Leveraging Linear Programming for pseudo-Boolean solving

Jo Devriendt †, Jan Elffers †, Ambros Gleixner ‡, Jakob Nordström *
† KTH Royal Institute of Technology, Sweden
‡ Zuse Institut Berlin, Germany
* University of Copenhagen, Denmark
jhmde@kth.se

Pseudo-Boolean background

- Pseudo-Boolean (PB) **constraint**:
  - Bounded weighted sum of literals:

$$x + 2\overline{y} + 3z + 4\overline{w} \geq 5$$

  - Note: $\overline{x} = 1 - x$
  - Equivalent to *0-1 integer linear programming* constraint

Pseudo-Boolean background

- Pseudo-Boolean (PB) **constraint**:
  - Bounded weighted sum of literals:

$$x + 2\overline{y} + 3z + 4\overline{w} \geq 5$$

  - Note: $\overline{x} = 1 - x$
  - Equivalent to *0-1 integer linear programming* constraint
- PB formula: **conjunction** of PB constraints

Pseudo-Boolean background

- Pseudo-Boolean (PB) **constraint**:
  - Bounded weighted sum of literals:

$$x + 2\overline{y} + 3z + 4\overline{w} \geq 5$$

  - Note: $\overline{x} = 1 - x$
  - Equivalent to *0-1 integer linear programming* constraint
- PB formula: **conjunction** of PB constraints
- PB solvers decide **satisfiability** of PB formula

Pseudo-Boolean background

- Pseudo-Boolean (PB) **constraint**:
  - Bounded weighted sum of literals:

$$x + 2\overline{y} + 3z + 4\overline{w} \geq 5$$

  - Note: $\overline{x} = 1 - x$
  - Equivalent to *0-1 integer linear programming* constraint
- PB formula: **conjunction** of PB constraints
- PB solvers decide **satisfiability** of PB formula
- PB formula can be **rationally infeasible**
  - no assignment to interval [0,1] satisfies the formula, e.g.:

$$x + y + z \geq 2$$

$$\overline{x} + \overline{y} + \overline{z} \geq 2$$

Pseudo-Boolean background

- Pseudo-Boolean (PB) **constraint**:
  - Bounded weighted sum of literals:

$$x + 2\overline{y} + 3z + 4\overline{w} \geq 5$$

  - Note: $\overline{x} = 1 - x$
  - Equivalent to *0-1 integer linear programming* constraint
- PB formula: **conjunction** of PB constraints
- PB solvers decide **satisfiability** of PB formula
- PB formula can be **rationally infeasible**
  - no assignment to interval [0,1] satisfies the formula, e.g.:

$$x + y + z \geq 2$$

$$\overline{x} + \overline{y} + \overline{z} \geq 2$$

- Rationally infeasible **implies UNSAT**

SAT background

- A **clause** is a special PB constraint:

$$x + \overline{y} + z + \overline{w} \geq 1$$

SAT background

- A **clause** is a special PB constraint:

$$x + \overline{y} + z + \overline{w} \geq 1$$

- **CNF formula** can be rationally infeasible
  - But only if unit propagation leads to conflict

SAT background

- A **clause** is a special PB constraint:

$$x + \overline{y} + z + \overline{w} \geq 1$$

- **CNF formula** can be rationally infeasible
  - But only if unit propagation leads to conflict
- Otherwise, **trivial rational solution**: assign 0.5 to all non-propagated variables
  - All clauses not satisfied by unit propagation have at least 2 unassigned literals
  - Such clauses are satisfied by 0.5-assignment

SAT background

- A **clause** is a special PB constraint:

$$x + \overline{y} + z + \overline{w} \geq 1$$

- **CNF formula** can be rationally infeasible
  - But only if unit propagation leads to conflict
- Otherwise, **trivial rational solution**: assign 0.5 to all non-propagated variables
  - All clauses not satisfied by unit propagation have at least 2 unassigned literals
  - Such clauses are satisfied by 0.5-assignment

For CNF, deciding rational infeasibility is trivial

Pseudo-Boolean background

- Deciding rational infeasibility of PB formulas is **easy in theory**:
  - Algorithmic complexity class: **P** [K1979]
  - Proof-theoretic complexity: short *cutting plane* [CCT87] proofs exist [F1902]

Pseudo-Boolean background

- Deciding rational infeasibility of PB formulas is **easy in theory**:
  - Algorithmic complexity class: **P** [K1979]
  - Proof-theoretic complexity: short *cutting plane* [CCT87] proofs exist [F1902]
- **Linear programming** (LP) solvers efficiently decide rational feasibility

Pseudo-Boolean background

- Deciding rational infeasibility of PB formulas is **easy in theory**:
  - Algorithmic complexity class: **P** [K1979]
  - Proof-theoretic complexity: short *cutting plane* [CCT87] proofs exist [F1902]
- **Linear programming** (LP) solvers efficiently decide rational feasibility
- In practice, many **PB solvers struggle** on rationally infeasible formulas [EGNV18]
  - Even PB solvers that natively build cutting plane proofs, e.g., *RoundingSat* and *Sat4J*

Pseudo-Boolean background

- Deciding rational infeasibility of PB formulas is **easy in theory**:
  - Algorithmic complexity class: **P** [K1979]
  - Proof-theoretic complexity: short *cutting plane* [CCT87] proofs exist [F1902]
- **Linear programming** (LP) solvers efficiently decide rational feasibility
- In practice, many **PB solvers struggle** on rationally infeasible formulas [EGNV18]
  - Even PB solvers that natively build cutting plane proofs, e.g., *RoundingSat* and *Sat4J*

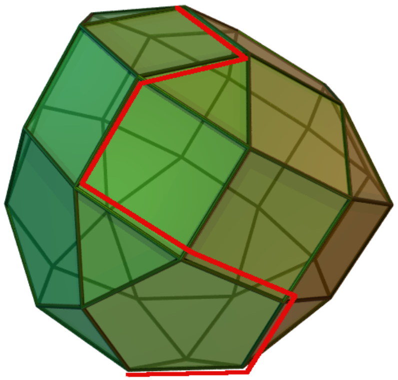
Goal of our work:

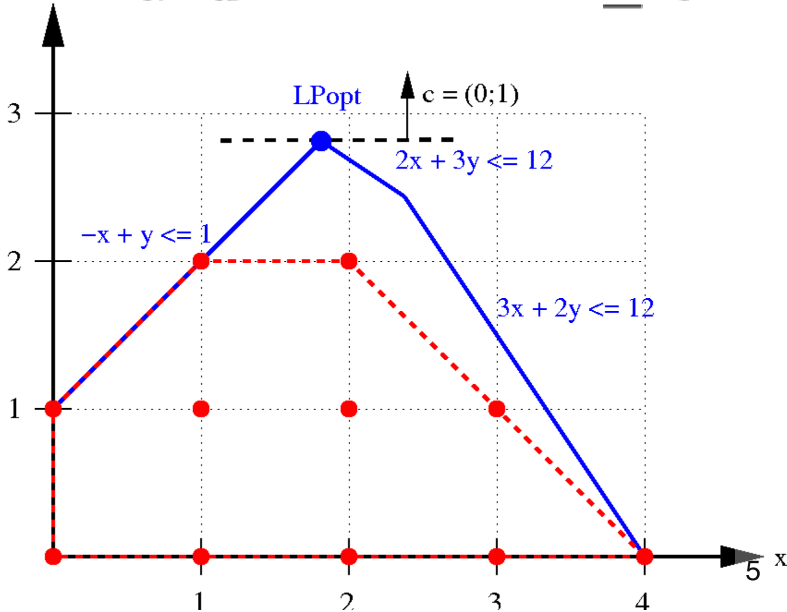
use LP solver to check rational feasibility during PB search

Linear Programming (LP) solver

- Input:
  - conjunction of linear constraints
  - variable bounds
  - objective function

$$
\begin{array}{ll}
\text{minimize} & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\
\text{subject to} & A\mathbf{x} \leq \mathbf{b} \\
\text{and} & \mathbf{x} \geq \mathbf{0}
\end{array}
$$

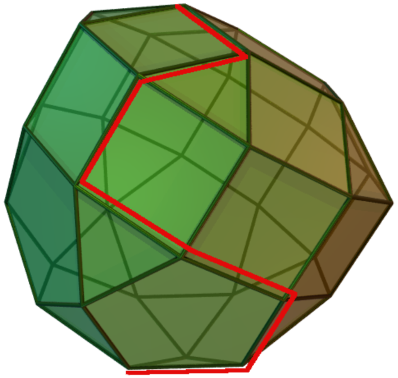


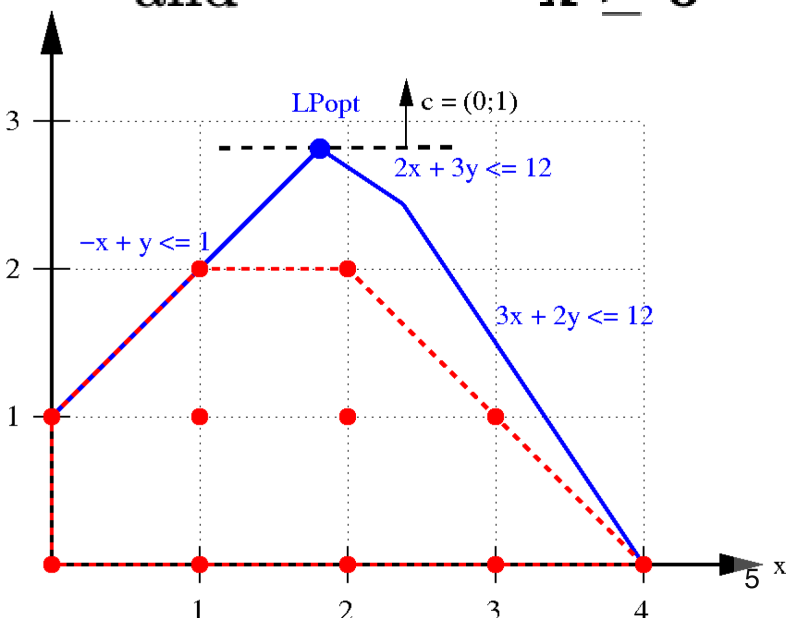


Linear Programming (LP) solver

- Input:
  - conjunction of linear constraints
  - variable bounds
  - objective function
- Output: either
  - SAT: optimal rational solution
  - UNSAT: **Farkas multipliers**
    - defines violated positive linear combination of input constraints

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ \text{and} & \mathbf{x} \geq \mathbf{0} \end{array}$$

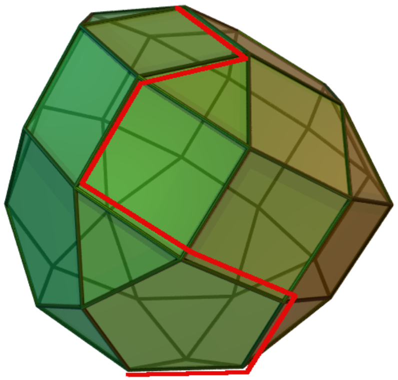


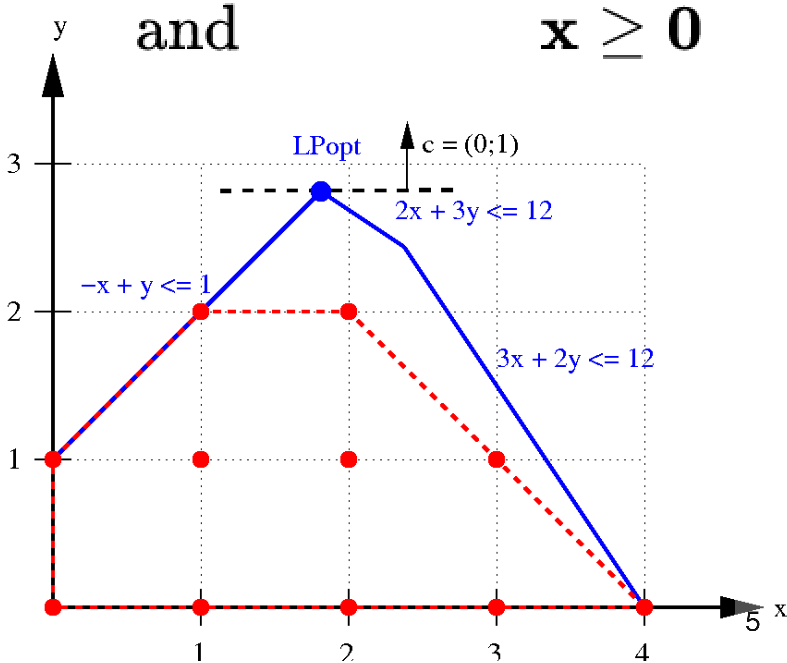


Linear Programming (LP) solver

- Input:
  - conjunction of linear constraints
  - variable bounds
  - ~~objective function~~
- Output: either
  - ~~SAT: optimal rational solution~~
  - UNSAT: **Farkas multipliers**
    - defines violated positive linear combination of input constraints

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ \text{and} & \mathbf{x} \geq \mathbf{0} \end{array}$$





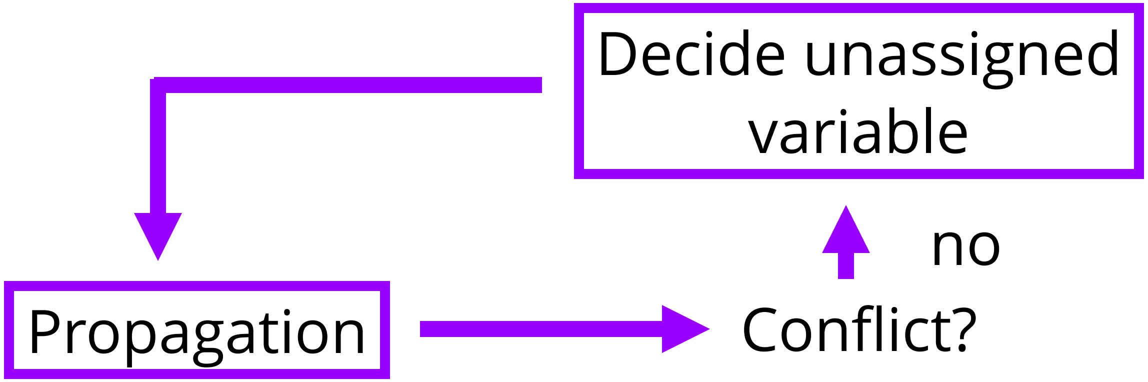
PB search loop

Propagation

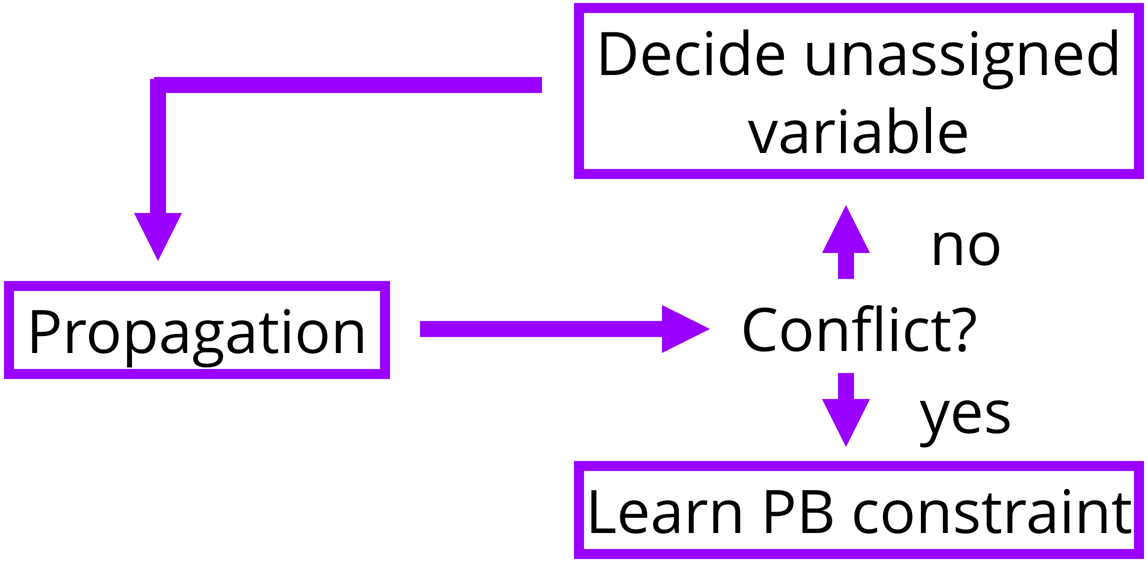
PB search loop

Propagation → Conflict?

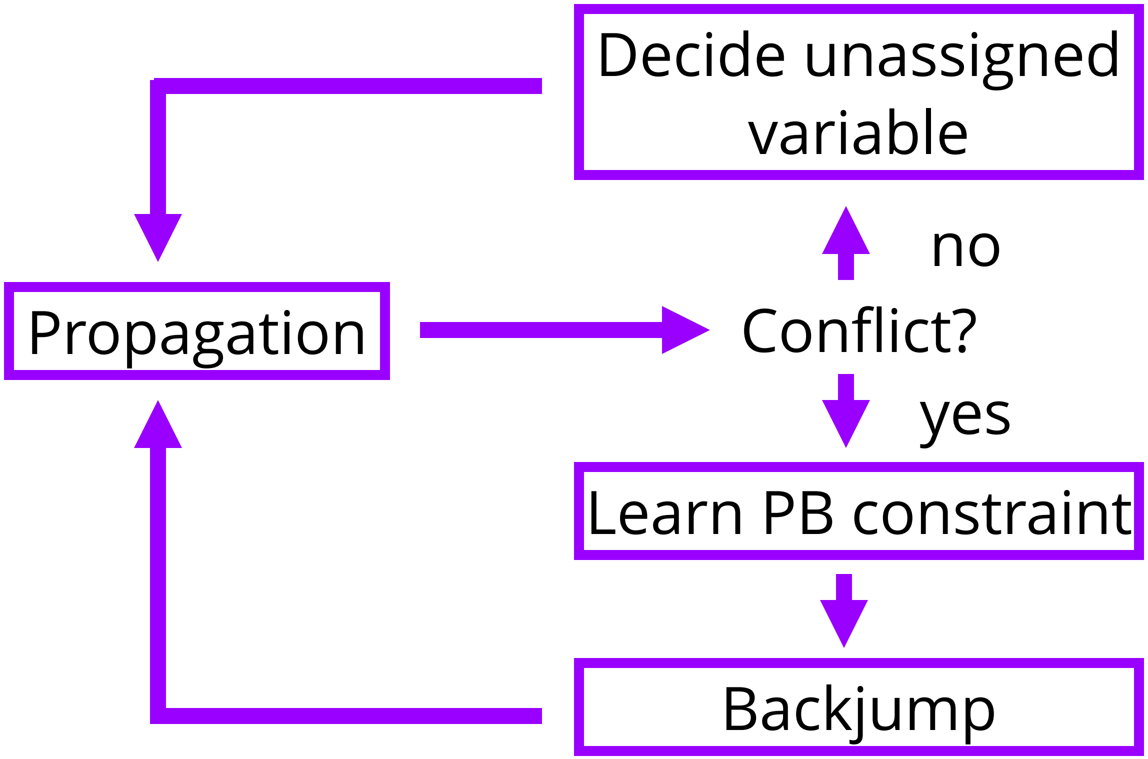
PB search loop



PB search loop

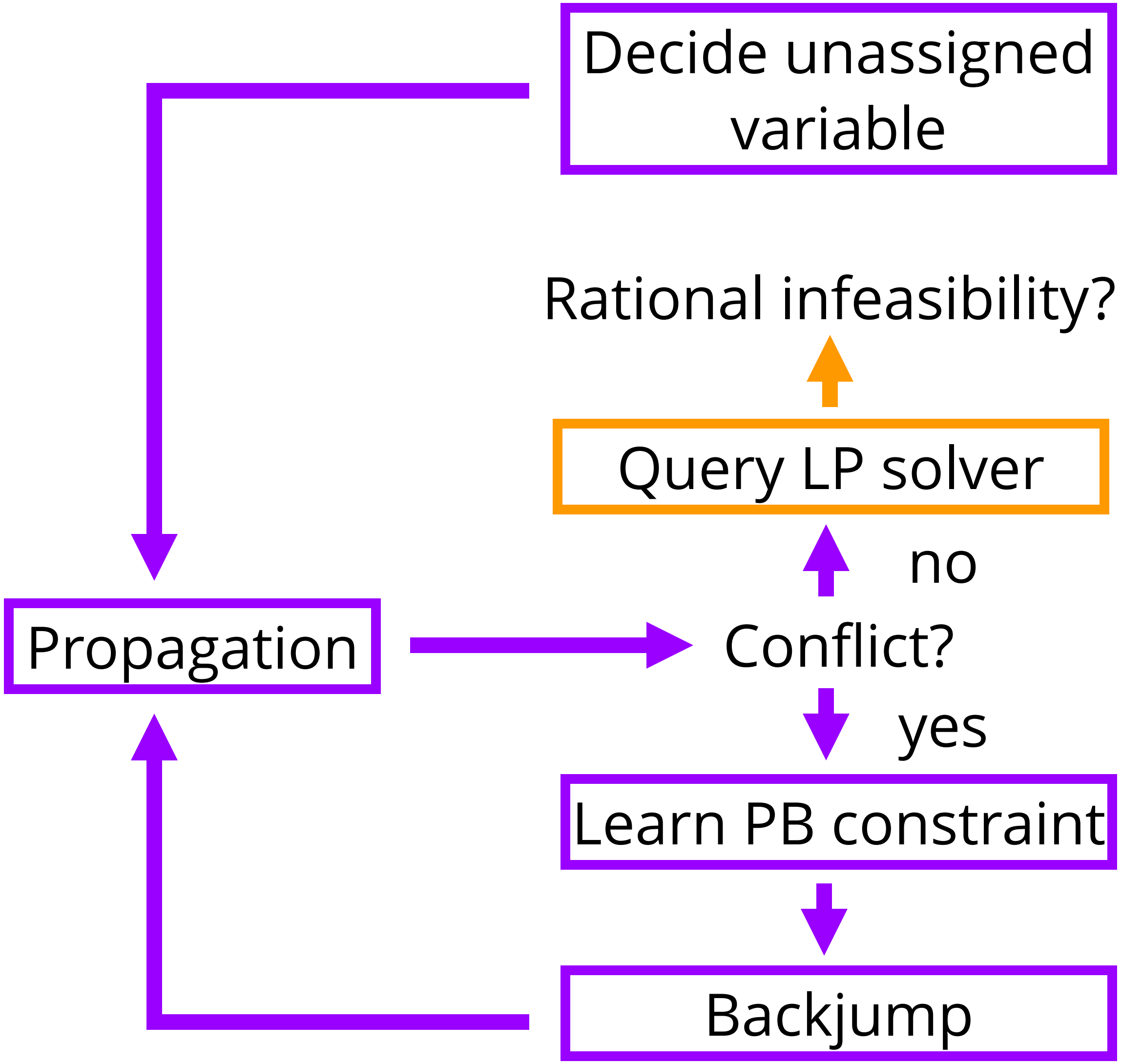


PB search loop



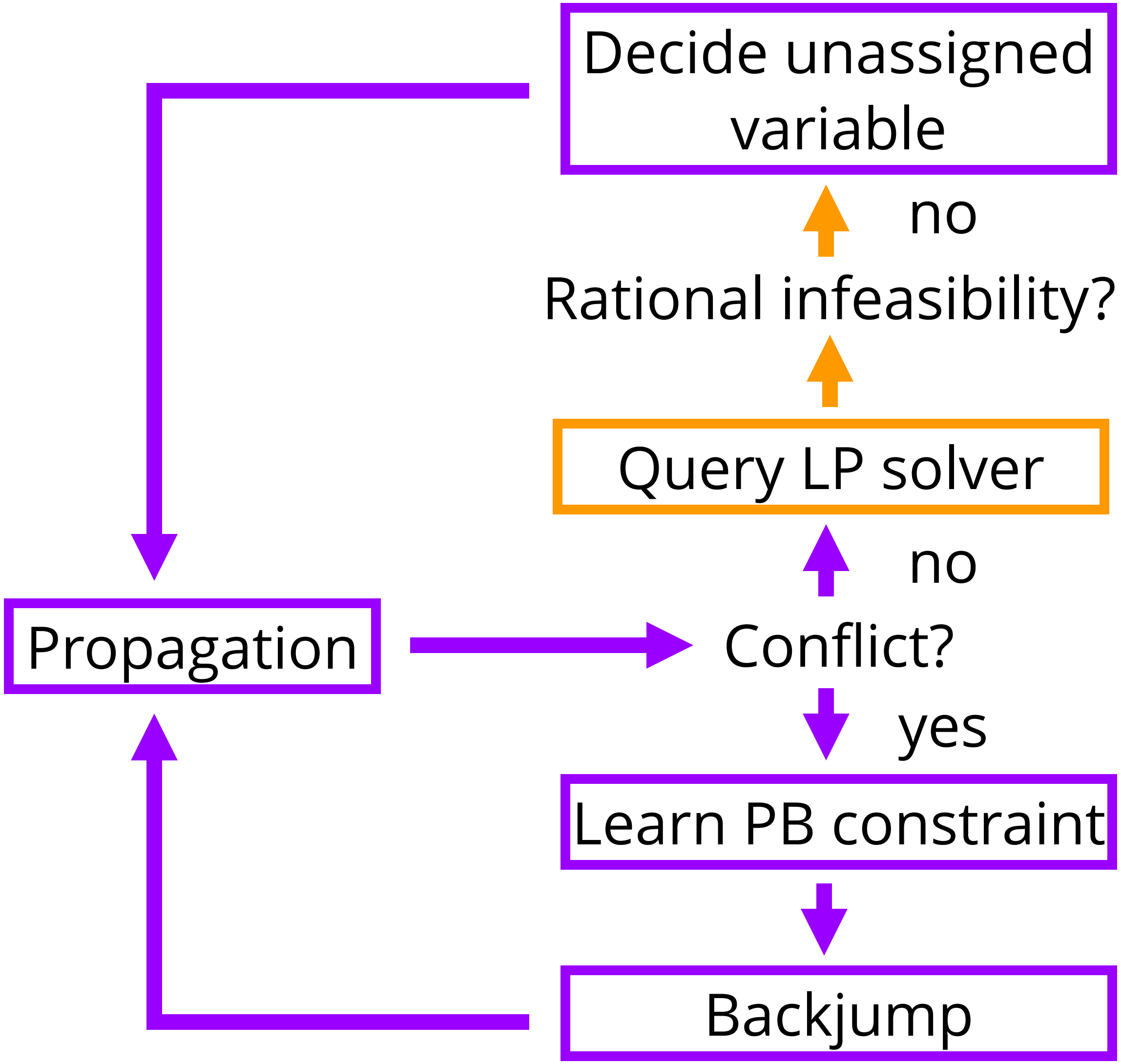
PB search loop

with LP solver call



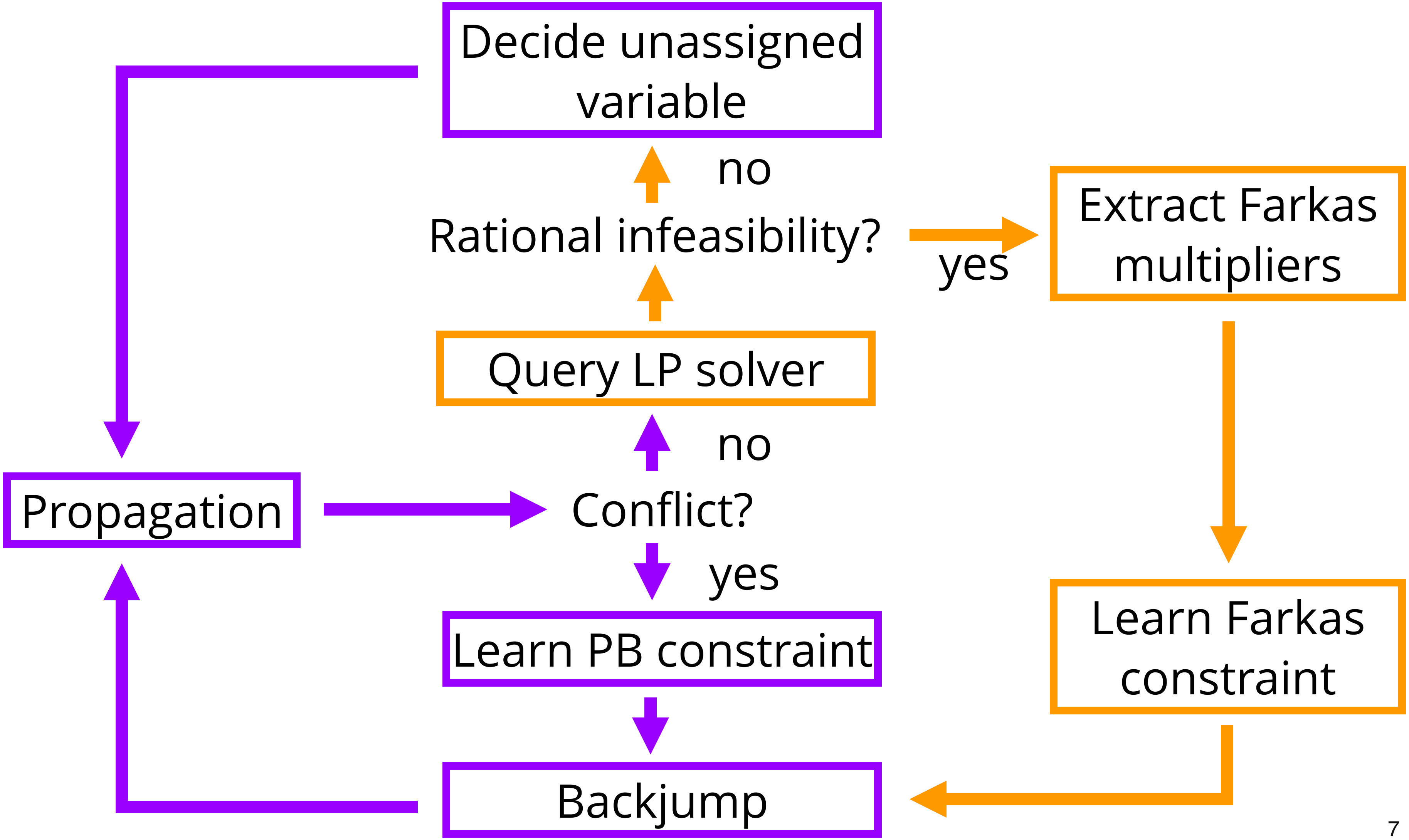
PB search loop

with LP solver call



PB search loop

with LP solver call



Two technical hurdles

- LP solvers are **slow** compared to PB search loop
  - Limit calls to LP solver
  - Limit LP solver running time
  - Deterministic measure: compare **#conflicts** in PB solver to **#pivots** in LP solver

Two technical hurdles

- LP solvers are **slow** compared to PB search loop
  - Limit calls to LP solver
  - Limit LP solver running time
  - Deterministic measure: compare **#conflicts** in PB solver to **#pivots** in LP solver
- Learned constraint must be implied by input formula
  - LP solver uses inexact floating point arithmetic
  - **Recalculate** Farkas constraint with exact arithmetic
  - **Verify** Farkas constraint is still conflicting

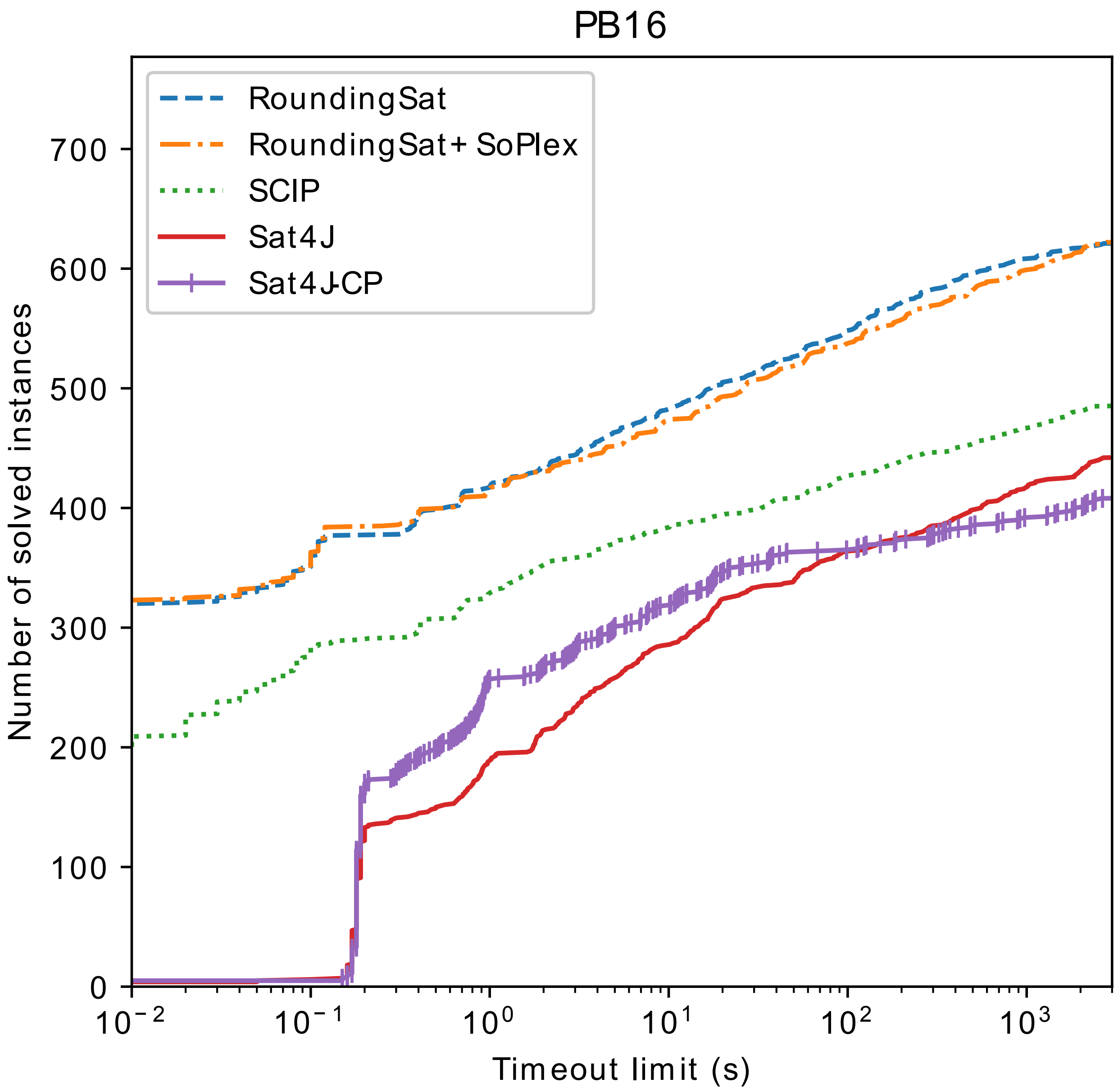
Working implementation

- PB solver **RoundingSat** [EN18]
  - Native cutting plane proofs
  - Performed well in past PB competitions
- LP solver **SoPlex** [ZIB]
  - SCIP's native LP solver
  - Fast
  - Open source

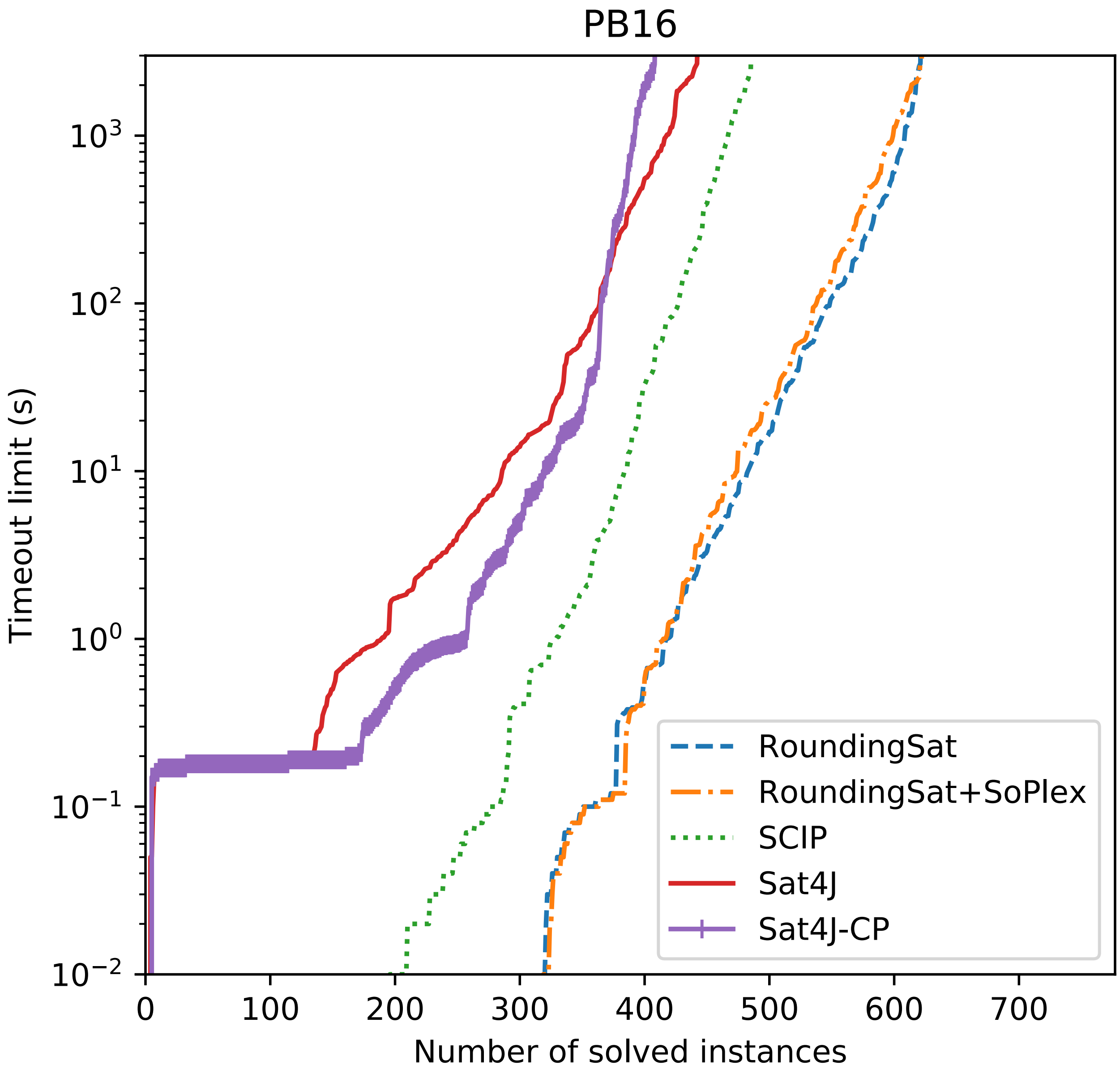
Experiments!

- 5 solver configurations
  - RoundingSat
  - **RoundingSat+SoPlex**
  - SCIP
  - Sat4J
  - Sat4J-CP
- 3000s on 16GiB machines
- 4 benchmark families:
  - PB12
  - PB16
  - MIPLIB
  - PROOF

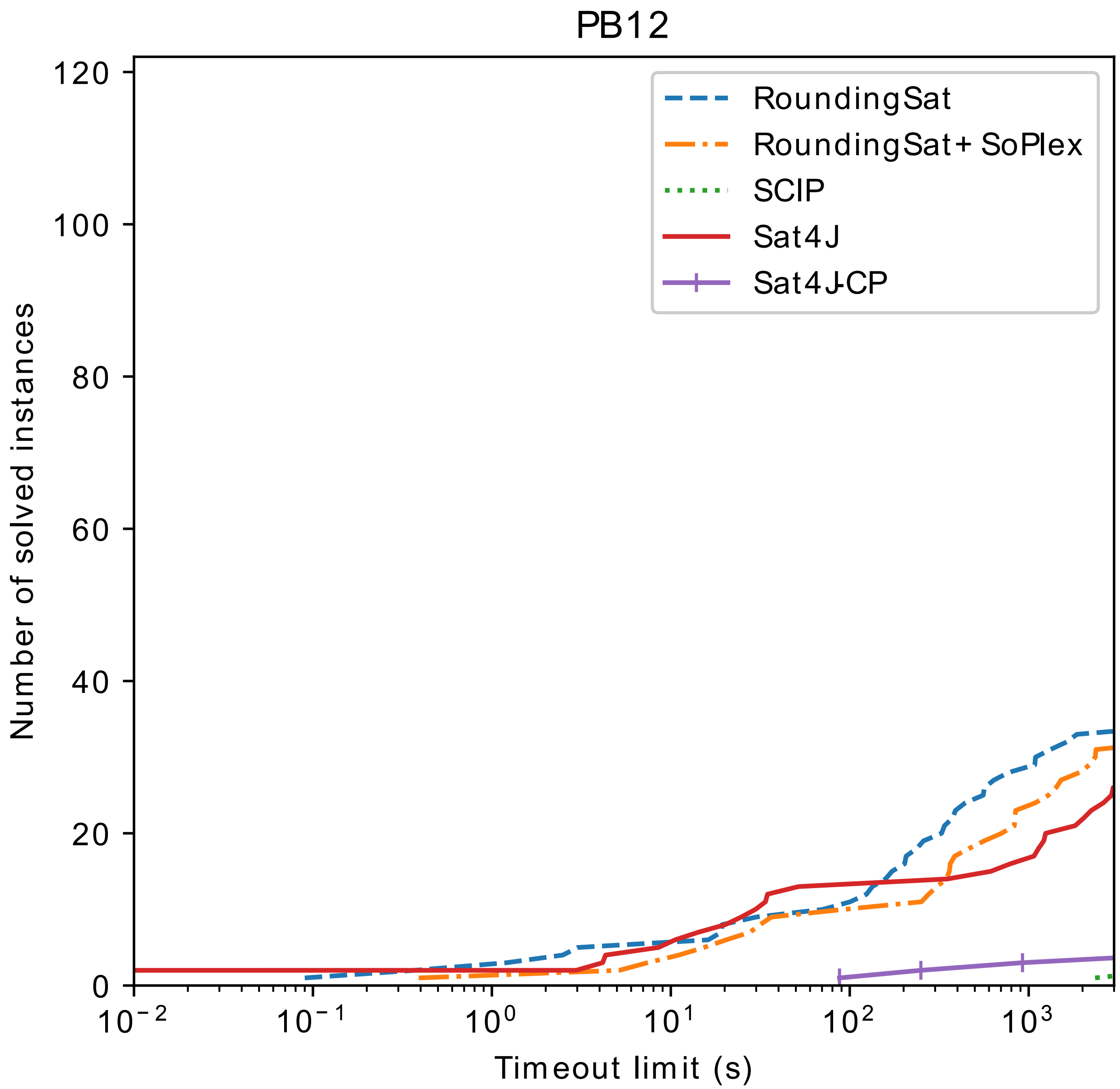
Performance experiment



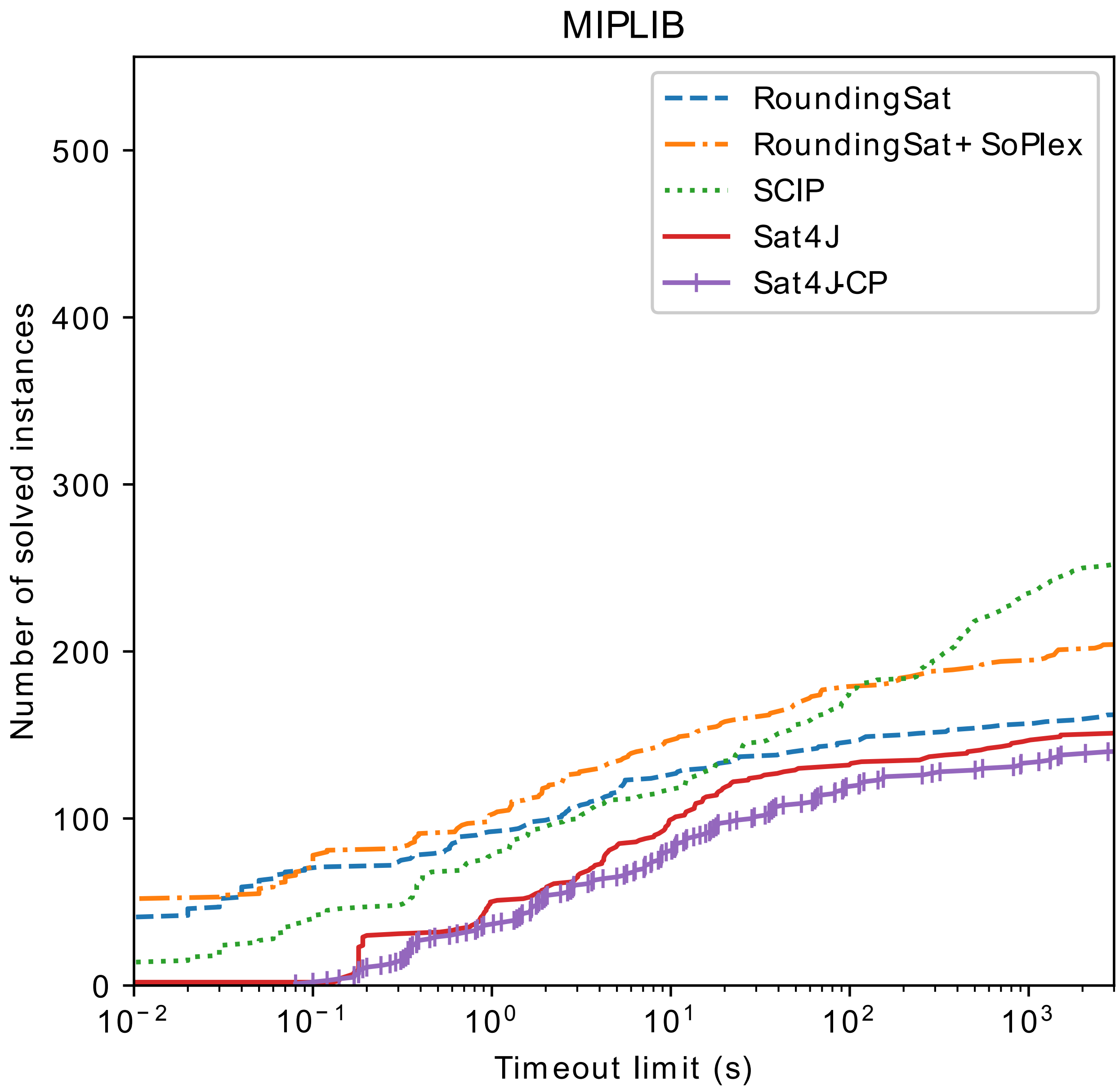
Performance experiment



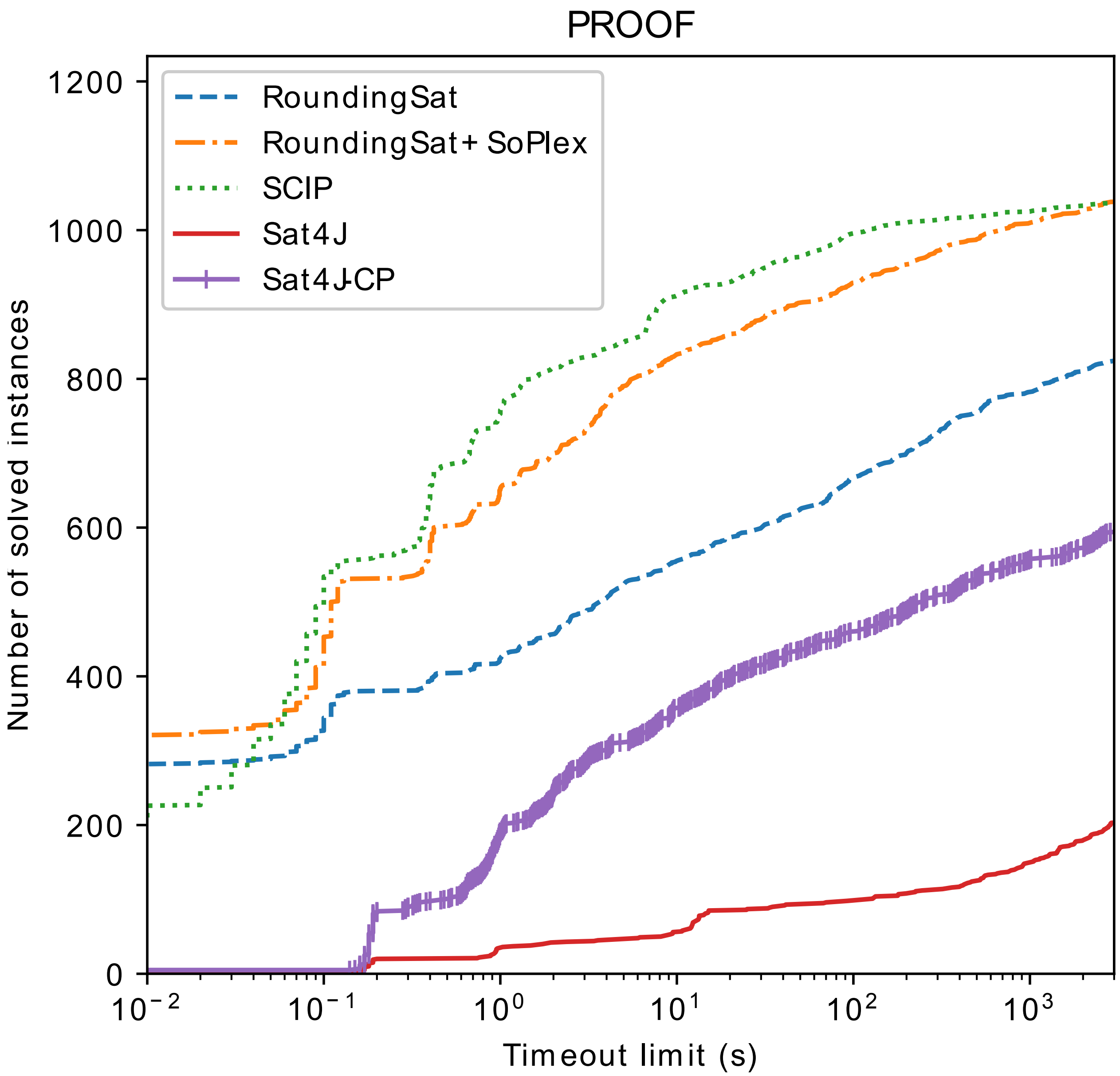
Performance experiment



Performance experiment



Performance experiment



Performance experiment

- RoundingSat+SoPlex **never really worse than RoundingSat**
  - small LP overhead at worst, huge speedups at best

Performance experiment

- RoundingSat+SoPlex **never really worse than RoundingSat**
  - small LP overhead at worst, huge speedups at best
  - Not only more solved UNSAT instances (+16%), but also more solved SAT instances (+14%)

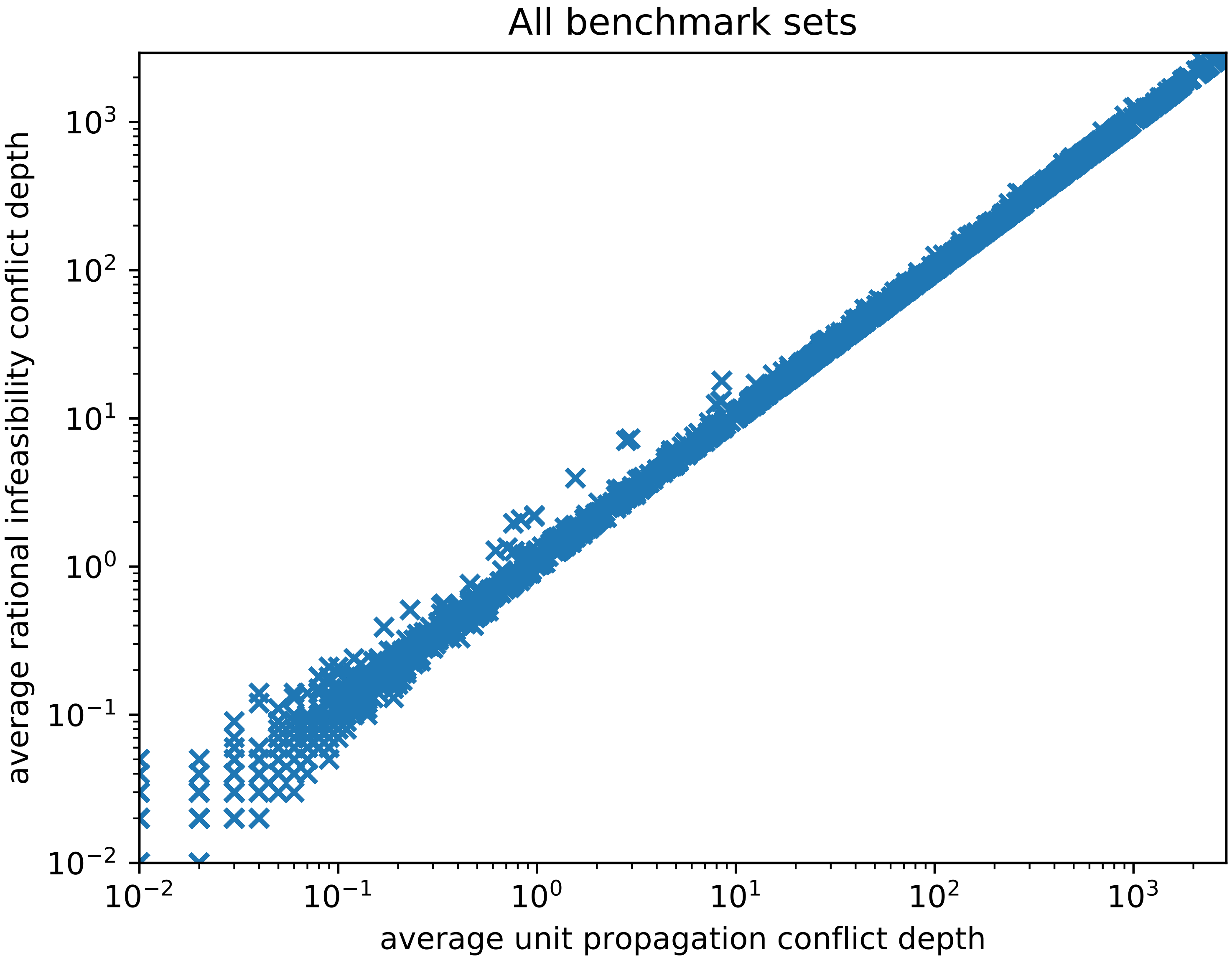
Performance experiment

- RoundingSat+SoPlex **never really worse than RoundingSat**
  - small LP overhead at worst, huge speedups at best
  - Not only more solved UNSAT instances (+16%), but also more solved SAT instances (+14%)
- RoundingSat+SoPlex and **SCIP trade places**

Performance experiment

- RoundingSat+SoPlex **never really worse than RoundingSat**
  - small LP overhead at worst, huge speedups at best
  - Not only more solved UNSAT instances (+16%), but also more solved SAT instances (+14%)
- RoundingSat+SoPlex and **SCIP trade places**
- SoPlex does not like PB12

Conflict depth experiment



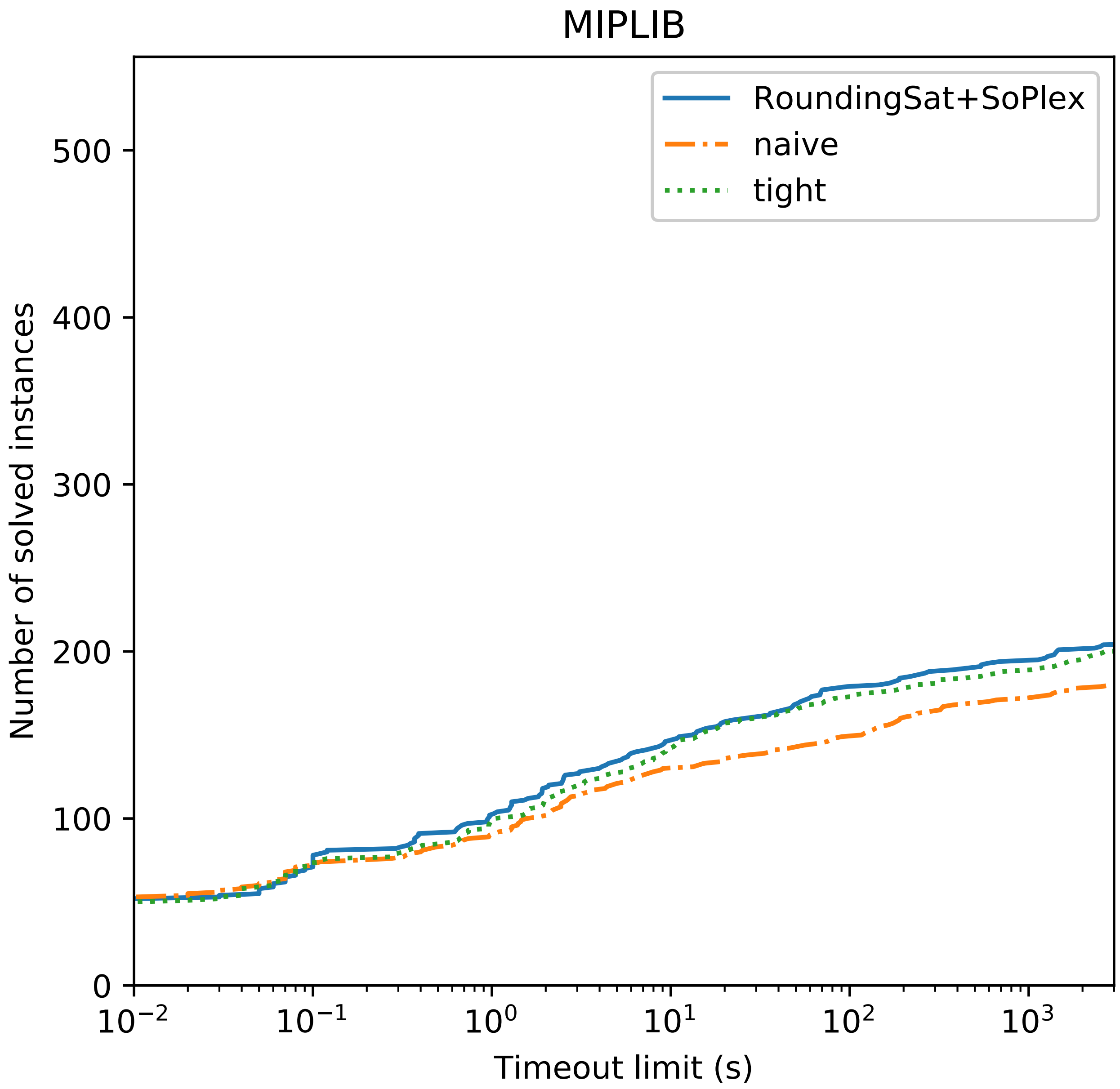
Conflict depth experiment

- Conflict depth for rational infeasibility check and unit propagation are **similar**

Conflict depth experiment

- Conflict depth for rational infeasibility check and unit propagation are **similar**
- Technique detects rational infeasibility also in **deep search nodes**

Learned constraint addition experiment



Learned constraint addition experiment

- Adding learned constraints to LP solver does **not** lead to **more solved** instances
  - Also no improvement in number of conflicts needed

Learned constraint addition experiment

- Adding learned constraints to LP solver does **not** lead to **more solved** instances
  - Also no improvement in number of conflicts needed
- Hypothesis 1: no objective function to guide "tight" variant

Learned constraint addition experiment

- Adding learned constraints to LP solver does **not** lead to **more solved** instances
  - Also no improvement in number of conflicts needed
- Hypothesis 1: no objective function to guide "tight" variant
- Hypothesis 2: rational solution at deep search nodes is not useful

Learned constraint addition experiment

- Adding learned constraints to LP solver does **not** lead to **more solved** instances
  - Also no improvement in number of conflicts needed
- Hypothesis 1: no objective function to guide "tight" variant
- Hypothesis 2: rational solution at deep search nodes is not useful
- Other hypotheses?

Conclusion

- Use LP solver to tackle rational infeasibility during search
- Implemented sound integration of LP solver in PB solver
- Experiments indicate small LP overhead at worst, huge speedups at best

Conclusion

- Use LP solver to tackle rational infeasibility during search
- Implemented sound integration of LP solver in PB solver
- Experiments indicate small LP overhead at worst, huge speedups at best

~~Future~~ Current work

- Optimization
- LP *cut* generation

Conclusion

- Use LP solver to tackle rational infeasibility during search
- Implemented sound integration of LP solver in PB solver
- Experiments indicate small LP overhead at worst, huge speedups at best

~~Future~~ Current work

- Optimization
- LP *cut* generation

Thanks for your attention!

Questions?

References

[K1979] A polynomial algorithm for linear programming - 1979 - Khachiyan

[CCT87] On the Complexity of Cutting-Plane Proofs - Cook, Coullard, Turán

[F1902] Über die Theorie der Einfachen Ungleichungen - 1902 - Farkas

[EGNV18] Using combinatorial benchmarks to probe the reasoning power of pseudo-Boolean solvers - 2018 - Elffers, Giráldez-Cru, Nordström, Vinyals

[EN18] Divide and conquer: Towards faster pseudo-boolean solving - 2018 - Elffers, Nordström

[ZIB] SoPlex - soplex.zib.de