Hybrid Metabolic Network Completion

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

Hybrid Metabolic Network Completion

Clémence Frioux
Univ Rennes, Inria, CNRS, IRISA F-35000 Rennes, France

Torsten Schaub
Inria, Rennes, France and Universität Potsdam, Germany

Sebastian Schellhorn
Universität Potsdam, Germany

Anne Siegel
Univ Rennes, Inria, CNRS, IRISA F-35000 Rennes, France

Philipp Wanko
Universität Potsdam, Germany

submitted xxxx; revised xxxx; accepted xxxx

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. *Under consideration in Theory and Practice of Logic Programming (TPLP).*

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

This is an extended version of a paper presented at LPNMR-17, invited as a rapid publication in TPLP. The authors acknowledge the assistance of the conference chairs Tomi Janhunen and Marco Balduccini.

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, also described as non-model organisms. However, the resulting metabolic networks are usually 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 (Schaub and Thiele, 2009), we introduced a logical approach to *metabolic network completion* by drawing on the work in (Handorf et al., 2005). We formulated the problem as a qualitative combinatorial (optimization) problem and solved it with Answer Set Programming (ASP (Baral, 2003)). 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 compounds in the resulting network is called the *scope* of the seeds and represents all compounds 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* compounds are in the scope of some given seeds. For instance, in the follow-up work in (Collet et al., 2013; Prigent et al., 2014), 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 Metacyc (Caspi et al., 2016).

We evidenced in (Prigent et al., 2017) that our ASP-based method partly restores the bio-synthetic capabilities of a large proportion of moderately degraded networks: it fails to restore the ones of both some moderately degraded and most 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 at the steady state of the system. 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*[1] 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 some 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[2] (Maranas and Zomorrodi, 2016)), the state-of-the-art 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* (Gebser et al., 2016). More precisely, we use an extension of *clingo* with linear constraints over reals, as dealt with in Linear Programming (LP (Dantzig, 1963)). This

[1] See also `https://en.wikipedia.org/wiki/Stoichiometry`.
[2] See also `https://en.wikipedia.org/wiki/Flux_balance_analysis`.

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

Metabolism is the sum of all chemical reactions occurring within an organism. As the products of a reaction may be reused as reactants, reactions can be chained to complex chemical pathways. Such complex pathways are described by a metabolic network.

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 *compounds* (also called 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. Metabolites and reactions nodes can both have multiple ingoing and outgoing edges. 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 *edge labeling* $s : E \to \mathbb{R}$ gives the stoichiometric coefficients of a reaction's reactants and products, respectively, i.e., their relative quantities involved in the reaction. 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 compounds with M and R (or M', R' etc), respectively.

We distinguish a set $S \subseteq M$ of compounds as initiation *seeds*, that is, compounds initially present due to experimental evidence. Another set of compounds is assumed to be activated by default. These *boundary compounds* 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 follow-up concepts like reachability and activity in network completion are independent of this assumption.

For illustration, consider the metabolic network in Fig. 1. The network consists of 9 reactions, r_{s_1}, r_{s_2}, r_e and r_0 to r_5, and 8 compounds, $A, \ldots, F$, S_1, S_2 and S_3. Here, $S = \{S_1, S_2, S_3\}$, S_1 and S_2 being the two boundary compounds of the network. Dashed rectangle describes the boundary of the system, outside of which is the environment of the organism. Consider reaction $r_4 : E \to 2C$ transforming one unit of E into two units of C (stoichiometric coefficients of 1 are omitted in the graphical representation; cf. Fig. 1). 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 (reaction r_5 in Fig. 1) is activated from seed compounds in S by possibly extending the metabolic network with reactions from a reference network (cf. shaded part in Fig. 2).

Formally, given a metabolic network $G = (R \cup M, E, s)$, a set $S \subseteq M$ of seed compounds

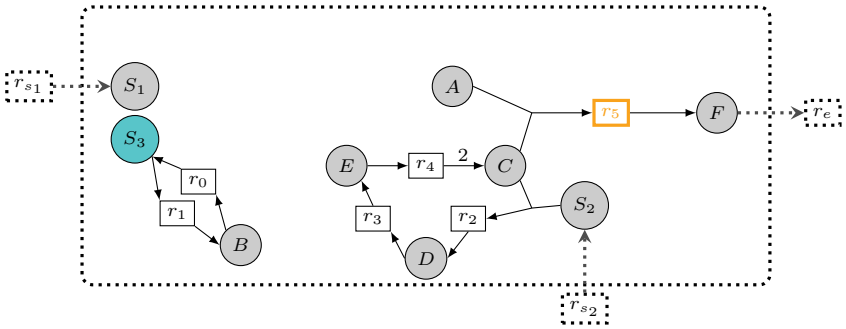

Fig. 1: Example of a metabolic network. Compounds and reactions are depicted by circles and rectangles respectively. Dashed reactions are reactions involving the boundary between the organism's metabolism and its environment. r_5 is the target reaction. S_1 and S_2 are boundary (and initiation) seeds. S_3 is assumed to be an initiation seed. Numbers on arrows describe the stoichiometry of reaction (default value is 1).

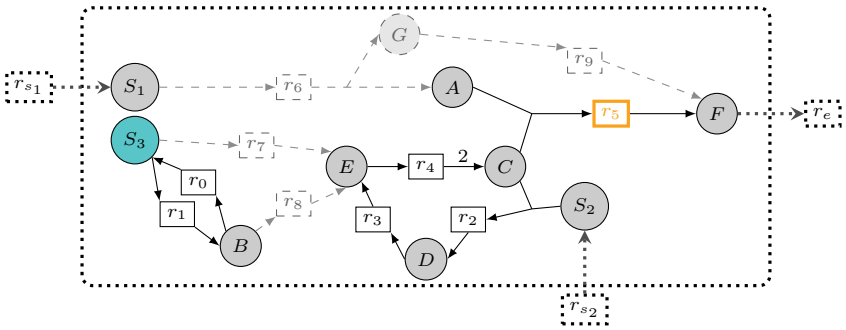

Fig. 2: Metabolic network completion problem. The purpose of its solving is to select the minimal number of reactions from a database (dashed shaded reactions) such that activation of target reaction r_5 is restored from boundary and/or initiation seeds. There are three formalisms for activation of target reaction: stoichiometric, topological and hybrid.

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 $R_T \subseteq \mathit{active}_{G''}(S)$ where[3]

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

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

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

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

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 different biological paradigms to be captured. Accordingly, different formulations of metabolic network completion can be characterized: the stoichiometric,

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

the relaxed stoichiometric, the topological, and the hybrid one. We elaborate upon their formal characterizations in the following sections.

2.1 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 stoichiometries are governed by the *law of mass conservation* under a steady state assumption; in other words, the mass of the system remains constant over the reaction. The input and output fluxes 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:

$$r_T \in \mathit{active}^s_G(S) \;\text{ iff }\; v_{r_T} > 0 \text{ and (5) and (6) hold for } M \text{ and } R. \tag{7}$$

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 compounds in $S_b\,(G)$ for which (6) is always satisfied and thus initiates the fluxes. Since boundary compounds are produced by at least one reaction without prerequisite, an arbitrary amount might be produced. Therefore, the incoming flux value always balances the sum of the flux values associated to outgoing edges. Intuitively, boundary compounds are nutrients that are expected to be available in the system for the consumption by the metabolic network, thus initiating the reactions within. In our draft network G, consisting of all non-dashed nodes and edges depicted in Fig. 2 (viz. reactions r_{s_1}, r_{s_2}, r_e and r_0 to r_5 and compounds $A, \ldots, F$, S_1, S_2, and S_3 and r_5 the single target reaction) and the reference network G', consisting of the shaded part of Fig 2, (viz. reactions r_6 to r_9 and metabolite G) a strict stoichiometry-based completion aims to obtain a solution with $r_5 \in \mathit{active}^s_{G''}(\{S_1, S_2, S_3\})$ where v_{r_5} is maximal. This can be achieved by adding the completion $R''_1 = \{r_6, r_9\}$ (Fig. 3). The cycle made of compounds E, C, D and the boundary seed S_2 is already balanced and notably self-activated. Indeed, initiation of D and E producibility requires the producibility of C (in addition to the presence of the boundary seed S_2) that itself depends on D and E. Yet, according the flux conditions, that models steady state conditions, the cycle is 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 because 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. This could be the case provided there exist another way to enable the production of one or several components of the cycle (here an activable

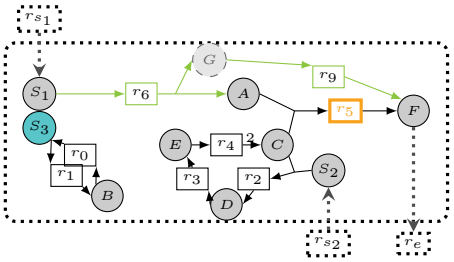

Fig. 3: Solution to metabolic network completion under stoichiometric activation hypothesis in order to satisfy Equations (5), (6) and (7). Within this network, there exists at least one flux distribution which activates r_5.

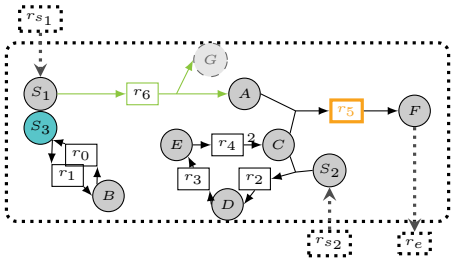

Fig. 4: Solution to metabolic network completion under relaxed stoichiometric activation hypothesis in order to satisfy Equations (5), (8) and (9). Notice that within this completed network, there exist no flux distribution allowing the reaction r_5 to be activated.

reaction producing E for instance) (Prigent et al., 2017). The instance of Equation (6) controlling the reaction rates related to metabolite C is $2 \cdot v_{r_4} - v_{r_2} - v_{r_5} = 0$.

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. Nonetheless, this problem turns out to be hard to solve in practice and existing approaches scale poorly to real-life applications (cf. (Orth and Palsson, 2010)).

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

$$\sum_{r \in R} s(r,m) \cdot v_r + \sum_{r \in R} -s(m,r) \cdot v_r \geq 0 \qquad \text{for } m \in M. \tag{8}$$

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

$$r_T \in \mathit{active}^r_G(S) \;\text{ iff }\; v_{r_T} > 0 \text{ and (5) and (8) hold for } M \text{ and } R. \tag{9}$$

The resulting problem can now be efficiently solved with Linear Programming (Satish Kumar et al., 2007). Existing systems addressing strict stoichiometric network completion either cannot guarantee optimal solutions (Latendresse, 2014) or do not support a focus on specific target reactions (Thiele et al., 2014). Other approaches either partially relax the problem (Vitkin and Shlomi, 2012) or solve the relaxed problem based on Equation (8), like the popular system *gapfill* (Satish Kumar et al., 2007). Applied to the network of Fig. 2, the minimal completion under the relaxed stoichiometric activation is $R''_1 = \{r_6\}$ (Fig. 4) but does not carry flux because of the accumulation of metabolite G, allowed by Equation (8). Note however that for strict steady-state modeling an *a posteriori* verification of solutions is needed to warrant the exact mass-balance equation (6).

2.2 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 $\mathit{rcts}(r)$ are reachable from S.

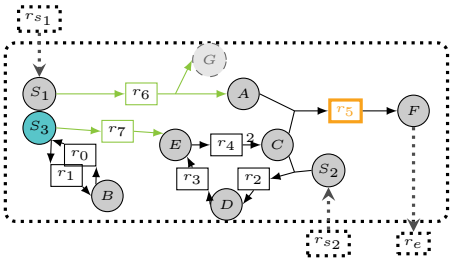

Fig. 5: First solution to metabolic network completion under topological activation hypothesis satisfying Equation (10). The production of C cannot be explained by a self-activated cycle and requires an external source of compounds via S_3 and reaction r_7.

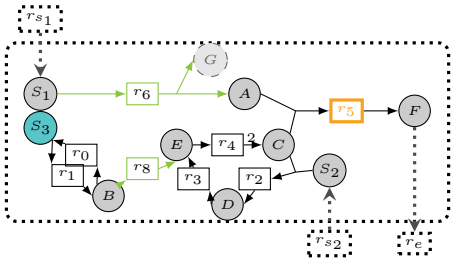

Fig. 6: Second solution to metabolic network completion under topological activation hypothesis satisfying Equation (10).

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 compounds reachable from S. In this setting, *topological activation* of reactions from a set of seeds S is defined as follows:

$$r_T \in active^t_G(S) \;\text{ iff }\; rcts(r_T) \subseteq \Sigma_G(S). \tag{10}$$

Note that this semantics avoids self-activated cycles by imposing an external entry sufficient to initiate all cycles (S_3 is not enough to activate the cycle as it does not activate one of its reaction on its own). The resulting network completion problem can be expressed as a combinatorial optimization problem and effectively solved with ASP (Schaub and Thiele, 2009).

For illustration, consider again the draft and reference networks G and G' in Fig. 1 and Fig. 2. We get $\Sigma_G(\{S_1, S_2, S_3\}) = \{S_1, S_2, S_3, B\}$, indicating that target reaction r_5 is not activated from the seeds with the draft network because A and C, its reactants, are not reachable. This changes once the network is completed. Valid minimal completions are $R''_2 = \{r_6, r_7\}$ (Fig. 5) and $R''_3 = \{r_6, r_8\}$ (Fig. 6) 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.

2.3 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:

$$r_T \in active^h_G(S) \;\text{ iff }\; r_T \in active^s_G(S) \text{ and } r_T \in active^t_G(S). \tag{11}$$

Applying this to our example in Fig. 2, we get the (minimal) hybrid solutions $R''_4 =$

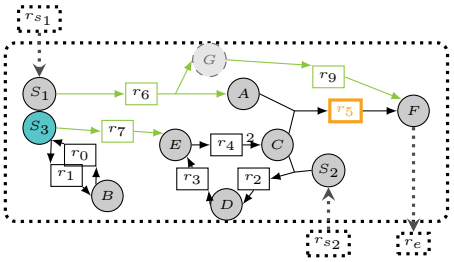

Fig. 7: First solution to metabolic network completion under hybrid activation hypothesis satisfying Equation (11) (that is Equations (5), (6), (7) and (10)).

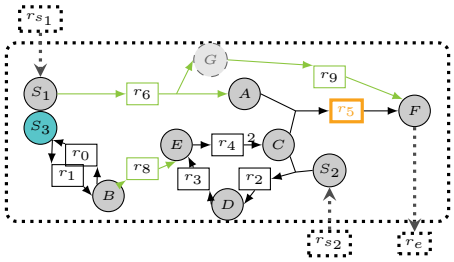

Fig. 8: Second solution to metabolic network completion under hybrid activation hypothesis satisfying Equation (11) (that is Equations (5), (6), (7) and (10)).

$\{r_6, r_7, r_9\}$ (Fig. 7) and $R''_5 = \{r_6, r_8, r_9\}$ (Fig. 8). Both (topologically) initiate paths of reactions from the seeds to the target, ie. $r_5 \in \mathit{active}^t_{G''_i}(\{S_1, S_2, S_3\})$ since $\{A, C\} \subseteq \Sigma_{G''_i}(\{S_1, S_2, S_3\})$ 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 $r_5 \in \mathit{active}^s_{G''_i}(\{S_1, S_2, S_3\})$.

2.4 Union of Metabolic Network Completions

As depicted in the toy examples for the topological (Fig. 5 and Fig. 6) and hybrid (Fig. 7 and Fig. 8) activation, several minimal solutions to one metabolic network completion problem may exist. There might be dozens of minimal completions, depending on the degradation of the original draft network, hence leading to difficulties for biologists and bioinformaticians to discriminate the individual results. One solution to facilitate this curation task is to provide, in addition to the enumeration of solutions, their union. This has been done previously for the topological completion (Prigent et al., 2017).

Notably, the concept of "union of solutions" is particularly relevant from the biological perspective since it provides in a single view all possible reactions that could be inserted in a solution to the network completion problem. Additionally, verifying the union according to the desired (stoichiometric and hybrid) activation semantics, offers a way to analyze the quality of approximation methods (topological and relaxed-stoichiometric ones). If individual solutions contradict a definition of activation that the union satisfies, it suggests that the family of reactions contained in the union, although possibly non-minimal, may be of interest. Thus providing merit to the approximation method and their results.

Importantly, we notice that the operation of performing the union of solutions is stable with the concept of activation, although it can contradict the minimality of the size of completion. Indeed, the union of solutions to the topological network completion problem is itself a (non-minimal) solution to the topological completion problem. Similarly, the union of minimal stoichiometric solutions always displays the stoichiometric activation of the target reaction(s). In fact, adding an arbitrary set of reactions to a metabolic network still maintains stoichiometric activation, since flux distribution for the newly

added reactions may be set to zero. Consequently, the union of minimal hybrid solutions always displays the hybrid activation in the target reaction(s).

The following theorems (Theorems 1, 2 and 3) are a formalization of the stability of the union of solutions with respect to the three concepts of activation.

The union $G = G_1 \cup G_2$ of two metabolic networks $G_1 = (R_1 \cup M_1, E_1, s_1)$ and $G_2 = (R_2 \cup M_2, E_2, s_2)$ is defined by

$$G = (R \cup M, E, s), \tag{12}$$
$$R = R_1 \cup R_2, \tag{13}$$
$$M = M_1 \cup M_2, \tag{14}$$
$$E = E_1 \cup E_2, \tag{15}$$
$$s = s_1 \cup s_2. \tag{16}$$

Theorem 1. *Let G_1 and G_2 be metabolic networks. If $R_T \subseteq active^t_{G_1}(S)$, then $R_T \subseteq active^t_{G_1 \cup G_2}(S)$.*

Proof. The proof is given by monotonicity of the union and the monotonicity of the closure. Thus it can never be case that having more reactions disables reachability. More formal, $R_T \subseteq active^t_{G_1}(S)$ holds iff $rcts(r_T) \subseteq \Sigma_{G_1}(S)$. Furthermore, we have $\Sigma_{G_1}(S) \subseteq \Sigma_{G_1 \cup G_2}(S)$ by the definition of the closure. This implies $rcts(r_T) \subseteq \Sigma_{G_1 \cup G_2}(S)$. Finally, we have $R_T \subseteq active^t_{G_1 \cup G_2}(S)$. □

Theorem 2. *Let G_1 and G_2 be metabolic networks. If $R_T \subseteq active^s_{G_1}(S)$, then $R_T \subseteq active^s_{G_1 \cup G_2}(S)$.*

Proof. First, we define following bijective functions

$$\begin{aligned}
f :& R_1 \to \{1, \ldots, l\} \subseteq \mathbb{N}, \\
& r \mapsto f(r) = i \\
g :& M_1 \to \{1, \ldots, k\} \subseteq \mathbb{N}, \\
& m \mapsto g(m) = j \\
f' :& R_1 \cup R_2 \to \{1, \ldots, l'\} \subseteq \mathbb{N}, \\
& r \mapsto f'(r) = \begin{cases} f(r) & , \text{ if } f(r) \text{ is defined} \\ i & , \text{ otherwise} \end{cases} \\
g' :& M_1 \cup M_2 \to \{1, \ldots, k'\} \subseteq \mathbb{N} \\
& m \mapsto g'(m) = \begin{cases} g(m) & , \text{ if } g(m) \text{ is defined} \\ j & , \text{ otherwise} \end{cases}
\end{aligned}$$

for $k = |M_1|$, $l = |R_1|$, $k' = |M_1 \cup M_2|$ and $l' = |R_1 \cup R_2|$ regarding G_1 and $G_1 \cup G_2$, respectively. Now, we rewrite the system of (6) regarding G_1 as a matrix equation $Av = 0$ of form

$$\begin{pmatrix} a_{11} & \ldots & a_{1l} \\ \vdots & \ddots & \vdots \\ a_{k1} & \ldots & a_{kl} \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_l \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

where A is a $k \times l$ matrix with coefficients

$$a_{g(m)f(r)} = \begin{cases} s_1(r,m) & ,(r,m) \in E_1 \\ -s_1(m,r) & ,(m,r) \in E_1 \\ 0 & , \text{ otherwise} \end{cases}$$

and v consists of variables $v_{f(r)}$ for $r \in R_1$. By $L = \{v \mid Av = 0\}$ we denote the set of solutions induced by $Av = 0$.

Furthermore, we represent the system of linear equations of (6) regarding $G_1 \cup G_2$ as a matrix equation $A'v' = 0$ of form

$$\begin{pmatrix} a_{11} & \dots & a_{1l} & a_{1l+1} & \dots & a_{1l'} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{k1} & \dots & a_{kl} & a_{kl+1} & \dots & a_{kl'} \\ 0 & \dots & 0 & a_{k+1l+1} & \dots & a_{k+1l'} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & a_{k'l+1} & \dots & a_{k'l'} \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_l \\ v_{l+1} \\ \vdots \\ v_{l'} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

where A' is a $k' \times l'$ matrix with coefficients

$$a_{g'(m)f'(r)} = \begin{cases} s(r,m) & ,(r,m) \in E_1 \cup E_2 \\ -s(m,r) & ,(m,r) \in E_1 \cup E_2 \\ 0 & , \text{ otherwise} \end{cases}$$

where $s = s_1 \cup s_2$ and v' consists of variables $v_{f'(r)}$ of (6) for $r \in R_1 \cup R_2$. Note that A' can always be written in this form, since switching columns and rows will not change solutions. By $L' = \{v' \mid A'v' = 0\}$ we denote the set of solutions induced by $A'v' = 0$.

Since $A'v' = 0$ is homogeneous, $L \subseteq L'$ holds by extending L with zeros for $v_{f'(r)}$ with $r \in R_2 \setminus R_1$. Thus $\{v \mid v \in L, \forall r_T \in R_T, v_{f(r_T)} > 0\} \subseteq \{v \mid v \in L', \forall r_T \in R_T, v_{f'(r_T)} > 0\}$ by extending the first set with zeros for $v_{f'(r)}$ with $r \in R_2 \setminus R_1$. From $R_T \subseteq active^s_{G_1}(S)$, we know that the homogeneous system of linear equations from (6) regarding G_1 is non-trivial satisfiable, which finally implies that $R_T \subseteq active^s_{G_1 \cup G_2}(S)$. □

Theorem 3. *Let G_1 and G_2 be metabolic networks. If $R_T \subseteq active^h_{G_1}(S)$, then $R_T \subseteq active^h_{G_1 \cup G_2}(S)$.*

Proof. Follows directly by the definition of hybrid activation together with Theorem 1 and Theorem 2. More formal, $R_T \subseteq active^h_{G_1}(S)$ holds iff $R_T \subseteq active^t_{G_1}(S)$ and $R_T \subseteq active^s_{G_1}(S)$. From Theorem 1 and $R_T \subseteq active^t_{G_1}(S)$ follows $R_T \subseteq active^t_{G_1 \cup G_2}(S)$. Analogously, from Theorem 2 and $R_T \subseteq active^s_{G_1}(S)$ follows $R_T \subseteq active^s_{G_1 \cup G_2}(S)$. Finally, this implies $R_T \subseteq active^h_{G_1 \cup G_2}(S)$. □

In particular, studying the union in case of topological modeling can pinpoint interesting cases. Individual solutions satisfying the topological activation can additionally satisfy the stoichiometric and thus the hybrid activation semantics. A union including such a solution will also adhere to the hybrid standard. In some cases, the union of solutions will display the stoichiometric activation whereas the individual solutions only satisfy the topological activation. Fig. 9 to Fig. 11 display an example of topological metabolic network completions that do not satisfy stoichiometric (and hybrid) activation whereas

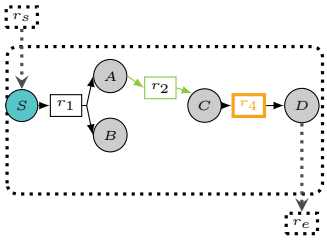

Fig. 9: Topological completion $R_1 = \{r_2\}$ satisfies $r_4 \in \mathit{active}^t_{G_1}(\{S\})$, but carries no flux, due to accumulation of compound B that contradicts Eq. 6.

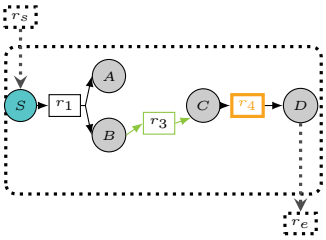

Fig. 10: Topological completion $R_2 = \{r_3\}$ satisfies $r_4 \in \mathit{active}^t_{G_2}(\{S\})$ and carries no flux as well, due to accumulation of compound A that contradicts Eq. 6.

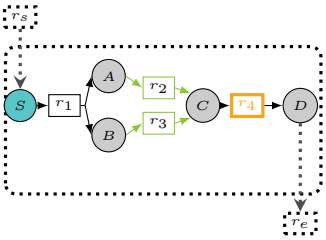

Fig. 11: Completion with the union $R_1 \cup R_2 = \{r_2, r_3\}$. $G = G_1 \cup G_2$ satisfies $r_4 \in \mathit{active}^h_G(\{S\})$ and thus is flux-balanced.

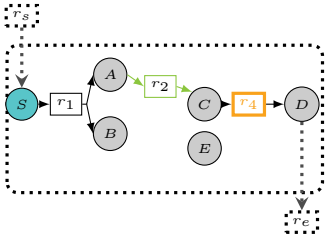

Fig. 12: Topological completion $R_1 = \{r_2\}$ satisfies $r_4 \in \mathit{active}^t_{G_1}(\{S\})$, but carries no flux, due to accumulation of compound B that contradicts Eq. 6.

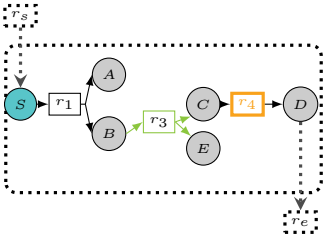

Fig. 13: Topological completion $R_1 = \{r_3\}$ satisfies $r_4 \in \mathit{active}^t_{G_2}(\{S\})$, but carries no flux, due to accumulation of compounds A and E that contradicts Eq. 6.

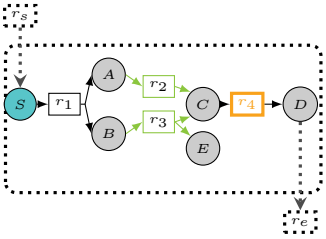

Fig. 14: Completion with the union $R_1 \cup R_2 = \{r_2, r_3\}$. $G = G_1 \cup G_2$ satisfies $r_4 \in \mathit{active}^t_G(\{S\})$, but contradicts minimality and carries no flux $r_4 \notin \mathit{active}^s_G(\{S\})$, due to accumulation of compound E that contradicts Eq. 6.

their union does. Fig. 12 to Fig. 14 provide an example of minimal topological completions that do not satisfy stoichiometric (and hybrid) activation and for which the union does not satisfy it either.

Both observations induce that in general we cannot derive anything about activation of reactions in a graph resulting from the union of two or more graphs. And similarly, we cannot infer about the activation of reactions in subgraphs arbitrarily derived from a graph in which these reactions are activated.

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 (Gebser et al., 2016).

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[4] '$\texttt{\&sum\{w}_1\texttt{*x}_1\texttt{;...;w}_l\texttt{*x}_l\texttt{\} <= k}$' that stands for the linear constraint $w_1 \cdot x_1 + \cdots + w_l \cdot x_l \leq k$. All $\texttt{w}_i$ and $\texttt{k}$ are finite sequences of digits with at most one dot[5] and represent real-valued coefficients w_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 (Gelfond and Lifschitz, 1991). Such a stable model X is an *LC-stable model* of a logic program P,[6] 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. For a more detailed introduction of ASP extended with linear constraints, illustrated with more complex examples, we refer the interested reader to (Janhunen et al., 2017).

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* (Simons et al., 2002). 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 `a` and $\texttt{b}_i$ are possibly default-negated (regular) literals and each $\texttt{d}_j$ is a conditional literal; `s` and `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 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 $\texttt{\#minimize\{}w_1\texttt{:}c_1\texttt{;...;}w_n\texttt{:}c_n\texttt{\}}$.

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

'$\texttt{\&sum\{w}_1\texttt{*x}_1\texttt{:c}_1\texttt{;...;w}_l\texttt{*x}_l\texttt{:c}_n\texttt{\} <= k}$'

where each $\texttt{c}_i$ is a condition. Moreover, the theory language for linear constraints offers a domain declaration for real variables, '`&dom{lb..ub} = x`' expressing that all

[4] In *clingo*, theory atoms are preceded by '`&`'.
[5] In the input language of *clingo*, such sequences must be quoted to avoid clashes.
[6] This corresponds to the definition of T-stable models using a *strict* interpretation of theory atoms (Gebser et al., 2016), and letting T be the theory of linear constraints over reals.

values of `x` must lie between `lb` and `ub`. And finally the maximization (or minimization) of an objective function can be expressed with `&maximize{`w_1`*x`$_1$`:c`$_1$`;...;`w_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:

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

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 compounds, 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:

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

where $T = \{m \in \mathit{rcts}(r) \mid r \in R_T\}$ is the set of target compounds 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.

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 (Schaub and Thiele, 2009).) 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 compounds 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 compounds 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 compounds. We draw on this in

```
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

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 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 $R_T \subseteq \mathit{active}^s_{G''}(S)$.

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* (Gebser et al., 2016) for implementing the combination of ASP and LP. We use *clingo* 5.2.0 incorporating as LP solvers either *cplex* 12.7.0.0 or *lpsolve* 5.5.2.5 via their respective Python

CORE- \ PROP-	0	25	50	75	100
0	383.20(130)	388.51(134)	384.46(133)	388.45(137)	398.21(134)
25	385.05(132)	385.95(133)	391.84(131)	382.29(137)	401.73(134)
50	383.95(131)	377.51(123)	385.46(132)	391.05(137)	399.88(141)
75	358.77(129)	360.54(127)	356.89(131)	390.69(137)	399.36(134)
100	376.03(133)	370.75(132)	375.77(133)	389.77(139)	401.18(139)

Table 1: Comparison of propagation and core minimization heuristics for BB.

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

$$\begin{aligned} R'' = \{&\texttt{completion}(r_6), \texttt{completion}(r_8), \texttt{completion}(r_9)\} \\ \text{and } \{&r_{s_1} = 49999.5, r_9 = 49999.5, r_3 = 49999.5, r_2 = 49999.5, \\ &r_e = 99999.0, r_6 = 49999.5, r_5 = 49999.5, r_4 = 49999.5\}. \end{aligned}$$

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 impact of different system configurations (ii) the quality of *fluto*'s approach to metabolic network completion, and (iii) compare the quality of *fluto*'s solutions with other approaches. To have a realistic setting, we use degradations of a functioning metabolic network of *Escherichia coli* (Reed et al., 2003) comprising 1075 reactions. The network was randomly degraded by 10, 20, 30 and 40 percent, creating 10 networks for each degradation by removing reactions until the target reactions were inactive according to *Flux Variability Analysis* (Becker et al., 2007). 90 target reactions with varied reactants were randomly chosen for each network, yielding 3600 problem instances in total (Prigent et al., 2017). 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 (Ansótegui et al., 2013), which provides at least some suboptimal solutions.

CORE- \ PROP-	0	25	50	75	100
0	297.38(102)	296.39(102)	296.48(103)	299.14(105)	475.12(200)
25	297.29(101)	293.69(100)	297.09(102)	293.43(101)	478.39(202)
50	292.65(102)	296.43(102)	294.4(103)	295.48(102)	477.67(200)
75	331.72(127)	336.34(129)	331.17(127)	294.17(103)	476.16(202)
100	308.88(108)	309.47(107)	324.9(122)	489.97(214)	476.17(201)

Table 2: Comparison of propagation and core minimization heuristics for USC.

5.1 System configurations

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 concentrate on distinguished features revealing an impact in our experiments.

The first focus are two system options controlling the hybrid solving nature of *fluto*. First, PROP-n controls the frequency of LP propagation: the consistency of linear constraints is only checked if n% of atoms are decided. Second, the *fluto* option CORE-n invokes the irreducible inconsistent set algorithm (Ostrowski and Schaub, 2012) whenever n% of atoms are decided. This algorithm extracts a minimal set of conflicting linear constraints for a given conflict. Note that the second parameter depends on the first one, since conflict analysis may only be invoked if the LP solver found an inconsistency.

The DEFAULT is to use CORE-100, PROP-0, and use LP solver *cplex*[7]. This allows us to detect conflicts among the linear constraints as soon as possible and only perform expensive conflict analysis on the full assignment.

To get an overview, we conducted a preliminary experiment using BB and USC with *fluto*'s default configuration on the 10, 20, and 30 percent degraded networks, 2700 instances in total, limiting execution time to 20 minutes. For our performance experiments, we selected at random three networks with at least one instance for which BB and USC could find the optimum in 100 to 600 seconds. With the resulting 270 medium to hard instances, we examined the cross product of values $n \in \{0, 25, 50, 75, 100\}$ for CORE-n and PROP-n, respectively, limiting time to 600 seconds.

Table 1 and Table 2 display the results using BB and USC respectively. The columns increase the value for PROP-n and the rows for CORE-n in steps of 25, i.e., LP propagation becomes less frequent from left to right, and conflict minimization from top to bottom. The first value in each cell is the average runtime in seconds and the value in brackets shows the number of timeouts. The shade of the cells depends on the average runtime, i.e., the darker the cell, the less performant the combination of propagation and conflict minimization heuristics.

Table 1 shows that propagation and conflict minimization heuristics have an overall small impact on the performance of BB optimization. Since BB relies on iterating solutions and learns weaker constraints, only pertaining to the best known bound, while optimizing, the improvement step is less constraint compared to USC. Due to this, conflicts are more likely to appear later on in the optimization process allowing for less impact of frequent LP propagation and conflict minimization Nevertheless, we see a slight perfor-

[7] We do not present results of *lpsolve* since it produced inferior results.

	FR		JP		TW		TR		CR		HD	
	T	TO	T	TO	T	TO	T	TO	T	TO	T	TO
BB	400.41	154	389.68	147	**360.54**	127	409.33	141	362.74	**120**	434.54	160
USC	227.38	78	293.96	100	316.54	107	293.54	102	**221.84**	**74**	297.32	104

Table 3: Comparison of *clingo*'s portfolio configurations for BB and USC.

mance improvement of propagating and conflict minimizing for every partial assignment (PROP-0, CORE-0) compared to only on full assignments (PROP-100, CORE-100). To prove the optimum, the solver is still required to cover the whole search space. For this purpose, early pruning and conflict minimization may be effective. Furthermore, we see the best average runtime in the area PROP-0-50 at CORE-75. That indicates a good tradeoff between the better quality conflicts which prune the search effectively and the overhead of the costly conflict minimization. There is no clear best configuration, but PROP-25 and CORE-75 shows the best tradeoff between average runtime and number of timeouts.

USC on the other hand (Table 2), clearly benefits from early propagation and conflict minimization. The area PROP-0-75 and CORE-0-50 has the lowest average runtime and number of timeouts, best among them PROP-25 and CORE-25 with the lowest timeouts and average runtime that is not significantly different from the best value. USC aims at quickly identifying unsatisfiable partial assignments and learning structural constraints building upon each other, which is enhanced by frequent conflict detection and minimization. Disabling LP propagation on partial assignments with USC leads to the overall worst performance and we also see deterioration with CORE-75 and CORE-100 in the interval PROP-0-50. Overall, USC is more effective than BB for the instances and we see a benefit in early LP propagation and conflict minimization as well as in fine-tuning the heuristics at which point both are applied.

Now, we focus on the portfolio configurations of *clingo*. Those configurations were crafted by experts to enhance the solving performance of problems with certain attributes. To examine their impact, we take the best result for BB (PROP-25 and CORE-75) and USC (PROP-25 and CORE-25), and employ the following *clingo* options:

FR Refers to *clingo*'s configuration *frumpy* that uses more conservative defaults.
JP Refers to *clingo*'s configuration *jumpy* that uses more aggressive defaults.
TW Refers to *clingo*'s configuration *tweety* that is geared toward typical ASP problems.
TR Refers to *clingo*'s configuration *trendy* that is geared toward industrial problems.
CR Refers to *clingo*'s configuration *crafty* that is geared towards crafted problems.
HD Refers to *clingo*'s configuration *handy* that is geared towards larger problems.

For more information on *clingo*'s configurations, see (Gebser et al., 2015).

Table 3 shows the average runtime in seconds (T) and number of timeouts (TO) for all six configurations using BB and USC on the same 270 instances. Even though CR has slightly higher average runtime for BB compared to TW, it is the overall best configuration. This configuration is geared towards problems with an inherent structure compared to randomly generated benchmarks which fits with the metabolic network completion problem at hand since the data is taken from an existing bacteria. Interestingly, BB performs worse under more specific configurations and favors moderate once like TW and CR. This might be due to the changing nature of improvement steps as the optimization process goes on from finding any random solutions to an unsatisfiability proof in the

	F(BB)		F(USC)		F(BB+USC)		F(BB+USC)
DEGRADATION	#SOLS	#OPTS	#SOLS	#OPTS	#SOLS	#OPTS	VERIFIED
10% (900)	**900**	**900**	892	892	900	900	900
20% (900)	**830**	669	793	**769**	867	814	867
30% (900)	**718**	88	461	**344**	780	382	780
all (2700)	**2448**	1657	2146	**2005**	2547	2096	2547

Table 4: Comparison of qualitative results.

	F(VBS)		VERIFIED
DEGRADATION	#SOLS	#OPTS	F(VBS)
10% (900)	900	900	900
20% (900)	896	855	896
30% (900)	848	575	848
40% (900)	681	68	681
all (3600)	3325	2398	3325

Table 5: Results using best system options.

end. USC on the other hand, benefits from a more structural heuristics in CR and more conservative defaults in FR which allow the solver to explore and collect conflicts instead of frequently restarting and forgetting.

5.2 Solution quality

Now, we examine the quality of the solutions provided by *fluto*. Table 4 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**. The default setting for *fluto* includes the default configurations for *clingo* and *cplex*. The data was obtained in our preliminary experiment using networks with 10, 20, and 30 percent degradation. 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 shows the quality of solutions provided by *fluto*. Each obtained best solution was checked with *cobrapy* 0.3.2 (Ebrahim et al., 2013), a renowned system implementing an FBA-based gold standard (for verification only). 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.

Finally, we change the system configuration and examine how *fluto* scales on harder instances. To this end, we use the best configurations from Section 5.1, PROP-25, CORE-75 and CR for BB, and PROP-25, CORE-25 and CR for USC, and rerun the experiment on all 3600 instances. The results are shown in (Table 5). F(VBS) denotes the virtual best results, meaning for each problem instance the best known solution among the two 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.

	fluto			*meneco*		
	min	average	max	min	average	max
solutions per instance	1	2.24	12	1	1.88	6
reactions per solution	1	6.66	9	1	6.24	9
verified solutions			100%			73.39%
instances with only verified solutions			100%			72.94%
instances without verified solutions			0%			26.61%
instance with some verified solutions			0%			0.45%

Table 6: Comparison of *fluto* and *meneco* solutions for 10 percent degraded networks.

	fluto	*meneco*	*gapfill*
verified union	100%	73.39%	6.20%
verified union of verified solutions	100%	72.94%	NA
verified union of unverified solutions	0%	0.00%	NA
verified union of partially verified solutions	0%	0.45%	NA

Table 7: Comparison of *fluto*, *meneco* and *gapfill* unions for 10 percent degraded networks.

The results show that *fluto* is capable of finding correct completions for even highly degraded networks for most of the instances in reasonable time.

5.3 Comparison to other approaches

We compare the quality of *fluto* with *meneco* 1.4.3 (Prigent et al., 2017) and *gapfill*[8] (Satish Kumar et al., 2007). [9] Both *meneco* and *gapfill* are systems for metabolic network completion. While *meneco* pursues the topological approach, *gapfill* applies the relaxed stoichiometric variant using Inequation (8). We performed an enumeration of all minimal solutions to the completion problem under the topological (*meneco*), the relaxed stoichiometric (*gapfill*), and hybrid (*fluto*) activation semantics for the 10 percent degraded networks of the benchmark set (900 instances to be completed).

First, we compare the quality of individual solutions of *fluto* and *meneco*. [10] Results are displayed in Table 6. The first two rows give the minimum, average and maximum number of solutions per instance, and reactions per solution, respectively, for *fluto* and *meneco*. While *fluto* finds 19% more solutions on average and twice as many maximum solutions per instance compared to *meneco*, the numbers of reactions in minimal solutions of both tools are similar. The next four rows pertain to the solution quality as established by *cobrapy*. First, what percent solutions over all instances could be verified, second, what percent of instances had verified solutions exclusively, third, how many instances had no verified solutions at all, and finally, percent of instances where only a portion of solutions could be verified. All of *fluto*'s solutions could be verified, compared to the 72.04% of *meneco* across all solutions and 72.94% of instances that were correctly solved. Interestingly, *meneco* achieves hybrid activation in some but not all solutions for 0.45% (4) of the instances. *fluto* does not only improve upon the quality of *meneco*, but also

[8] Update of 2011-09-23 see http://www.maranasgroup.com/software.htm
[9] The results for *meneco* and *gapfill* are taken from previous work (Prigent et al., 2017), where they were run to completion with *no* time limit.
[10] There was no data available for the individual solutions of *gapfill*.

provides more solution per instances without increasing the number of relevant reactions significantly.

To empirically evaluate the properties established in Section 2.4, and be able to compare to *gapfill*, for which only the union of reactions was available, we examine the union of minimal solutions provided by all three systems and present the results in Table 7. The four rows show, first, for what percent of instances the union of solutions could be verified, second, how many instances had only verified solutions and their union was also verified, third, the percentage of instances where the union of solutions displayed activation of the target reactions even though all individual solutions did not provide that, and forth, instances where the solutions were partly verifiable and their union could also be verified. While again 100% of *fluto*'s solutions could be verified, only 73.3% and 6.2% are obtained for *meneco* and *gapfill*, respectively, for 10 percent degraded networks. As reflected by the results, the ignorance of *meneco* regarding stoichiometry leads to possibly unbalanced networks. Still, the union of solutions provided a useful set of reactions in almost three quarters of the instances, showing merit in the topological approximation of the metabolic network completion problem. On the other hand, the simplified view of *gapfill* in terms of stoichiometry misguides the search for possible completions and eventually leads to unbalanced networks even in the union. Moreover, *gapfill*'s ignorance of network topology results in self-activated cycles. By exploiting both topology and stoichiometry, *fluto* avoids such cycles while still satisfying the stoichiometric activation criteria. The results support the observations made in Section 2.4. For both *fluto* and *meneco* all instances, for which the complete solution set could be verified, the union is also verifiable, as well as all unions for instances where *meneco* established hybrid activation for a fraction of solutions.

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. (Gebser et al., 2015)).

Acknowledgments This work was partially funded by DFG grant SCHA 550/9 and 11 and benefited from the support of the French Government via the National Research Agency investment expenditure program IDEALG ANR-10-BTBR-04.

References

Ansótegui, C., Bonet, M., and Levy, J. 2013. SAT-based MaxSAT algorithms. *Artificial Intelligence 196*, 77–105.

Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press.

Becker, S., Feist, A., Mo, M., Hannum, G., Palsson, B., and Herrgard, M. 2007. Quantitative Prediction of Cellular Metabolism with Constraint-based Models: The COBRA Toolbox. *Nature Protocols 2,* 3, 727–738.

Caspi, R., Billington, R., Ferrer, L., Foerster, H., Fulcher, C. A., Keseler, I. M., Kothari, A., Krummenacker, M., Latendresse, M., Mueller, L. A., Ong, Q., Paley, S., Subhraveti, P., Weaver, D. S., and Karp, P. D. 2016. The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases. *Nucleic acids research 44,* D1 (jan), D471–80.

Collet, G., Eveillard, D., Gebser, M., Prigent, S., Schaub, T., Siegel, A., and Thiele, S. 2013. Extending the metabolic network of Ectocarpus siliculosus using answer set programming. In *Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'13)*, P. Cabalar and T. Son, Eds. Lecture Notes in Artificial Intelligence, vol. 8148. Springer-Verlag, 245–256.

Dantzig, G. 1963. *Linear Programming and Extensions*. Princeton University Press.

Ebrahim, A., Lerman, J., Palsson, B., and Hyduke, D. 2013. COBRApy: COnstraints-Based Reconstruction and Analysis for Python. *BMC Systems Biology 7*, 74.

Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., and Wanko, P. 2016. Theory solving made easy with clingo 5. In *Technical Communications of the Thirty-second International Conference on Logic Programming (ICLP'16)*, M. Carro and A. King, Eds. Vol. 52. Open Access Series in Informatics (OASIcs), 2:1–2:15.

Gebser, M., Kaminski, R., Kaufmann, B., Romero, J., and Schaub, T. 2015. Progress in clasp series 3. In *Proceedings of the Thirteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'15)*, F. Calimeri, G. Ianni, and M. Truszczyński, Eds. Lecture Notes in Artificial Intelligence, vol. 9345. Springer-Verlag, 368–383.

Gelfond, M. and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing 9*, 365–385.

Handorf, T., Ebenhöh, O., and Heinrich, R. 2005. Expanding metabolic networks: Scopes of compounds, robustness, and evolution. *Journal of Molecular Evolution 61,* 4, 498–512.

Janhunen, T., Kaminski, R., Ostrowski, M., Schaub, T., Schellhorn, S., and Wanko, P. 2017. Clingo goes linear constraints over reals and integers. *Theory and Practice of Logic Programming 17,* 5-6, 872–888.

Latendresse, M. 2014. Efficiently gap-filling reaction networks. *BMC bioinformatics 15,* 1, 225.

Maranas, C. and Zomorrodi, A. 2016. *Optimization methods in metabolic networks.* John Wiley & sons.

Orth, J. and Palsson, B. 2010. Systematizing the generation of missing metabolic knowledge. *Biotechnology and bioengineering 107,* 3 (oct), 403–12.

Ostrowski, M. and Schaub, T. 2012. ASP modulo CSP: The clingcon system. *Theory and Practice of Logic Programming 12,* 4-5, 485–503.

Prigent, S., Collet, G., Dittami, S., Delage, L., Ethis de Corny, F., Dameron, O., Eveillard, D., Thiele, S., Cambefort, J., Boyen, C., Siegel, A., and Tonon, T. 2014. 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.

Prigent, S., .Frioux, C., Dittami, S., Thiele, S., Larhlimi, A., Collet, G., Gutknecht, F., Got, J., Eveillard, D., Bourdon, J., Plewniak, F., Tonon, T., and Siegel, A. 2017. Meneco, a Topology-Based Gap-Filling Tool Applicable to Degraded Genome-Wide Metabolic Networks. *PLOS Computational Biology 13,* 1 (jan), e1005276.

Reed, J., Vo, T., Schilling, C., and Palsson, B. 2003. An expanded genome-scale model of Escherichia coli K-12 (iJR904 GSM/GPR). *Genome Biology 4,* 9, R54.

Satish Kumar, V., Dasika, M., and Maranas, C. 2007. Optimization based automated curation of metabolic reconstructions. *BMC Bioinformatics 8,* 1, 212.

Schaub, T. and Thiele, S. 2009. Metabolic network expansion with ASP. In *Proceedings of the Twenty-fifth International Conference on Logic Programming (ICLP'09)*, P. Hill and D. Warren, Eds. Lecture Notes in Computer Science, vol. 5649. Springer-Verlag, 312–326.

Simons, P., Niemelä, I., and Soininen, T. 2002. Extending and implementing the stable model semantics. *Artificial Intelligence 138,* 1-2, 181–234.

Thiele, I., Vlassis, N., and Fleming, R. 2014. fastGapFill: efficient gap filling in metabolic networks. *Bioinformatics 30,* 17 (sep), 2529–2531.

Vitkin, E. and Shlomi, T. 2012. MIRAGE: a functional genomics-based approach for metabolic network model reconstruction and its application to cyanobacteria networks. *Genome Biology 13,* 11, R111.

Appendix A Factual representation of example metabolic network

The factual representation of the metabolic network in Fig. 2 is given in Listing 2.

```
1  metabolite("S1",s). metabolite("S2",s). metabolite("S3",s).
2  metabolite("a",t).  metabolite("b",d).  metabolite("c",t).
3  metabolite("d",d).  metabolite("e",d).  metabolite("f",d).
4  metabolite("g",r).

6  reaction("R_importS1",s).      reaction("R_importS2",s).
7  reversible("R_importS1").      reversible("R_importS2").
8  prd("S1","1","R_importS1",s). prd("S2","1","R_importS2",s).
9  reaction("R_exportF",d).       rct("f","1","R_exportF",d).

11 reaction("R0",d).       reaction("R1",d).
12 rct("b","1","R0",d).    rct("S3","1","R1",d).
13 prd("S3","1","R0",d).   prd("b","1","R1",d).

15 reaction("R2",d).       reaction("R3",d).
16 rct("c","1","R2",d).    rct("d","1","R3",d).
17 rct("S2","1","R2",d).   prd("e","1","R3",d).
18 prd("d","1","R2",d).

20 reaction("R4",d).       reaction("R9",r).
21 rct("e","1","R4",d).    rct("g","1","R9",r).
22 prd("c","2","R4",d).    prd("f","1","R9",r).

24 reaction("R5",t).       reaction("R6",r).
25 rct("a","1","R5",t).    rct("S1","1","R6",r).
26 rct("c","1","R5",t).    prd("a","1","R6",r).
27 prd("f","1","R5",t).    prd("g","1","R6",r).

29 reaction("R7",r).       reaction("R8",r).
30 rct("S3","1","R7",r).   rct("b","1","R8",r).
31 prd("e","1","R7",r).    prd("e","1","R8",r).

33 objective(R,T) :- reaction(R,T), T!=t.
34 objective(R,t) :- reaction(R,t).

36 bounds(R,"0","99999") :- reaction(R,_), not reversible(R).
37 bounds(R,"-99999","99999") :- reaction(R,_), reversible(R).
```

Listing 2: Example instance of metabolic network

Note that in lines 33 to 37 of Listing 2, the values of `objective` and `bounds` are set globally, but they may be arbitrary in general.