaches and yields greatly superior results compared to purely quantitative or qualitative existing systems. Our experiments show that *fluto* scales to more highly degraded networks and produces useful solutions in reasonable time. In fact, all of *fluto*'s solutions passed the biological gold standard. The exploitation of the network's topology guides the solver to more likely completion candidates, and furthermore avoids self-activated cycles, as obtained in FBA-based approaches. Also, unlike other systems, *fluto* allows for establishing optimality and address the strict stoichiometric completion problem without approximation.

fluto takes advantage of the hybrid reasoning capacities of the ASP system *clingo* for extending logic programs with linear constraints over reals. This provides us with a practically relevant application scenario for evaluating this hybrid form of ASP. To us, the most surprising empirical result was the observation that domain-specific heuristic allow for boosting unsatisfiable core based optimization. So far, such heuristics have only been known to improve satisfiability-oriented reasoning modes, and usually hampered unsatisfiability-oriented ones (cf. [8]).

Acknowledgments This work was partially funded by DFG grant SCHA 550/9 and 11.

References

1. C. Ansótegui, M. Bonet, J. Levy. SAT-based MaxSAT algorithms. *Artificial Intelligence*, 196:77–105, 2013.
2. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge, 2003.
3. S. Becker, A. Feist, M. Mo, G. Hannum, B. Palsson, M. Herrgard. Quantitative Prediction of Cellular Metabolism with Constraint-based Models: The COBRA Toolbox. *Nature Protocols*, 2(3):727–738, 2007.
4. G. Collet, D. Eveillard, M. Gebser, S. Prigent, T. Schaub, A. Siegel, S. Thiele. Extending the metabolic network of Ectocarpus siliculosus using answer set programming. *Proceedings LPNMR*, 245–256. Springer, 2013.
5. G. Dantzig. *Linear Programming and Extensions*. Princeton, 1963.

6. A. Ebrahim, J. Lerman, B. Palsson, D. Hyduke. COBRApy: COnstraints-Based Reconstruction and Analysis for Python. *BMC Systems Biology*, 7:74, aug 2013.
7. M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, P. Wanko. Theory solving made easy with clingo 5. *Technical Comm. ICLP*, 2:1–2:15. OASIcs, 2016.
8. M. Gebser, R. Kaminski, B. Kaufmann, J. Romero, T. Schaub. Progress in clasp series 3. *Proceedings LPNMR*, 368–383. Springer, 2015.
9. M. Gelfond, V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
10. T. Handorf, O. Ebenhöh, R. Heinrich. Expanding metabolic networks: Scopes of compounds, robustness, and evolution. *J. of Molec. Evolution*, 61(4):498–512, 2005.
11. M. Latendresse. Efficiently gap-filling reaction networks. *BMC bioinformatics*, 15(1):225, 2014.
12. C. Maranas, A. Zomorrodi. *Optimiz. methods in metabolic networks*. Wiley, 2016.
13. J. Orth, B. Palsson. Systematizing the generation of missing metabolic knowledge. *Biotechnology and bioengineering*, 107(3):403–12, oct 2010.
14. M. Ostrowski, T. Schaub. ASP modulo CSP: The clingcon system. *Theory and Practice of Logic Programming*, 12(4-5):485–503, 2012.
15. S. Prigent, G. Collet, S. Dittami, L. Delage, F. Ethis de Corny, O. Dameron, D. Eveillard, S. Thiele, J. Cambefort, C. Boyen, A. Siegel, T. Tonon. The genome-scale metabolic network of ectocarpus siliculosus (ectogem): a resource to study brown algal physiology and beyond. *The Plant Journal*, 80(2):367–381, 2014.
16. S. Prigent, C .Frioux, S. Dittami, S. Thiele, A. Larhlimi, G. Collet, F. Gutknecht, J. Got, D. Eveillard, J. Bourdon, F. Plewniak, T. Tonon, A. Siegel. Meneco, a Topology-Based Gap-Filling Tool Applicable to Degraded Genome-Wide Metabolic Networks. *PLOS Computational Biology*, 13(1):e1005276, jan 2017.
17. J. Reed, T. Vo, C. Schilling, B. Palsson. An expanded genome-scale model of Escherichia coli K-12 (iJR904 GSM/GPR). *Genome Biology*, 4(9):R54, 2003.
18. V. Satish Kumar, M. Dasika, C. Maranas. Optimization based automated curation of metabolic reconstructions. *BMC Bioinformatics*, 8(1):212, 2007.
19. T. Schaub, S. Thiele. Metabolic network expansion with ASP. *Proceedings ICLP*, 312–326. Springer, 2009.
20. P. Simons, I. Niemelä, T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1-2):181–234, 2002.
21. I. Thiele, N. Vlassis, R. Fleming. fastGapFill: efficient gap filling in metabolic networks. *Bioinformatics*, 30(17):2529–2531, sep 2014.
22. E. Vitkin, T. Shlomi. MIRAGE: a functional genomics-based approach for metabolic network model reconstruction and its application to cyanobacteria networks. *Genome Biology*, 13(11):R111, 2012.