CEGAR-Based Approach for Solving Combinatorial Optimization Modulo Quantified Linear Arithmetics Problems

Kerian Thuillier, Anne Siegel, Loïc Paulevé

CEGAR-Based Approach for Solving Combinatorial Optimization Modulo Quantified Linear Arithmetics Problems

Kerian Thuillier[1], **Anne Siegel**[1], **Loïc Paulevé**[2]

[1] Univ. Rennes, Inria, CNRS, IRISA, UMR6074, F-35000 Rennes, France
[2] Univ. Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR5800, F-33400 Talence, France
kerian.thuillier@irisa.fr, anne.siegel@irisa.fr, loic.pauleve@labri.fr

Abstract

Bioinformatics has always been a prolific domain for generating complex satisfiability and optimization problems. For instance, the synthesis of multi-scale models of biological networks has recently been associated with the resolution of optimization problems mixing Boolean logic and universally quantified linear constraints (OPT+qLP), which can be benchmarked on real-world models. In this paper, we introduce a Counter-Example-Guided Abstraction Refinement (CEGAR) to solve such problems efficiently. Our CEGAR exploits monotone properties inherent to linear optimization in order to generalize counter-examples of Boolean relaxations. We implemented our approach by extending *Answer Set Programming* (ASP) solver CLINGO with a quantified linear constraints propagator. Our prototype enables exploiting independence of sub-formulas to further exploit the generalization of counter-examples. We evaluate the impact of refinement and partitioning on two sets of OPT+qLP problems inspired by system biology. Additionally, we conducted a comparison with the state-of-the-art ASP solver *Clingo[lpx]* that handles non-quantified linear constraints, showing the advantage of our CEGAR approach for solving large problems.

Introduction

Satisfiability (SAT) solving has proven to be highly successful in addressing a wide range of real-world combinatorial satisfiability problems across various fields. In the last decades, many applications in bioinformatics have been formulated as complex combinatorial satisfiability and optimization problems according to biological knowledge and data. For decision-aided tasks, life-scientists then take advantage of sampling the full space of solutions in order to prioritize future experiments. Therefore, challenges reside both in solving such complex combinatorial problems on large-scale and real-world instances but also in enumerating part, if not all, the set of solutions.

Traditionally, the problems addressed in life-sciences were either linear programming and optimization (LP) problems (Orth, Thiele, and Palsson 2010; von Kamp and Klamt 2014) or Boolean optimization problems (Videla et al. 2017; Chevalier et al. 2019). In this case, efficient approaches based on *Answer Set Programming* (ASP), a logic programming framework for symbolic satisfiability problems (Baral 2003), have been developed. They take advantage of the ability of modern ASP solvers, like *Clingo* (Gebser et al. 2017), to support various reasoning modes, Boolean optimization, and model enumeration.

A recent evolution in life-sciences is the emergence of hybrid optimization problems combining Boolean logic and linear constraints (Frioux et al. 2019; Mahout, Carlson, and Peres 2020). ASP solvers handling quantifier-free linear constraints, like *Clingo[lpx]* (Janhunen et al. 2017), have been developed to solve such hybrid optimization problems, by extending ASP solver with a DPLL-adapted simplex algorithm (Dutertre and De Moura 2006) used by modern *Satisfiability Modulo Theory* (SMT) solvers. A new class of complexity appeared recently with the problem of inferring metabolic regulatory rules, which is formulated as a hybrid optimization problem with one level of *quantified* linear constraints (Thuillier et al. 2022) and associated with real-world benchmarks. The goal of this paper is to investigate efficient solutions to solve this new class of hybrid optimization problems, which we denote as OPT+qLP.

The state-of-the-art strategy to solve OPT+qLP problems is to rely on quantifier elimination to get back to quantifier-free hybrid optimization problems. There is an equivalence between universally quantified linear constraints and constraints on the optimum of LP problems. Hence, based on the *strong duality theorem*, universally quantified linear constraints can be converted into equi-satisfiable quantifier-free linear constraints through a dual transformation. This allows tackling OPT+qLP problems with standard hybrid approaches, as offered by *Clingo[lpx]* and SMT solvers.

An alternative lies in the *Counter-Example-Guided Abstraction Refinement* (CEGAR) method (Clarke et al. 2003). While sharing similarities with the DPLL algorithm (Nieuwenhuis, Oliveras, and Tinelli 2006) used in modern SMT solvers, the CEGAR approach enables to easily compose solvers for different tasks, including for Boolean optimization and enumeration problems. The strength of the CEGAR approach therefore lies in its generic and solver-independent nature, which allows for taking advantage of the structure of linear problems. It has been widely applied for the solving of quantified Boolean formula (Janota et al. 2016), and SMT problems (Brummayer and Biere 2008; Barrett and Tinelli 2018). However, CEGAR approaches have not been applied so far to OPT+qLP problems.

In this paper, we introduce a CEGAR-based algorithm to solve and enumerate models of OPT+qLP problems. Our ap-

proach refines Boolean abstraction of the OPT+qLP problem using monotone properties on LP problems structures and linear constraints partitioning. We rely on the resolution of a formula, a Boolean abstraction, that subsumes the models of the OPT+qLP problem. If this abstraction is unsatisfiable, then so is the OPT+qLP problem. Otherwise, a model of the Boolean abstraction is found. This model is a solution to the OPT+qLP problem if it satisfies the quantified linear constraints. Otherwise, it is a counter-example, and the abstraction is refined with additional constraints derived from the counter-example. This iterative process continues until either the OPT+qLP problem is proven to be unsatisfiable or all its models have been enumerated. To implement it, we developed a prototype based on ASP and evaluated its performance on real-world benchmarks based on biological models. Additionally, we conducted a comparison with *Clingo[lpx]* and compared the performance regarding both quantifier elimination and linear constraints partitioning.

Combinatorial Optimization Problems Modulo Quantified Linear Constraints

We focus on combinatorial optimization problems whose constraints merge propositional logic and quantified linear arithmetics (OPT+qLP). The quantified linear constraints are restricted to one level of quantifier. Solving OPT+qLP problems aims at finding variable assignments, or models, satisfying SAT+qLP constraints while minimizing a given objective function.

Let $x \in \mathbb{B}^n$ denotes Boolean variables and $y \in \mathbb{R}^m$ real-valued variables. We consider SAT+qLP formulas of the following form:

$$\bigwedge_{c \in C} c(x) \tag{1a}$$

$$\wedge \quad \bigwedge_{d \in D} d(x, y) \tag{1b}$$

$$\wedge \quad \forall z \in \mathbb{R}^p, \bigwedge_{e \in E} e(x, z) \implies \bigwedge_{h \in H} h(x, z) \tag{1c}$$

where C denotes Boolean clauses of the form $\bigvee_i x_i \bigvee_j \neg x_j$, and D (resp. E, H) denotes hybrid clauses of the form "$\bigvee_i x_i \bigvee_j \neg x_j \vee f(y) \leq 0$" (resp. $\bigvee_i x_i \bigvee_j \neg x_j \vee f(z) \leq 0$), with f denoting linear functions over reals. Given a hybrid clause $c \in D, E, H$, we will denote by f_c its linear constraint $f_c(y) \leq 0$ (resp. $f_c(z) \leq 0$).

Universally quantified linear constraints are modeled by Eq. 1c. The first part of the implication ($\bigwedge_{e \in E} e(x, z)$) defines the domain $\mathbb{D}(x)$ of the universal real-valued variables z according to x. The domain $\mathbb{D}(x)$ is a subset of $\mathbb{R}^p$, and contains all $z \in \mathbb{R}^p$ such that (x, z) satisfy $\bigwedge_{e \in E} e(x, z)$. Eq. 1c is therefore equivalent to $\forall z \in \mathbb{D}(x), \bigwedge_{h \in H} h(x, z)$.

Let ϕ be a SAT+qLP formula of the form of Eq. 1. A variable assignment $(x, y) \in \mathbb{B}^n \times \mathbb{R}^m$ is a model of ϕ if and only if it satisfies ϕ, *i.e.* $(x, y) \models \phi$. The formula ϕ is unsatisfiable, denoted by $\not\models \phi$, if there is no model ν satisfying ϕ. Otherwise, ϕ is satisfiable.

The SAT+qLP satisfiability problem can be extended into an OPT+qLP optimization problem by considering only the models (x, y) of ϕ that minimize an objective function over Boolean variables $g : \mathbb{B}^n \to \mathbb{R}$:

$$\begin{aligned} \text{minimize} \quad & g(x) && \text{(2a)} \\ \text{such that:} \quad & (x, y) \models \phi && \text{(2b)} \\ \text{with} \quad & x \in \mathbb{B}^n, y \in \mathbb{R}^m \end{aligned}$$

For the rest, let (g, ϕ) be an instance of an OPT+qLP problem. A pair $(x, y) \in \mathbb{B}^n \times \mathbb{R}^m$ is a model of (g, ϕ), denoted by $(x, y) \models (g, \phi)$, if and only if Eqs. 2a and 2b are verified.

Many applications can benefit from a comprehensive characterization of the solution space of satisfiability and optimization problems. Thus, in addition to *searching* for a model of an OPT + qLP problem, we will also consider the *enumeration* up to k different models of it.

Example. Let ψ be the SAT+qLP formula of Fig. 1a over Boolean variables x_1, x_2, x_3. It has no existentially quantified real-valued variables and 2 universally quantified real-valued variables z_1, z_2. Using the notations of Eq. 1, ψ has 1 Boolean ($C = \{(x_1 \vee x_2 \vee x_3)\}$) and 4 hybrid clauses ($D = \emptyset$, $E = \{(z_2 \geq 1 \vee \neg x_1), (z_1 + z_2 \leq 1 \vee \neg x_2), (-z_1 + z_2 \leq 0 \vee \neg x_3)\}$, $H = \{(z_2 \leq 0.6)\}$). Fig. 1b gives a graphical representation of the linear constraints.

For the rest, we will write a model ν as a set such that a Boolean variable x_i belongs to ν if and only if $x_i = \top$. Among the 8 models of ψ, only 2 satisfy it: $\nu_1 = \{x_2, x_3\}$ and $\nu_2 = \{x_1, x_2, x_3\}$. For the former, the set of hybrid clauses E is true if and only if at least $z_1 + z_2 \leq 1$ and $-z_1 + z_2 \leq 0$ hold. As shown in Fig. 1b, all assignments of (z_1, z_2) matching these two constraints satisfy $z_2 \leq 0.6$. For the latter, it does not exist an assignment of (z_1, z_2) that satisfies all hybrid clauses in E.

Let $g : \mathbb{B}^3 \to \mathbb{R}$ be an objective function such that $g(x_1, x_2, x_3) = |x_1| + |x_2| + |x_3|$ with $|x_i| = 1$ if $x_i = \top$, 0 else. Let (g, ψ) be an OPT+qLP problem. Its only model is $\{x_2, x_3\}$ ($g(\{x_2, x_3\}) = 2$ and $g(\{x_1, x_2, x_3\}) = 3$).

Contribution: A CEGAR for Solving OPT+qLP

We present a CEGAR-based approach for addressing OPT+qLP problems. Algorithm 1 summarizes the overall procedure. First, we define a Boolean abstraction $(g, \phi_{\text{approx}})$ of the OPT+qLP problem (g, ϕ), such that $(g, \phi) \implies (g, \phi_{\text{approx}})$ (line 2, see details below). Next, we introduce two necessary conditions (lines 3 and 4, see details below) to ensure that there exists a model of (g, ϕ) given a model of $(g, \phi_{\text{approx}})$. If at least one of the two conditions fails, then ϕ_{approx} is refined by generalizing the counter-examples that fail them (line 8, see details below). Finally, we propose a quantified linear constraints partitioning method to increase the efficiency of refinement functions.

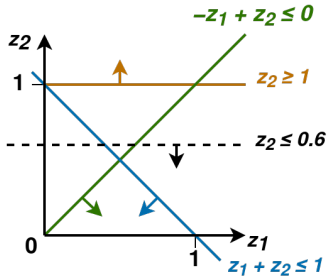
Proofs of the properties, lemmas, and theorems of this section are provided in the technical appendix (Thuillier, Siegel, and Paulevé 2023).

Boolean Abstractions of OPT+qLP Problems

Let c be a hybrid clause over Boolean variables $x \in \mathbb{B}^n$ and real-valued variables $y \in \mathbb{R}^m$ of the form "$\bigvee_i x_i \bigvee_j \neg x_j \vee f_c(y) \leq 0$". A Boolean abstraction $\bar{c}$ of c is a Boolean clause

$$\psi = (x_1 \vee x_2 \vee x_3)$$
$$\wedge\, \forall z \in \mathbb{R}^2, \begin{pmatrix} & (z_2 \geq 1 \vee \neg x_1) \\ \wedge & (z_1 + z_2 \leq 1 \vee \neg x_2) \\ \wedge & (-z_1 + z_2 \leq 0 \vee \neg x_3) \end{pmatrix} \Longrightarrow z_2 \leq 0.6$$

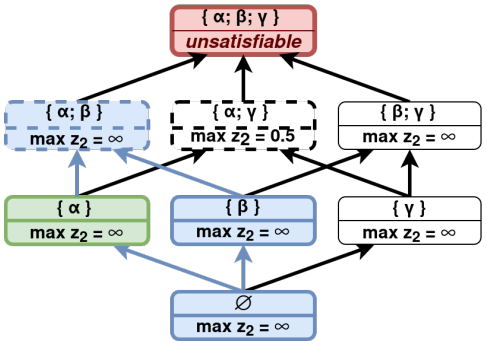
(a) Example SAT+qLP problem ψ.



(b) Visual representation of the quantified linear constraints. No assignments of z_1 and z_2 can satisfy the three linear constraints $z_2 \geq 1$, $z_1 + z_2 \leq 1$ and $-z_1 + z_2 \leq 0$.

$$\psi_{\text{approx}} = (x_1 \vee x_2 \vee x_3)$$
$$\wedge\, (\alpha \vee \neg x_1) \wedge (\beta \vee \neg x_2) \wedge (\gamma \vee \neg x_3) \wedge \delta$$

(c) Boolean abstraction ψ_{approx} of ψ described in **(a)**.



(d) Hasse diagram of all the quantified linear constraints subsets of the example OPT+qLP problem (Fig. 1a) with their optimums. Red block is unsatisfiable. Blocks with dashed borders are the optimal cores of the green block. Blue blocks are the subsets of $\{\alpha; \beta\}$.

Figure 1: Example of SAT+qLP formula ψ (a) over three Boolean variables (x_1, x_2, x_3) and two universally quantified real-valued variables (z_1, z_2). Visual representations of the four linear constraints involved in ψ are shown in (b). In (c) and (d), $\alpha, \beta, \gamma, \delta$ are Boolean variables associated with the linear constraints $z_2 \geq 1$, $z_1 + z_2 \leq 1$, $-z_1 + z_2 \leq 0$ and $z_2 \leq 0.6$, respectively. The Boolean abstraction ψ_{approx} is defined in (c) following Eqs. 4. (d) shows the maximum value of z_2 for each subset of linear constraints.

Algorithm 1: CEGAR for solving OPT+qLP problem

Input: an OPT+qLP problem (g, ϕ) of the form Eq. 2
Output: a model $(x, y) \in \mathbb{B}^n \times \mathbb{R}^m$ s.t. $(x, y) \models (g, \phi)$

1: $\phi_{\text{approx}} \leftarrow$ a Boolean abstraction of ϕ of the form Eq. 4
2: **while** $\exists (x, \bar{f}) \models (g, \phi_{\text{approx}})$ **do**
3: **if** $\exists y \models \mathcal{C}_x^D$ **then**
4: **if** $\not\models \mathcal{C}_x^E$ or $\forall h \in \mathcal{C}_x^H, f_h^*(\mathcal{C}_x^E) \leq 0$ **then**
5: **return** x, y
6: **end if**
7: **end if**
8: $\phi_{\text{approx}} \leftarrow \phi_r^{\exists}(x) \wedge \phi_r^{\forall}(x) \wedge \phi_{\text{approx}}$
9: **end while**
10: **return** UNSAT

over the Boolean variables $x \in \mathbb{B}^n$ and $\bar{f}_c \in \mathbb{B}$. The clause $\bar{c}$ is defined by Eq. 3.

$$\bigvee_i x_i \bigvee_j \neg x_j \vee \bar{f}_c \quad \text{denoted by } \bar{c}(x, \bar{f}_c) \tag{3}$$

Let ϕ be a SAT+qLP formula with C its set of Boolean clauses and D, E, H its sets of hybrid clauses. Let $\bar{d}$, $\bar{e}$ and $\bar{h}$ denote Boolean abstractions of the hybrid clauses $d \in D$, $e \in E$ and $h \in H$, respectively. We define the *Boolean abstraction* of ϕ as the following SAT formula:

$$\bigwedge_{c \in C} c(x) \tag{4a}$$
$$\wedge \bigwedge_{d \in D} \bar{d}(x, \bar{f}_d) \tag{4b}$$
$$\wedge \bigwedge_{e \in E} \bar{e}(x, \bar{f}_e) \wedge \bigwedge_{h \in H} \bar{h}(x, \bar{f}_h) \tag{4c}$$

Theorem 1 ($\phi \Rightarrow \phi_{\text{approx}}$). *Let ϕ a SAT+qLP problem and ϕ_{approx} its Boolean abstraction. For any model $(x, y) \in \mathbb{B}^n \times \mathbb{R}^m$ of ϕ, there exists $\bar{f} \in \mathbb{B}^{|D|+|E|+|H|}$ such that $(x, \bar{f})$ is a model of ϕ_{approx}.*

From the above theorem, one can remark that the value $g(x)$ of the objective function on any model (x, y) of an OPT+qLP problem (g, ϕ) is the same on the corresponding model of ϕ_{approx}. In Algorithm 1, the abstraction $(g, \phi_{\text{approx}})$ of the OPT+qLP problem (g, ϕ) is computed line 1. In line 2, the search for $(g, \phi_{\text{approx}})$ models can be performed using a pure Boolean optimization solver. By Theorem 1, if $(g, \phi_{\text{approx}})$ is unsatisfiable, then so is (g, ϕ).

Example. Consider the OPT+qLP problem (g, ψ) from the previous example. Let α, β, γ, δ be four Boolean variables associated with the linear constraints $z_2 \geq 1$, $z_1 + z_2 \leq 1$, $-z_1 + z_2 \leq 0$ and $z_2 \leq 0.6$, respectively. The set of Boolean variables associated with linear constraints is $\bar{f} =$

$\{\alpha, \beta, \gamma, \delta\}$. The Boolean abstraction of ψ is the SAT formula ψ_{approx} defined by Fig. 1c. Formula ψ has two models $\nu_1 = \{x_2, x_3\}$ and $\nu_2 = \{x_1, x_2, x_3\}$. Using the conversion procedure used to prove Theorem 1, $\bar{\nu}_1 = \{x_2, x_3, \beta, \gamma, \delta\}$ and $\bar{\nu}_2 = \{x_1, x_2, x_3, \alpha, \beta, \gamma, \delta\}$ are two models of ψ_{approx}. The model ν_1 is the only model of (g, ψ). It has the optimal score $g^* = 2$. The model $\bar{\nu}_1$ associated with ν_1 has the same score.

Ensuring Quantified Linear Constraints

Let $\mathcal{C}$ be a set of linear constraints of the form $f(y) \leq 0$. A variable assignment $y \in \mathbb{R}^m$ is a model of $\mathcal{C}$, denoted by $y \models \mathcal{C}$, if and only if $y \models \bigwedge_{f \in \mathcal{C}} f(y) \leq 0$. Given $f : \mathbb{R}^m \to \mathbb{R}$ a linear function, $y \in \mathbb{R}^m$ is a model of the *linear optimization problem* $(f, \mathcal{C})$ if and only if $y \models \mathcal{C}$ and it maximizes the objective function f, *i.e.* $\forall y' \in \mathbb{R}^m, y' \models \mathcal{C} \implies f(y') \leq f(y)$. The optimum value of $(f, \mathcal{C})$ will be denoted by $f^*(\mathcal{C}) = \max_{y \models \mathcal{C}} f(y)$.

Let C_h be a set of hybrid clauses and $x \in \mathbb{B}^n$ a Boolean variable assignment. For x to be a model of C_h, it must exist $y \in \mathbb{R}^m$ such that each hybrid clause $h \in C_h$ is satisfied by either x or y. Let $\mathcal{C}_x^{C_h}$ be the set of linear constraints of clauses for which x is not a model:

$$\mathcal{C}_x^{C_h} = \{f_c(y) \leq 0 | c \in C_h, x \not\models c\} \tag{5}$$

Hence, given $c \in C_h$ and $(x, \bar{f}_c) \models \bar{c}(x, \bar{f}_c)$, if $f_c \in \mathcal{C}_x^{C_h}$ then $\bar{f}_c = \top$. Otherwise, x would be a model of $\bar{c}(x, \bar{f}_c)$.

Theorem 2. *Let ϕ be a* SAT+*q*LP *formula and ϕ_{approx} its Boolean abstraction. Given $x \in \mathbb{B}^n$ and $y \in \mathbb{R}^m$, $(x, y) \models \phi$ if and only if the following three conditions hold:* ***(C1)*** *$\exists \bar{f}, (x, \bar{f}) \models \phi_{approx}$;* ***(C2)*** *$y \models \mathcal{C}_x^D$;* ***(C3)*** *$(\not\models \mathcal{C}_x^E) \vee (\bigwedge_{h \in \mathcal{C}_x^H} f_h^*(\mathcal{C}_x^E) \leq 0)$.*

Theorem 2 can be further extended for OPT+qLP problems. Let (g, ϕ) be an OPT+qLP problem and $(g, \phi_{\text{approx}})$ its Boolean abstraction. Any variable assignment $(x, y) \in \mathbb{B}^n \times \mathbb{R}^m$ minimizing g and satisfying *C1*, *C2* and *C3* is a model of (g, ϕ).

Corollary 2.1. *Given $x \in \mathbb{B}^n$ and $y \in \mathbb{R}^m$ a real-valued variables assignment, if* ***(C1')*** *$\exists \bar{f}, (x, \bar{f}) \models (g, \phi_{approx})$,* ***C2*** *and* ***C3*** *hold, then $(x, y) \models (g, \phi)$.*

In Algorithm 1, the condition *C1'* is ensured if a model $(x, \bar{f})$ of $(g, \phi_{\text{approx}})$ is found (line 2). Condition *C2* is ensured in line 3 by finding a model y of the set of linear constraints $\mathcal{C}_x^D$ using a linear programming (LP) solver. *C2* holds only if y exists. Finally, condition *C3* is ensured in line 4. If $\mathcal{C}_x^E$ is satisfiable, a linear optimization problem $(f_h, \mathcal{C}_x^E)$ is solved for each $f_h \in \mathcal{C}_x^H$. The linear optimization problems are solved using LP solvers. Each optimum $f_h^*(\mathcal{C}_x^E)$ is then compared to 0. If at least one optimum is strictly greater than 0, then *C3* does not hold. If the three conditions *C1'*, *C2* and *C3* hold, $(x, y) \models (g, \phi)$ is returned. Otherwise, $(x, \bar{f})$ is a counter-example.

Example. Consider the OPT+qLP problem (g, ψ) and its Boolean abstraction ψ_{approx} (Fig. 1c) from the previous example. The variable assignment $\{x_1, \alpha, \delta\}$ is a model of ψ_{approx} that minimize g, with $g(\{x_1, \alpha\}) = 1$. By Corollary 2.1, $\{x_1\}$ is also a model of (g, ψ) if either $\not\models \{z_2 \geq 1\}$ or if the linear optimization problem $(f_\delta(z_1, z_2) = z_2, \{z_2 \geq 1\})$ has an optimum less or equals to 0.6. From Fig. 1b, we can see that $\{z_2 \geq 1\}$ is satisfiable and that $f_\delta^*(\{z_2 \geq 1\})$ is $+\infty$. Therefore, *C3* does not hold and $\{x_1, \delta\}$ is not a model of (g, ψ). The variable assignment $\{x_1, \alpha, \delta\}$ is a counter-example.

Counter-Examples Generalization

Let ϕ be a SAT+qLP formula and ϕ_{approx} its Boolean abstraction. Theorem 2 states that for any model $\bar{\nu} = (x, \bar{f})$ of ϕ_{approx} there is a corresponding model ν of ϕ if conditions *C2* and *C3* hold. If either *C2* or *C3* is not satisfied, then $\bar{\nu}$ is a counter-example. From $\bar{\nu}$, new Boolean logic constraints $\phi_r(\bar{\nu})$ can be deduced and used to refine ϕ_{approx}. The new Boolean abstraction of ϕ becomes $\phi_{\text{approx}} \wedge \phi_r(\bar{\nu})$, such that $\phi \implies \phi_{\text{approx}} \wedge \phi_r(\bar{\nu})$.

Existential counter-example. Suppose that $(x, \bar{f})$ does not satisfy *C2*. The set of linear constraints $\mathcal{C}_x^D$ is unsatisfiable, *i.e.* $\not\models \mathcal{C}_x^D$. Therefore, any supersets of linear constraints of $\mathcal{C}_x^D$ will be unsatisfiable too. An *unsatisfiable core* ($\mathcal{C}_{\text{unsat}}$) of a given set of linear constraints $\mathcal{C}$ is the smallest subset of $\mathcal{C}$ for which $\not\models \mathcal{C}_{\text{unsat}}$. In other words, for all $\mathcal{C}' \subset \mathcal{C}_{\text{unsat}}$, there exists a vector $y \in \mathbb{R}^m$ that satisfies $\mathcal{C}'$. When $\mathcal{C}$ is satisfiable, $\mathcal{C}_{\text{unsat}}$ is an empty set. Unsatisfiable cores have been widely used in SMT solvers and CEGAR-based approaches for generalizing sets of unsatisfiable constraints (Cimatti, Griggio, and Sebastiani 2011; Khasidashvili, Korovin, and Tsarkov 2015).

Let $\mathcal{C}_{\text{unsat}}$ be an unsatisfiable core of $\mathcal{C}_x^D$. The refinement function $\phi_r^\exists(x)$ is defined by Eq. 6.

$$\phi_r^\exists(x) = \bigvee_{f \in \mathcal{C}_{\text{unsat}}} \neg \bar{f} \tag{6}$$

Note that refinement function $\phi_r^\exists(x)$ does not generate any constraints if *C2* holds ($\mathcal{C}_{\text{unsat}} = \emptyset$).

Lemma 3. $\phi \implies \phi_{approx} \wedge \phi_r^\exists(x)$.

Universal counter-example. Suppose that $(x, \bar{f})$ does not satisfy *C3*. This implies that there is at least one hybrid clause $h \in H$ such that $\mathcal{C}_x^E$ is satisfiable and $f_h^*(\mathcal{C}_x^E) > 0$. Then, any model (x', y') such that $\mathcal{C}_{x'}^E \subseteq \mathcal{C}_x^E$ will be such $f_h^*(\mathcal{C}_{x'}^E) > 0$, as stated by the following property:

Property 4. *Given a linear objective function $\boldsymbol{f}$ and two linear optimization problems $(\boldsymbol{f}, \mathcal{C}_1)$ and $(\boldsymbol{f}, \mathcal{C}_2)$, $\mathcal{C}_1 \subseteq \mathcal{C}_2 \implies \boldsymbol{f}^*(\mathcal{C}_1) \geq \boldsymbol{f}^*(\mathcal{C}_2)$.*

Similarly to unsatisfiable cores, we can introduce the notion of optimal cores. Given a linear objective function f and a set of linear constraints $\mathcal{C}$, an *optimal core* is a biggest superset $\mathcal{C}_{\text{opt}}^f$ of $\mathcal{C}$ such that $\mathcal{C}_{\text{opt}}^f$ is satisfiable and $f^*(C) = f^*(C_{\text{opt}}^f)$.

Let $\mathcal{C}_{\text{opt}}^f$ be an optimal core of $(f, \mathcal{C}_x^E)$. The refinement function $\phi_r^\forall(x)$ is defined by Eq. 7.

$$\phi_r^\forall(x) = \bigwedge_{\substack{h \in \mathcal{C}_x^H \\ f_h^*(\mathcal{C}_x^E) > 0}} \neg \bar{f}_h \vee \bigvee_{\substack{e \in E \\ f_e \notin \mathcal{C}_{\text{opt}}^{f_h}}} \bar{f}_e \tag{7}$$

Lemma 5. $\phi \implies \phi_{approx} \wedge \phi_r^{\forall}(x)$

Constraints generated by the refinement functions $\phi_r^{\exists}(x)$ and $\phi_r^{\forall}(x)$ do not involve the same sets of variables. Therefore, $\phi_r^{\exists}(x) \wedge \phi_r^{\forall}(x) \wedge \phi_{\text{approx}}$ still subsumes ϕ.

Theorem 6. *Given* $(x, \bar{f}) \models \phi_{approx}$, $\phi \implies \phi_r^{\exists}(x) \wedge \phi_r^{\forall}(x) \wedge \phi_{approx}$.

Corollary 6.1. $(g, \phi) \implies \phi_r^{\exists}(x) \wedge \phi_r^{\forall}(x) \wedge \phi_{approx}$.

Corollary 6.2. $\forall \nu^* \models (g, \phi) \implies \exists \nu' \models \phi_r^{\exists}(x) \wedge \phi_r^{\forall}(x) \wedge \phi_{approx}, g(\nu') = g(\nu^*)$.

Algorithm 1 refines the Boolean abstraction ϕ_{approx} in line 8. Corollaries 6.1 and 6.2 ensure that the refined Boolean abstraction is still an overapproximation of (g, ϕ). Therefore, Corollary 2.1 still holds for the next iteration.

Example. Consider ψ_{approx} as defined in Fig. 1c and the counter-example $\{x_1, \alpha, \delta\}$ find previously. This counter-example satisfies *C2* since there are no existentially quantified linear constraints in ψ. Hence, $\phi_r^{\exists}(\{x_1\})$ does not generate any constraints. However, it fails to satisfy *C3*. A Hasse diagram of all the subsets of the set of linear constraints of ψ is shown in Fig. 1d. It can be seen that $\{\alpha\}$ has two optimal cores: $\{\alpha, \beta\}$ and $\{\alpha, \gamma\}$. The set $\{\alpha, \beta, \gamma\}$ is not an optimal core since it is not satisfiable. All linear optimization problems whose linear constraints are either a subset of $\{\alpha, \beta\}$ or of $\{\alpha, \gamma\}$ will also fail *C3*. Suppose that the optimal core $\{\alpha, \beta\}$ has been selected by the refinement function $\phi_r^{\forall}(\{x_1\})$. It will generate the constraints $\neg\delta \vee \gamma$, and it will prohibit selecting any model containing a subset of $\{\alpha, \beta\}$, blue and green boxes in Fig. 1d.

Partitioning Quantified Linear Constraints

Let (g, ϕ) be an OPT+qLP problems with $(g, \phi_{\text{approx}})$ its Boolean abstraction. Linear constraints of ϕ can be partitioned to exploit the sparsity of the underlying linear optimization problems. Let $\mathcal{P} = \{\mathcal{P}_1, ..., \mathcal{P}_k\}$ be a partition of the linear constraints of ϕ such that **(i)** no two linear constraints share variables among different subsets; **(ii)** each subset contains either existentially quantified linear constraints or universally quantified linear constraints.

Let $(x, \bar{f}) \models (g, \phi_{\text{approx}})$. The set of linear constraints $\mathcal{C}_x^D$ can be partitioned in $\mathcal{P}_x^D$ according to the partition $\mathcal{P}$. Deciding the satisfiability of $\mathcal{C}_x^D$ comes down to deciding the satisfiability of each subset $\mathcal{P}_i \in \mathcal{P}_x^D$. If at least one subset is unsatisfiable, so is $\mathcal{C}_x^D$. Otherwise, it exists a model y_i for each subset $\mathcal{P}_i \in \mathcal{P}_x^D$ and $\{y_i\}_i \models \mathcal{C}_x^D$.

Lemma 7. $\exists y \in \mathbb{R}^m, \mathcal{C}_x^D \iff \bigwedge_{\mathcal{P}_i \in \mathcal{P}_x^D} y \models \mathcal{P}_i$.

If $(x, \bar{f})$ fails *C2*, one can exhibit a subset of sets of $\mathcal{P}_x^D$ that are unsatisfiable. Unsatisfiable cores can be computed independently for each unsatisfiable set, which reduces the computational cost of finding unsatisfiable cores. Let $\mathbb{C}_{\text{unsat}}$ be the set of unsatisfiable cores associated with the unsatisfiable sets. The existential refinement function $\phi_r^{\exists}(x)$ can be reformulated as:

$$\phi_r^{\exists}(x) = \bigwedge_{\mathcal{C}_{\text{unsat}} \in \mathbb{C}_{\text{unsat}}} \bigvee_{f \in \mathcal{C}_{\text{unsat}}} \neg \bar{f} \tag{8}$$

Benchmark	*Small-scale*	*Large-scale*
Instances SAT	29	32
Instances UNSAT	31	28
Boolean variables	6.5×10^4	4×10^9
Existential real variables	2×10^3	8×10^3
Universal real variables	2×10^3	8×10^3
Boolean constraints	2.7×10^5	1.8×10^6
Existential linear constraints	6×10^3	25×10^3
Universal linear constraints	6×10^3	25×10^3

Table 1: Benchmarks descriptions. Only the order of magnitude of the number of constraints and variables is given.

Similarly, all linear constraints $f_h \in \mathcal{C}_x^H$ are partitioned with the linear constraints of $\mathcal{C}_x^E$ that can impact their values. Let $\mathcal{P}' \in \mathcal{P}$ be the partitioned containing f_h and $\mathcal{P}_x'^E$ the set of all linear constraints of $\mathcal{C}_x^E$ in $\mathcal{P}'$.

Lemma 8. *If* $\mathcal{C}_x^E$ *is satisfiable, then* $f_h^*(\mathcal{C}_x^E) = f_h^*(\mathcal{P}_x'^E)$.

If $(x, \bar{f})$ fails *C3*, it is necessarily since there is not enough constraints in $\mathcal{P}_x'^E$. Since only linear constraints in $\mathcal{P}'$ have an impact on f_h^*, the computation of an optimal core $\mathcal{P}'_{\text{opt}}$ can be restricted to the set of linear constraints in $\mathcal{P}'$. The universal refinement function $\phi_r^{\forall}(x)$ can be reformulated as:

$$\phi_r^{\forall}(x) = \bigwedge_{\substack{h \in \mathcal{C}_x^H \\ f_h^*(\mathcal{P}_x'^E) > 0}} \neg \bar{f}_h \vee \bigvee_{\substack{e \in E \\ f_e \notin \mathcal{P}'_{\text{opt}}}} \bar{f}_e \tag{9}$$

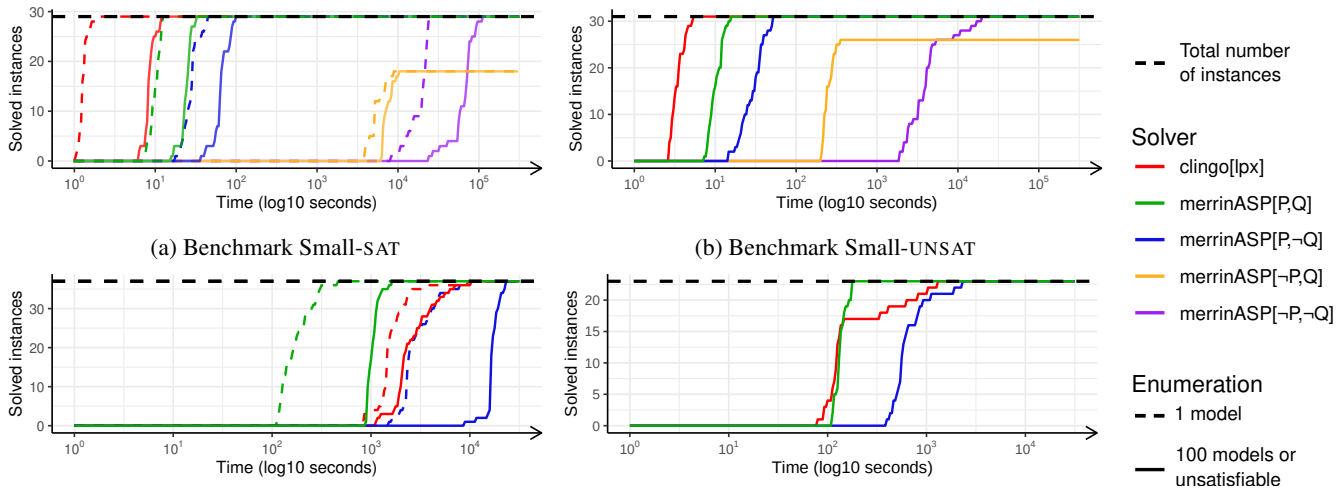
It is important to note that Theorem 6 still holds with these new definitions of $\phi_r^{\exists}$ and $\phi_r^{\forall}$. They generate smaller refinement constraints and allow reducing the computational cost of finding unsatisfiable and optimal cores.

Experiments

We propose MERRINASP (https://github.com/kthuillier/merrinasp), an ASP-based implementation of Algorithm 1. It extends the *Clingo* solver, using its *Python* API, with a linear constraint propagator, implemented with the *Python* PULP library and the LP solver COIN (Lougee-Heimer 2003). Model enumeration is made through the *Clingo* solver which keeps track of all refinements during the enumeration process. The partitioning is explicitly specified in the input problem.

Benchmark

Problem description. Regulatory flux balance analysis (rFBA) is a common model of dynamics of bacteria (Covert, Schilling, and Palsson 2001). The rFBA framework consists in sequentially solving maximum flow problems on weighted hypergraphs. The hyperedge capacities are updated at each step according to Boolean rules. Capacities are either set to 0 or to their initial value. The metabolic regulatory rules inference problem (Thuillier et al. 2022) is an inverse problem. Given a weighted hypergraph and sequences of observed maximum flows, it consists in inferring a set of Boolean rules controlling the hyperedge capacities matching the sequences of observations. For each



(a) Benchmark Small-SAT

(b) Benchmark Small-UNSAT

(c) Benchmark Large-SAT

(d) Benchmark Large-UNSAT

Figure 2: Runtime distribution of 4 configurations of our MERRINASP implementation of the CEGAR-based Algorithm 1 and *Clingo[lpx]* on OPT+QLP problem instances. All variants were applied to a benchmark built from a small-scale real biological model (Figs. (a) and (b), 60 instances) and a large-scale real biological model (Figs. (c) and (d), 60 instances). Small-scale and large-scale benchmarks contain both satisfiable instances (panels (a) and (c)) and unsatisfiable instances (panels (b) and (d)). The four configurations of MERRINASP include a partitioning option (P) and the use of universally quantified linear constraints (Q). Time is given in seconds in $\log_{10}$ scale. Dashed black horizontal lines represent the total number of instances

Benchmark	Partitioned (P)	Quantified (Q)	Deciding SAT Time (s)	Enumeration Time (s)	LP solver Time (s)	Number of LP solvers calls	Number of refinements
Small-SAT	×	×	$18\,761 \pm 4\,759$	$49\,952 \pm 18\,515$	$3\,812 \pm 2\,727$	$16\,795 \pm 2\,364$	2 ± 0
	×	✓	$5\,528 \pm 1\,498$	$2\,116 \pm 1\,044$	$1\,433 \pm 223$	$9\,944 \pm 1\,470$	$\mathbf{1 \pm 0}$
	✓	×	28 ± 6	40 ± 11	34 ± 7	937 ± 111	5 ± 1
	✓	✓	$\mathbf{9 \pm 1}$	$\mathbf{15 \pm 3}$	$\mathbf{15 \pm 2}$	$\mathbf{501 \pm 41}$	6 ± 1
Small-UNSAT	×	×	$5\,143 \pm 4\,395$	NA	$1\,112 \pm 766$	$6\,596 \pm 3\,723$	$\mathbf{1 \pm 0}$
	×	✓	247 ± 38	NA	137 ± 17	$2\,039 \pm 115$	$\mathbf{1 \pm 0}$
	✓	×	30 ± 10	NA	24 ± 10	669 ± 221	9 ± 4
	✓	✓	$\mathbf{10 \pm 2}$	NA	$\mathbf{7 \pm 1}$	$\mathbf{252 \pm 54}$	9 ± 4
Large-SAT	✓	×	$3\,163 \pm 1\,538$	$13\,922 \pm 1\,946$	801 ± 236	$17\,957 \pm 5\,032$	41 ± 16
	✓	✓	$\mathbf{183 \pm 75}$	$\mathbf{865 \pm 112}$	$\mathbf{121 \pm 74}$	$\mathbf{3\,548 \pm 2\,184}$	$\mathbf{21 \pm 11}$
Large-UNSAT	✓	×	739 ± 454	NA	374 ± 248	$7\,480 \pm 4\,673$	17 ± 8
	✓	✓	$\mathbf{135 \pm 19}$	NA	$\mathbf{41 \pm 11}$	$\mathbf{1\,155 \pm 307}$	$\mathbf{13 \pm 3}$

Table 2: Comparative analysis of MERRINASP performance under different configurations. Results are presented as average value $\pm$ standard deviation. Deciding SAT times denote the time needed to find a first model or to decide unsatisfiable. NA indicates information not available. Bold values indicate the best value among all configurations for the current benchmark.

observation, it must find which capacities were set to 0 for the maximum flow to match the observation. In this problem, Boolean clauses delimit admissible Boolean rules according to biological knowledge. For each observation, existential constraints ensure the existence of a corresponding flow, while universal constraints ensure that no flow is strictly higher than the observed one. We refer the reader to the above-mentioned paper for a formal definition of the problem.

Benchmark description. We conducted experiments using MERRINASP on real-world benchmarks of metabolic regulatory rules inference problems (Thuillier, Siegel, and Paulevé 2023). Our benchmarks are composed of 120 instances divided into 60 small-scale instances and 60 large-scale instances. The small-scale benchmark is directly sourced from (Thuillier et al. 2022), while the large-scale benchmark is generated based on a large-scale regulated metabolic network (Covert and Palsson 2002), following the methodology outlined in the aforementioned paper. Benchmarks are described in table 1. Instances of the large-scale benchmarks have approximately 10 times more variables and constraints than instances of the small-scale benchmarks. Linear constraints can be partitioned into about 200

sets for small-scale instances and 140 sets for large-scale instances.

Configuration. Each instance was executed on Haswell Intel Xeon E5-2680 v3 CPU at 2.5GHz and 128GB of RAM and 100 models were enumerated.

Results

We compared MERRINASP with *Clingo[lpx]*, a state-of-the-art ASP solver that handles quantifier-free linear constraints (Janhunen et al. 2017) by extending *Clingo* with a DPLL-adapted simplex algorithm (Dutertre and De Moura 2006). *Clingo[lpx]* supports neither linear constraints partitioning nor universal linear constraints. We further conducted a comparative analysis of MERRINASP under four configurations: with and without partioning of linear constraints (denoted by P and $\neg P$), using the CEGAR approach over quantified linear constraints (denoted by Q) or using quantifier elimination ($\neg Q$). Note that *Clingo[lpx]* is equivalent to the configuration $[\neg P, \neg Q]$, and that MERRINASP$[P, Q]$ exploits all the properties described in previous sections.

Comparison with *Clingo[lpx]*. As shown in Fig. 2a and 2b, on small-scale instances, MERRINASP and *Clingo[lpx]* solve the instances in a similar order of magnitude (10s in average for *Clingo[lpx]* and 30s in average for MERRINASP). On large-scale instances, MERRINASP outperforms *Clingo[lpx]* by a factor of 10 (see Figs. 2c and 2d).

As shown in Fig. 2c, MERRINASP excels at finding the first model in large-scale satisfiable instances, outperforming *Clingo[lpx]* by a factor of 30. The difference in performance between the two solvers heavily depends on the enumeration phase. The CEGAR method requires many checks to ensure that a model of the Boolean abstraction is a model of the original OPT+QLP problem, even after reaching equisatisfiability. Consequently, while MERRINASP is significantly faster than *Clingo[lpx]* in finding the first model for satisfiable problems, both solvers exhibit similar performance in enumerating the other 99 models.

Impact of partitioning (P). Figs. 2a and 2b suggest that linear constraints partitioning (P) increase the performance of MERRINASP by a factor of 1000 on satisfiable instances and a factor of 20 on unsatisfiable instances. No instance of the large-scale benchmark has finished in 48 hours for the not-partitioned configurations. Table 2 shows that while partitioning entails solving a larger number of linear optimization problems, the total number of linear optimization problems solved is reduced by a factor of 10 compared to without partitioning. On the small-scale satisfiable (*resp.* unsatisfiable) instances, MERRINASP$[P, Q]$ solved in average 501 (*resp.* 252) linear optimization problems, against 9 944 (*resp.* 2 039) for MERRINASP$[\neg P, Q]$.

Impact of quantified linear constraints (Q). Our counter-example generation for universally quantified linear constraints consistently outperforms quantifier elimination reformulations by a factor of 3 on the small-scale and 20 large-scale benchmarks. From Table 2, we can see that twice fewer refinements are made when using quantified linear constraints (Q) compared to using quantifier elimination ($\neg Q$). For large-scale (*resp.* small-scale) instances, these refinements were generated using 7 (*resp.* 2) times fewer calls to the linear solvers when using (Q) compared to ($\neg Q$).

Discussion. These results highlight that both linear constraint partitioning (P) and counter-example generation for universally quantified linear constraints (Q) have significant impacts on performance. Using both of them allows dividing computation time by 2 000 compared to not using any of them. They allow for generating more efficient refinements (gain of 2) while reducing the number of linear solver calls (gain of 7). This reduction is attributed to the partitioning approach, which enables solving independent linear optimization problems with a reduced number of constraints and variables. Their small size leads to faster computation of unsatisfiable and optimal cores for each counter-example, and their independence allows for reducing the number of verifications: a set that has passed the linear checks does not have to be checked again.

MERRINASP is a prototype and does not use efficient approaches to instantiate and solve linear optimization problems. In contrast, *Clingo[lpx]* and SMT solvers, such as *z3* (De Moura and Bjørner 2008), use an incremental implementation of the simplex algorithm to check linear constraints (Dutertre and De Moura 2006). Our approach is not dependent on the method used to solve linear constraints. This suggests that MERRINASP has the potential to further enhance its performance by integrating these algorithms.

Conclusion and Future Work

In this paper, we presented a novel approach for solving combinatorial optimization problems with Boolean logic and quantified linear constraints (OPT+QLP), based on Counter-Example-Guided Abstraction Refinement (CEGAR). Our implementation, MERRINASP, was developed using Answer Set Programming.

To evaluate the effectiveness of our approach, we introduced a new benchmark of small-scale and large-scale OPT+QLP problems inspired by systems biology. We compared MERRINASP against a state-of-the-art ASP modulo quantifier-free linear constraints solver, *Clingo[lpx]*. The results highlight that MERRINASP scales significantly better than *Clingo[lpx]* on large-scale satisfiable instances, especially for the search of one model on satisfiable instances. The enumeration of models and unsatisfiable instances remain competitive with *Clingo[lpx]* but suggest room of improvement to improve the CEGAR approach and reduce the number of counter-example checks (Brummayer and Biere 2009; Lagniez et al. 2017).

Looking ahead, we plan to automate the linear constraint partitioning process and explore the integration of our approach with the DPLL-based simplex algorithm used in *Clingo[lpx]*. Moreover, the integration of quantified Linear Real Arithmetics theory (LRA) (Reynolds, King, and Kuncak 2017) could provide complementary refinements using linear constraints, while our approach refines by the means of combinatorial constraints. These future advancements hold the promise of further enhancing the efficiency and applicability of CEGAR-based OPT+QLP solvers.

Acknowledgments

Work of KT and LP is supported by the French Agence Nationale pour la Recherche (ANR) in the scope of the project "BNeDiction" (grant number ANR-20-CE45-0001).

References

Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*. New York, NY, USA: Cambridge University Press. ISBN 0521818028.

Barrett, C.; and Tinelli, C. 2018. *Satisfiability modulo theories*. Springer.

Brummayer, R.; and Biere, A. 2008. Lemmas on Demand for the Extensional Theory of Arrays. In *Proceedings of the Joint Workshops of the 6th International Workshop on Satisfiability Modulo Theories and 1st International Workshop on Bit-Precise Reasoning*, SMT '08/BPR '08, 6–11. New York, NY, USA: Association for Computing Machinery. ISBN 9781605584409.

Brummayer, R.; and Biere, A. 2009. Effective bit-width and under-approximation. In *International Conference on Computer Aided Systems Theory*, 304–311. Springer.

Chevalier, S.; Froidevaux, C.; Paulevé, L.; and Zinovyev, A. 2019. Synthesis of Boolean Networks from Biological Dynamical Constraints using Answer-Set Programming. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 34–41.

Cimatti, A.; Griggio, A.; and Sebastiani, R. 2011. Computing small unsatisfiable cores in satisfiability modulo theories. *Journal of Artificial Intelligence Research*, 40: 701–728.

Clarke, E.; Grumberg, O.; Jha, S.; Lu, Y.; and Veith, H. 2003. Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM (JACM)*, 50(5): 752–794.

Covert, M. W.; and Palsson, B. Ø. 2002. Transcriptional Regulation in Constraints-based Metabolic Models of Escherichia coli* 210. *Journal of Biological Chemistry*, 277(31): 28058–28064.

Covert, M. W.; Schilling, C. H.; and Palsson, B. 2001. Regulation of gene expression in flux balance models of metabolism. *Journal of theoretical biology*, 213(1): 73–88.

De Moura, L.; and Bjørner, N. 2008. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, 337–340. Springer.

Dutertre, B.; and De Moura, L. 2006. Integrating simplex with DPLL (T). *Computer Science Laboratory, SRI International, Tech. Rep. SRI-CSL-06-01*.

Frioux, C.; Schaub, T.; Schellhorn, S.; Siegel, A.; and Wanko, P. 2019. Hybrid metabolic network completion. *Theory and Practice of Logic Programming*, 19(1): 83–108.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2017. Multi-shot ASP solving with clingo. *CoRR*, abs/1705.09811.

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

Janota, M.; Klieber, W.; Marques-Silva, J.; and Clarke, E. 2016. Solving QBF with counterexample guided refinement. *Artificial Intelligence*, 234: 1–25.

Khasidashvili, Z.; Korovin, K.; and Tsarkov, D. 2015. EPR-based k-induction with Counterexample Guided Abstraction Refinement. In *GCAI*, 137–150.

Lagniez, J.-M.; Berre, D. L.; de Lima, T.; and Montmirail, V. 2017. A Recursive Shortcut for CEGAR: Application To The Modal Logic K Satisfiability Problem. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 674–680.

Lougee-Heimer, R. 2003. The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1): 57–66.

Mahout, M.; Carlson, R. P.; and Peres, S. 2020. Answer Set Programming for Computing Constraints-Based Elementary Flux Modes: Application to Escherichia coli Core Metabolism. *Processes*, 8(12).

Nieuwenhuis, R.; Oliveras, A.; and Tinelli, C. 2006. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL (T). *Journal of the ACM (JACM)*, 53(6): 937–977.

Orth, J. D.; Thiele, I.; and Palsson, B. Ø. 2010. What is flux balance analysis? *Nature biotechnology*, 28(3): 245–248.

Reynolds, A.; King, T.; and Kuncak, V. 2017. Solving quantified linear arithmetic by counterexample-guided instantiation. *Formal Methods in System Design*, 51: 500–532.

Thuillier, K.; Baroukh, C.; Bockmayr, A.; Cottret, L.; Paulevé, L.; and Siegel, A. 2022. MERRIN: MEtabolic regulation rule INference from time series data. *Bioinformatics*, 38(Supplement_2): ii127–ii133.

Thuillier, K.; Siegel, A.; and Paulevé, L. 2023. CEGAR-based approach for solving combinatorial optimization modulo quantified linear arithmetics problems – Code and Appendix. https://doi.org/10.5281/zenodo.10361533. Accessed: 2023-12.

Videla, S.; Saez-Rodriguez, J.; Guziolowski, C.; and Siegel, A. 2017. caspo: a toolbox for automated reasoning on the response of logical signaling networks families. *Bioinformatics*, 33(6): 947–950.

von Kamp, A.; and Klamt, S. 2014. Enumeration of Smallest Intervention Strategies in Genome-Scale Metabolic Networks. *PLOS Computational Biology*, 10(1): 1–13.