How to solve ASP modulo quantified linear constraints?

CEGAR-based approach for solving combinatorial optimization modulo quantified linear arithmetics problems

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

February 2024

[1] Univ. Rennes, Inria, CNRS, IRISA, Rennes, France
[2] Univ. Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR5800, F-33400 Talence, France

Bioinformatics

A prolific domain that generates complex optimization problems

- Many applications formulated as complex combinatorial satisfiability/optimization problems
  - Models according to biological knowledge
  - (Partial-)Enumeration of the solutions is mandatory

- Traditionally:
  - Linear optimization problems — *use LP/ILP solvers*
  - Boolean optimization problems — *use ASP/SAT solvers*

Recently, new hybrid Boolean logic + linear constraints formulations have emerged

Combinatorial optimization problems

Conjunctive Normal Form definition

$$\text{minimize} \quad f_{\text{obj}}(x)$$
$$\text{such that}$$
$$\bigwedge_{c \in C} c(x)$$

SAT problem
ASP, Boolean Optimization

$$\text{with} \quad x \in \mathbb{B}^n$$

$c(x)$ of the form $\bigvee_i x_i \vee \bigvee_j \neg x_j$

Combinatorial optimization problems

Conjunctive Normal Form definition

$$\text{minimize} \quad f_{\text{obj}}(x)$$
$$\text{such that}$$
$$\bigwedge_{c \in C} c(x)$$
$$\wedge \quad \bigwedge_{d \in D} d(x, y)$$

SAT problem

ASP, Boolean Optimization

SAT problem with LP constraints

SMT solvers, Clingo[lpx]

$$\text{with} \;\; x \in \mathbb{B}^n, y \in \mathbb{R}^m$$

$c(x)$ of the form $\bigvee_i x_i \vee \bigvee_j \neg x_j$

$d(x,y)$ of the form $\bigvee_i x_i \vee \bigvee_j \neg x_j \vee g(y) \leq 0$ with $g(y)$ a linear function

Combinatorial optimization problems

Conjunctive Normal Form definition

$$\begin{aligned}
&\text{minimize} \quad f_{\text{obj}}(x) \\
&\text{such that} \\
&\quad \bigwedge_{c \in C} c(x) \\
&\quad \wedge \quad \bigwedge_{d \in D} d(x, y) \\
&\quad \wedge \quad \forall z \in \mathbb{R}^p, \bigwedge_{e \in E} e(x, z) \implies \bigwedge_{h \in H} h(x, z) \\
&\text{with} \quad x \in \mathbb{B}^n, y \in \mathbb{R}^m
\end{aligned}$$

SAT problem
ASP, Boolean Optimization

SAT problem with LP constraints
SMT solvers, Clingo[lpx]

OPT+qLP
SAT problem with one level of quantified linear constraints
SMT solvers, Clingo[lpx] with quantifier elimination

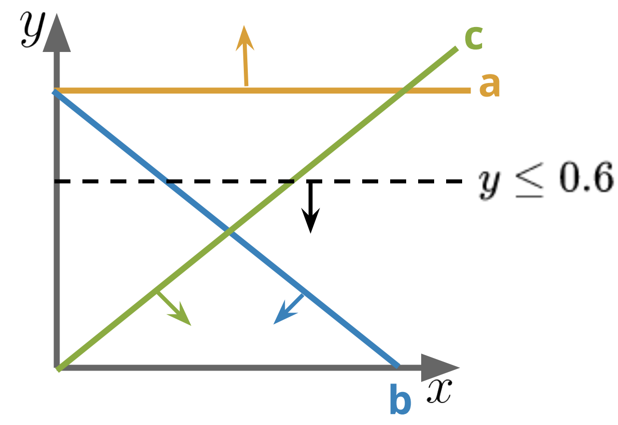
$c(x)$ of the form $\bigvee_i x_i \vee \bigvee_j \neg x_j$

$d(x, y)$ of the form $\bigvee_i x_i \vee \bigvee_j \neg x_j \vee g(y) \leq 0$ with $g(y)$ a linear function

Example of OPT+qLP problem

Definition

$$\text{minimize} \quad a + b + c$$
$$\text{such that}$$
$$(a \lor b \lor c)$$
$$\land \forall x, y \in \mathbb{R}, \begin{pmatrix} (y \geq 1 \lor \neg a) \\ \land \quad (x + y \leq 1 \lor \neg b) \\ \land \quad (-x + y \leq 0 \lor \neg c) \end{pmatrix} \implies y \leq 0.6$$
$$\text{with} \;\; a, b, c \in \mathbb{B}$$

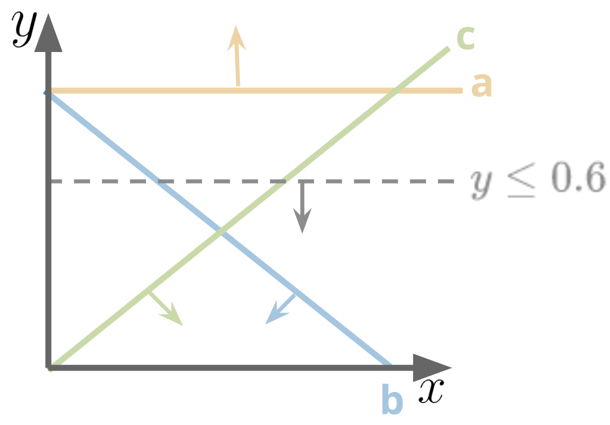


A solution is a variable assignment of $a, b, c \in \mathbb{B}$ **satisfying all constraints**

Example of OPT+qLP problem

Ensuring that a solution is valid

$$\begin{aligned}&\text{minimize} \quad a+b+c\\&\text{such that}\\&\quad (a \vee b \vee c)\\&\quad \wedge\, \forall x, y \in \mathbb{R}, \begin{pmatrix} & (y \geq 1 \vee \neg a) \\ \wedge & (x+y \leq 1 \vee \neg b) \\ \wedge & (-x+y \leq 0 \vee \neg c) \end{pmatrix} \implies y \leq 0.6\\&\text{with} \;\; a, b, c \in \mathbb{B}\end{aligned}$$



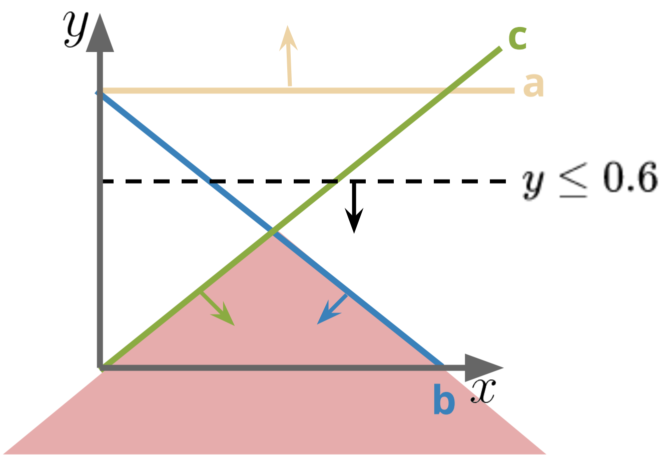
Given an assignment of $a, b, c \in \mathbb{B}$:

- Select the least number of linear constraints that should be satisfied
  - First part of the imply should be satisfied if possible
- Check that all the solution space they defined match the universal constraints

Example of OPT+qLP problem

Ensuring that a solution is valid

$$\text{minimize} \quad a+b+c$$
$$\text{such that}$$
$$(a \lor b \lor c)$$
$$\land\, \forall x, y \in \mathbb{R}, \begin{pmatrix} (y \geq 1 \lor \neg a) \\ \land \quad (x+y \leq 1 \lor \neg b) \\ \land \quad (-x+y \leq 0 \lor \neg c) \end{pmatrix} \implies y \leq 0.6$$
$$\text{with} \;\; a,b,c \in \mathbb{B}$$



Valid assignment

Given an assignment of $a, b, c \in \mathbb{B}$:

- Select the least number of linear constraints that should be satisfied
  - First part of the imply should be satisfied if possible
- Check that all the solution space they defined match the universal constraints

Example of OPT+qLP problem

Ensuring that a solution is valid

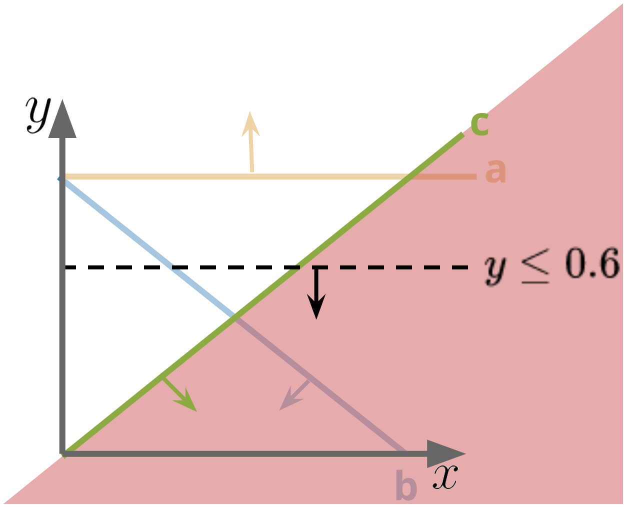
$$\text{minimize} \quad a + b + c$$

such that

$$(a \lor b \lor c)$$

$$\land \forall x, y \in \mathbb{R}, \begin{pmatrix} (y \geq 1 \lor \neg a) \\ \land \quad (x + y \leq 1 \lor \neg b) \\ \land \quad (-x + y \leq 0 \lor \neg c) \end{pmatrix} \implies y \leq 0.6$$

with $a, b, c \in \mathbb{B}$



Not a valid assignment

Given an assignment of $a, b, c \in \mathbb{B}$:

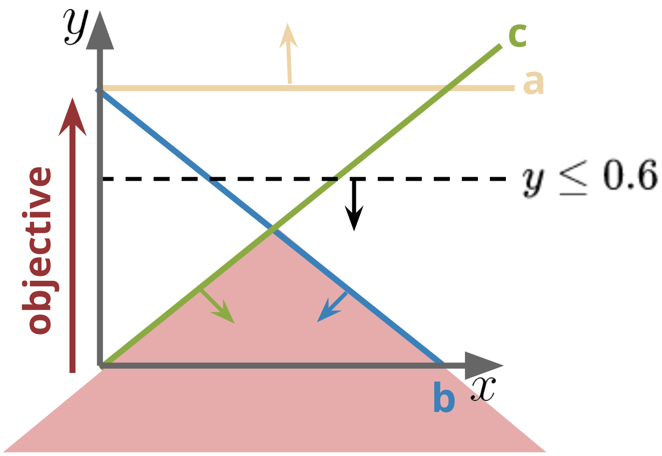
- Select the least number of linear constraints that should be satisfied
    - First part of the imply should be satisfied if possible
- Check that all the solution space they defined match the universal constraints

Link with linear programming

$$\forall v \in \mathbb{R}^n, \bigwedge_f f(v) \leq 0 \implies g(v) \leq 0 \quad \iff \quad \begin{array}{rl} \text{maximise} & g(v) \\ \text{such that:} & \forall f, f(v) \leq 0 \\ \text{with} & v \in \mathbb{R}^n \end{array}$$

Example

$$\begin{array}{l} \text{minimize} \quad a + b + c \\ \text{such that} \\ \quad (a \lor b \lor c) \\ \quad \land \forall x, y \in \mathbb{R}, \begin{pmatrix} & (y \geq 1 \lor \neg a) \\ \land & (x + y \leq 1 \lor \neg b) \\ \land & (-x + y \leq 0 \lor \neg c) \end{pmatrix} \implies y \leq 0.6 \\ \text{with} \;\; a, b, c \in \mathbb{B} \end{array}$$



$\{a : \bot, b : \top, c : \top\}$ holds if the optimum of

$$\begin{array}{rl} \text{maximize} & y \\ \text{such that} & \\ & x + y \leq 1 \\ & -x + y \leq 0 \\ \text{with} & x, y \in \mathbb{R} \end{array}$$

is less than ≤ 0.6

Solving OPT+qLP problems

In practice

Many approaches for solving **quantifier free** OPT+LP problems

For OPT+qLP problems:

1. ***Approach handling quantifier over real variables*** *– mostly SMT solvers*
   - Rely on *e-match tree algorithm*[1]
     example: z3[2]
   - Do ***not allow quantifiers over optimized variables***
   - Do ***not natively support solutions enumeration***

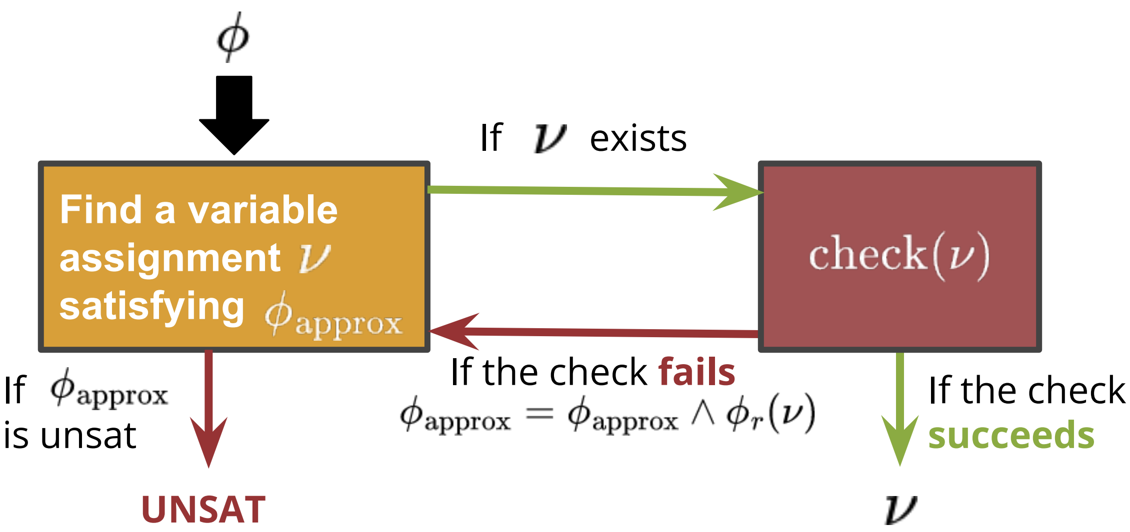
2. **Universal quantifier eliminations**
   - Remove universal quantifiers
   - Solvable on state of the art ASP+LP solvers
     example: clingo[lpx][3]

[1] L. de Moura and N. Bjørner, **Automated Deduction**, 2007
[2] L. de Moura et al., **TACAS**, 2008
[3] T. Janhunen et al., **TPLP**, 2017

Counter-Example-Guided Abstraction Refinement — CEGAR[1]

Rely on:

1. An over-approximation of the OPT+qLP problem — $\phi \implies \phi_{\text{approx}}$
2. Methods to check the validity of an assignment — $\text{check}(\nu)$
3. Refinements functions to generalize counter-examples — $\phi_r(\nu)$



- Similar to Guess-and-Check
- Not solver dependent
- Easy to implement with *clingo Propagator API*

[1] E. Clarke et al., **Journal of the ACM**, 2003

Contribution: Boolean abstraction

Replace each linear constraint by a new Boolean variable

$$\bigwedge_{c\in C} c(x) \wedge \bigwedge_{d\in D} d(x,y) \wedge \forall z \in \mathbb{R}^p, \bigwedge_{e\in E} e(x,z) \implies \bigwedge_{h\in H} h(x,z)$$

$$\phi$$

$$\rightarrow$$

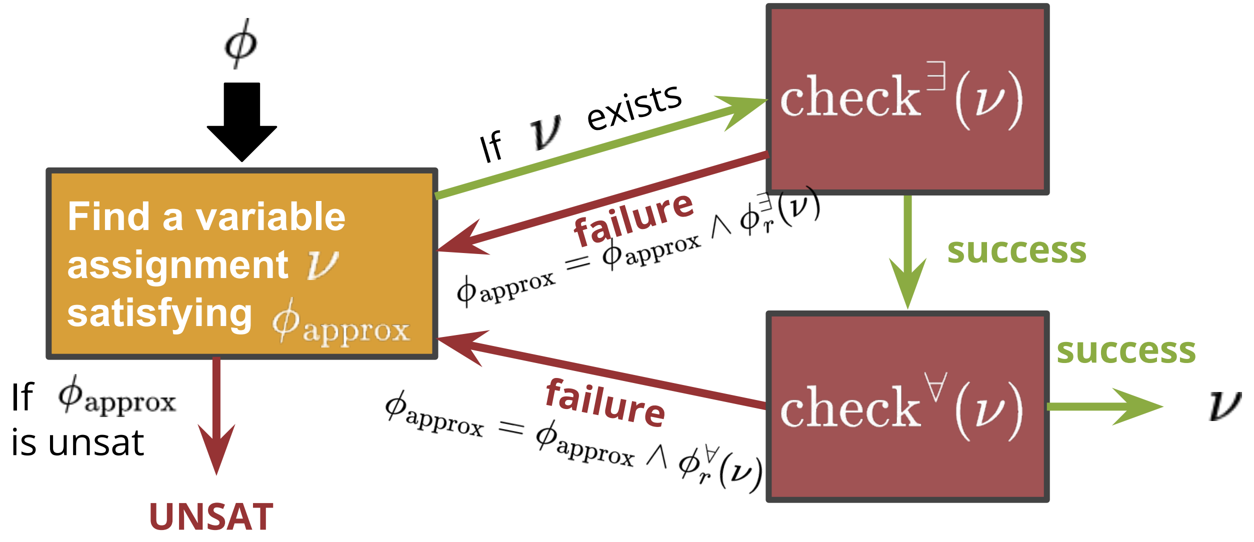
$$\bigwedge_{c\in C} c(x) \wedge \bigwedge_{d\in D} \bar{d}(x,\bar{f}_d) \wedge \bigwedge_{e\in E} \bar{e}(x,\bar{f}_e) \wedge \bigwedge_{h\in H} \bar{h}(x,\bar{f}_h)$$

$$\phi_{\text{approx}}$$

Theorem $\phi \implies \phi_{\text{approx}}$

Contribution: checks and refinement functions

Check separately existential and universal linear constraints



- ***Unsatisfiable cores*** are used to generalize unsatisfiable existential linear constraints
    - method used in most *SMT solvers* and *Clingo[lpx]*
- In practice: checks are made using dedicated LP solvers

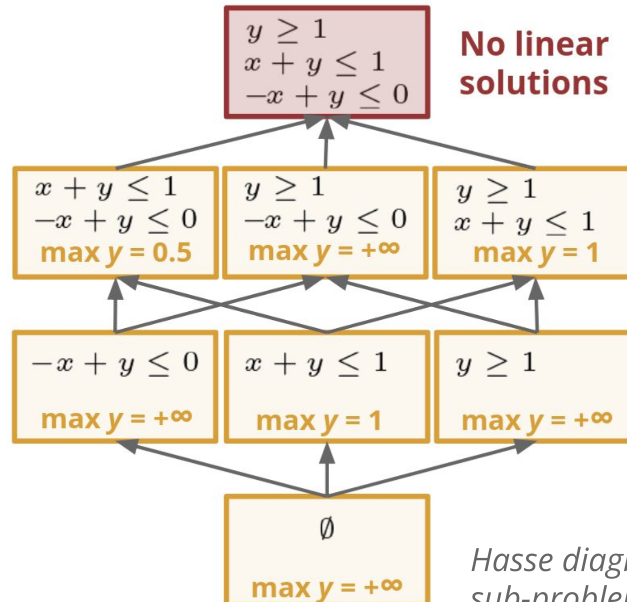
Reasoning over LP problems optimums

Property adding a constraint to a LP problem can not increase its maximum

Example

$$
\begin{aligned}
\text{maximise} \quad & y \\
\text{such that:} \quad & y \geq 1 \\
& x + y \leq 1 \\
& -x + y \leq 0 \\
\text{with} \quad & x, y \in \mathbb{R}
\end{aligned}
$$

$$\forall y \in \mathbb{R}, y \leq 0.6$$



Hasse diagram of LP sub-problems

Property If a set of constraints does not satisfy a universal constraint, then all its subset will be not valid

Reasoning over LP problems optimums

In practice

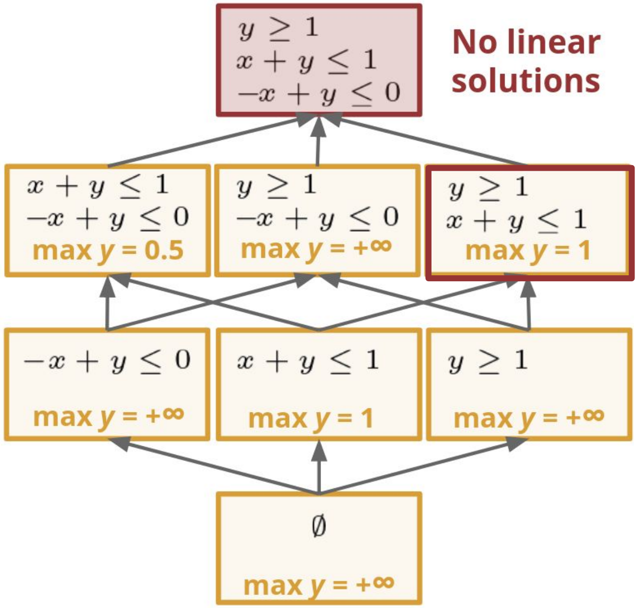
Prohibit all subsets of LP constraints of a conflicting problem

Example

$$\forall y \in \mathbb{R}, y \leq 0.6$$

Resolution:

1. $\begin{matrix} y \geq 1 \\ x + y \leq 1 \end{matrix} \rightarrow \max \quad y = 1$



Reasoning over LP problems optimums

In practice

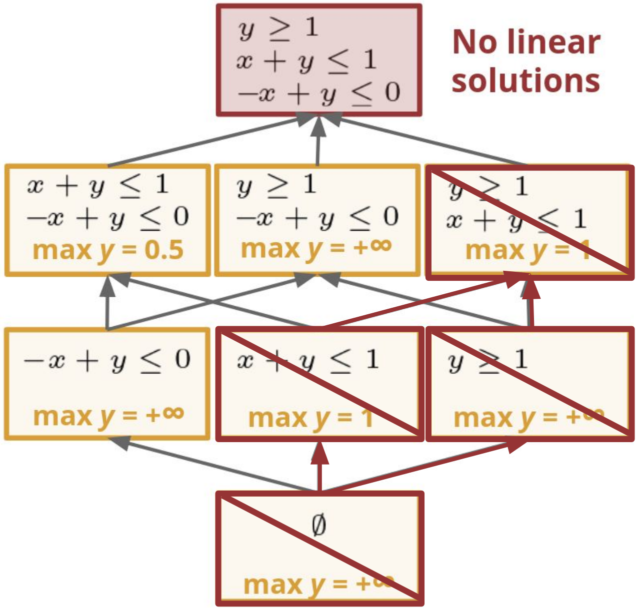
Prohibit all subsets of LP constraints of a conflicting problem

Example

$$\forall y \in \mathbb{R}, y \leq 0.6$$

Resolution:

1. $\begin{matrix} y \geq 1 \\ x + y \leq 1 \end{matrix} \rightarrow \max \quad y = 1$



Prohibited all sets with $\neg(-x + y \leq 0)$

All subset will have an optimum ≥ 1

Reasoning over LP problems optimums

In practice

Prohibit all subsets of LP constraints of a conflicting problem

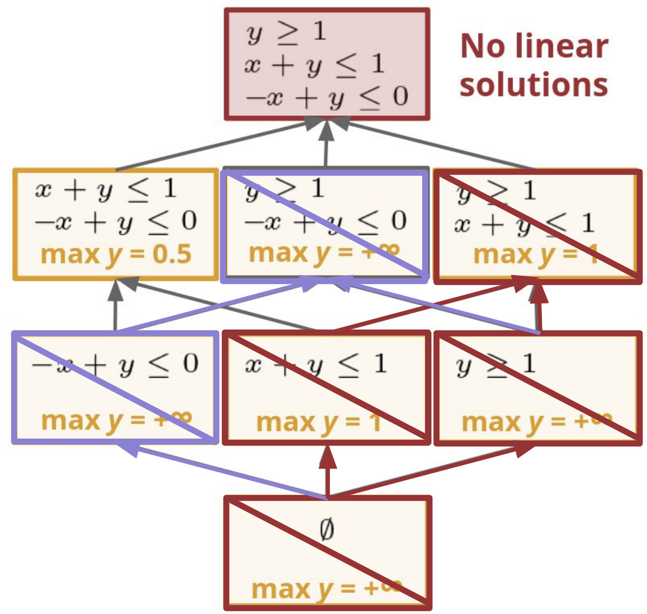
Example

$$\forall y \in \mathbb{R}, y \leq 0.6$$

Resolution:

1. $\begin{matrix} y \geq 1 \\ x + y \leq 1 \end{matrix} \rightarrow \max \quad y = 1$

2. $\begin{matrix} y \geq 1 \\ -x + y \leq 0 \end{matrix} \rightarrow \max \quad y = \infty$



Prohibited all sets with $\neg(x + y \leq 1)$

All subset will be +∞

Reasoning over LP problems optimums

In practice

Prohibit all subsets of LP constraints of a conflicting problem

Example

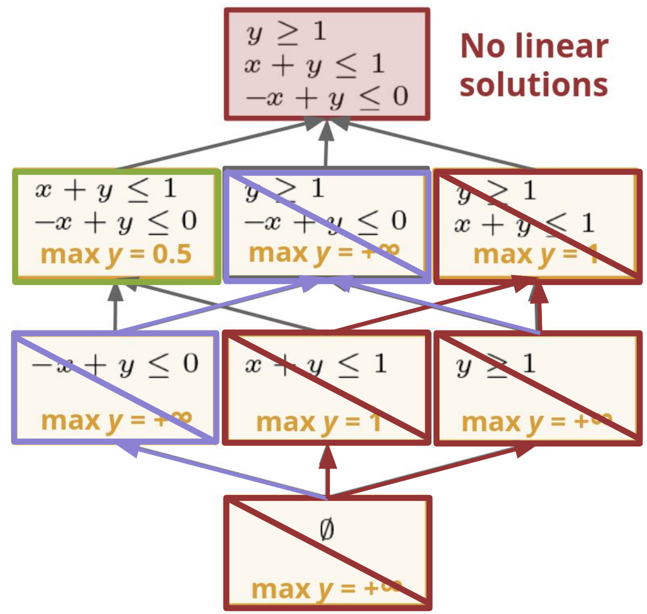
$$\forall y \in \mathbb{R}, y \leq 0.6$$

Resolution:

1. $\begin{matrix} y \geq 1 \\ x + y \leq 1 \end{matrix} \rightarrow \max \quad y = 1$

2. $\begin{matrix} y \geq 1 \\ -x + y \leq 0 \end{matrix} \rightarrow \max \quad y = \infty$

3. $\begin{matrix} x + y \leq 1 \\ -x + y \leq 0 \end{matrix} \rightarrow \max \quad y = 0.5$



A solution is found

Linear constraints partitioning

LP problems partitioning:

- Independent LP problems are build separately
  - No real variables shared among different independent LP problems
- Solved successively and independently

Advantages:

- Several small LP problems are solved rather than a big one
  - Reduces UNSAT core computation costs
- Generate more precise *refinements*
  - A refinement only account for one LP sub-problem

But… efficient only if the set of linear constraints is sparse

Benchmark

No benchmark for OPT+qLP problems

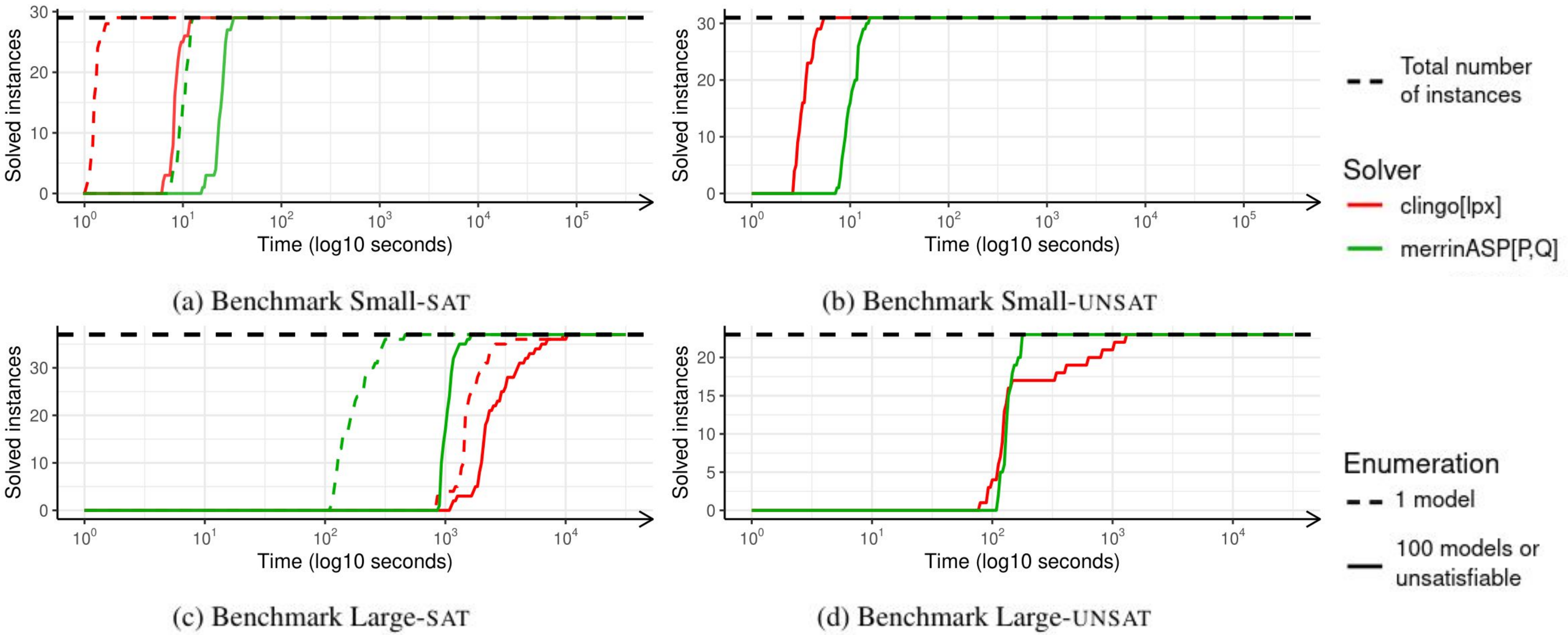
Introduce 2 benchmarks of metabolic regulatory rules inference problems

Benchmark	***Small-scale*** [1]	***Large-scale*** [2]
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

[1] K. Thuillier et al., **Oxford Bioinformatics**, 2022
[2] M. W. Covert et al, **Journal of biological chemistry**, 2002

Results

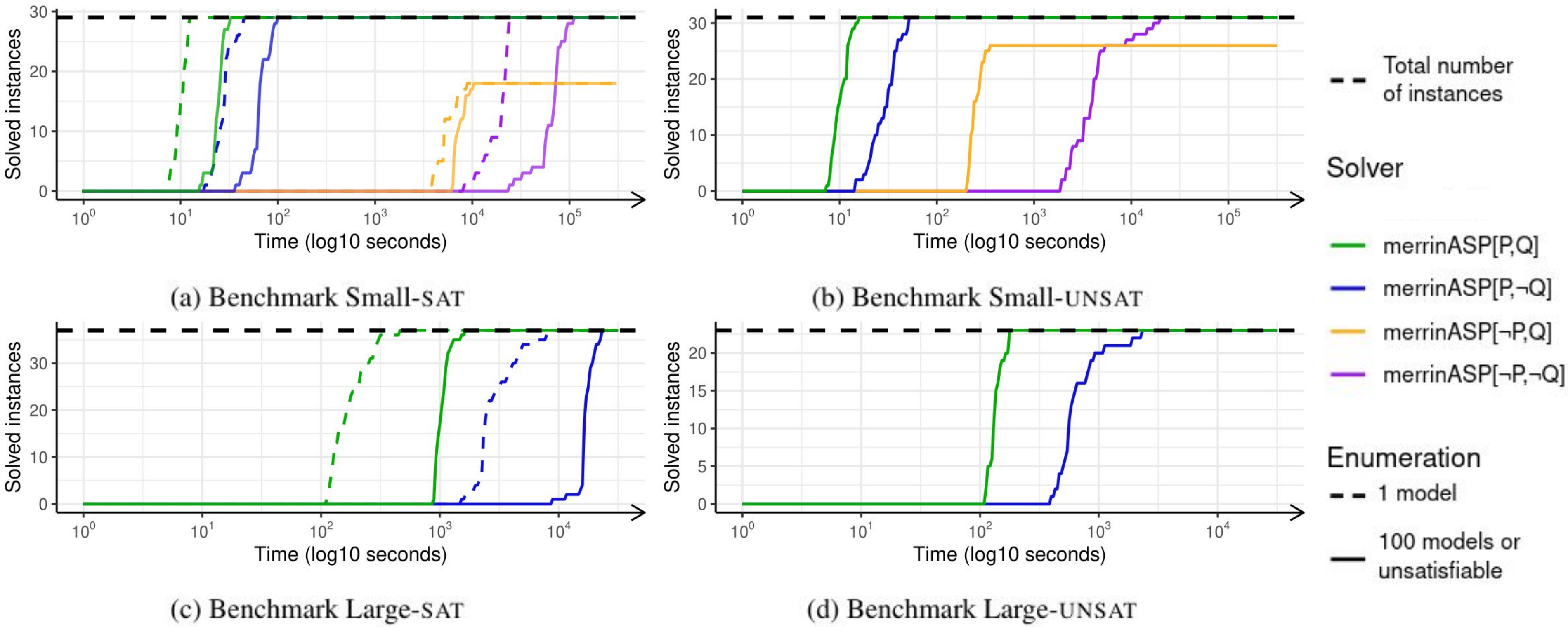
Comparison with Clingo[lpx] — with quantifier elimination



- Outperforms *clingo[lpx]* on large-scale benchmark
  - *Factor of 10*
- Equivalent computation times to enumerate solutions

Results

Impact of linear constraints partitioning [P] and universal refinement [Q]



- *Partitioning* [P]: gain of factor 1000
- *Universal refinement* [Q]: gain of factor 3 (small-scale) - 20 (large-scale)

Results

Impact on refinements and LP solver calls

Benchmark	Partitioned (P)	Quantified (Q)	LP solver Time (s)	Number of LP solvers calls	Number of refinements
Small-SAT	×	×	$3\,812 \pm 2\,727$	$16\,795 \pm 2\,364$	2 ± 0
	×	✓	$1\,433 \pm 223$	$9\,944 \pm 1\,470$	$\mathbf{1 \pm 0}$
	✓	×	34 ± 7	937 ± 111	5 ± 1
	✓	✓	$\mathbf{15 \pm 2}$	$\mathbf{501 \pm 41}$	6 ± 1
Small-UNSAT	×	×	$1\,112 \pm 766$	$6\,596 \pm 3\,723$	$\mathbf{1 \pm 0}$
	×	✓	137 ± 17	$2\,039 \pm 115$	$\mathbf{1 \pm 0}$
	✓	×	24 ± 10	669 ± 221	9 ± 4
	✓	✓	$\mathbf{7 \pm 1}$	$\mathbf{252 \pm 54}$	9 ± 4
Large-SAT	✓	×	801 ± 236	$17\,957 \pm 5\,032$	41 ± 16
	✓	✓	$\mathbf{121 \pm 74}$	$\mathbf{3\,548 \pm 2\,184}$	$\mathbf{21 \pm 11}$
Large-UNSAT	✓	×	374 ± 248	$7\,480 \pm 4\,673$	17 ± 8
	✓	✓	$\mathbf{41 \pm 11}$	$\mathbf{1\,155 \pm 307}$	$\mathbf{13 \pm 3}$

- Number of LP calls, reduced by a factor of:
  - 10 with partitioning / 7 with universal refinement
- More refinements with partitioning, but fewer calls to LP solvers

Conclusion

- A CEGAR-based approach to solve OPT+qLP problems
  - Refinement functions based on monotone properties on LP problem structures
  - Propose two benchmarks of *OPT+qLP problems*
- Implementation of a prototype by extending *clingo*
  - Benchmark our implementation against *Clingo[lpx]*
  - Significantly scale better than *Clingo[lpx] on SAT instances* (x10)
  - Partitioning and universal refinement decrease computation time by ~2000

Future works

- Does not rely on efficient algorithm to handle linear constraints, one can use DPLL-adapted simplex algorithm[1]
- Study the impact of the underlying linear solvers on performance

[1] B. Dutertre et al., **ICTACAS**, 2006