Approximately Propagation Complete and Conflict Propagating Constraint Encodings

Rüdiger Ehlers Francisco Palau Romero

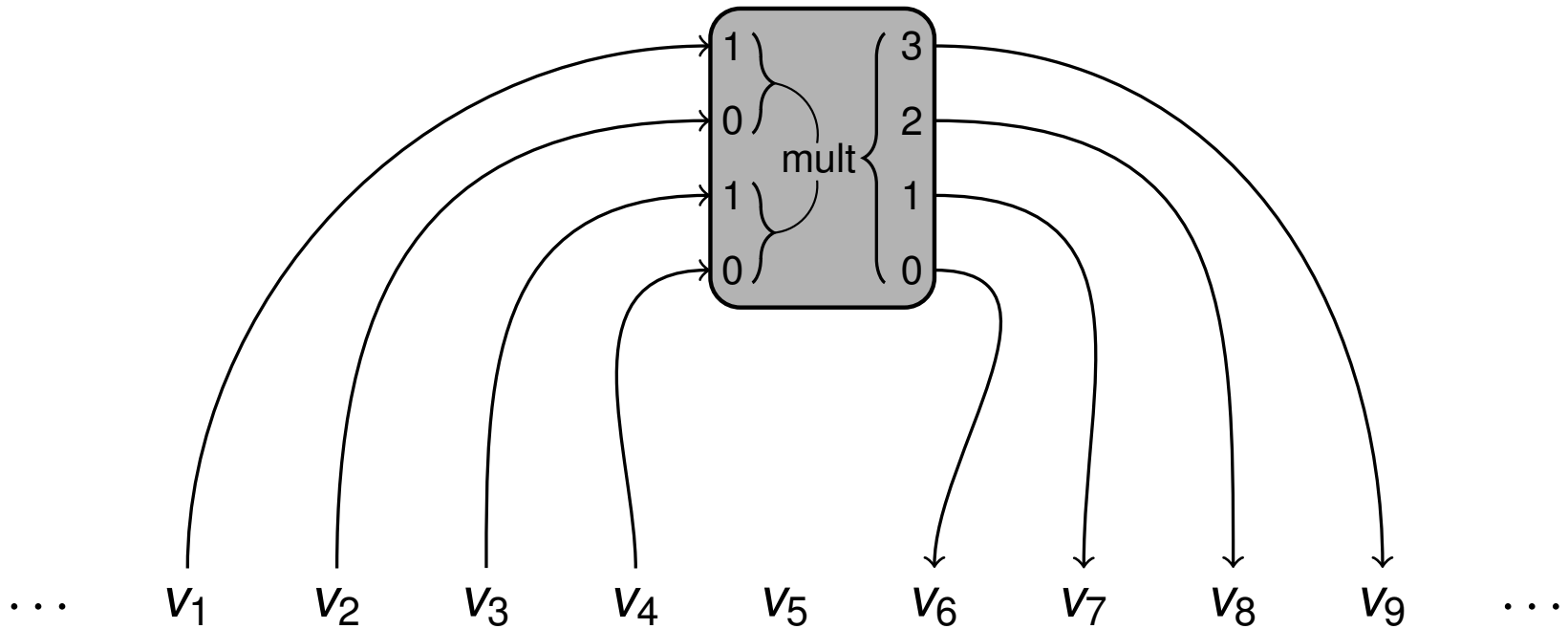
University of Bremen

SAT 2018, Oxford, UK

SAT Constraints

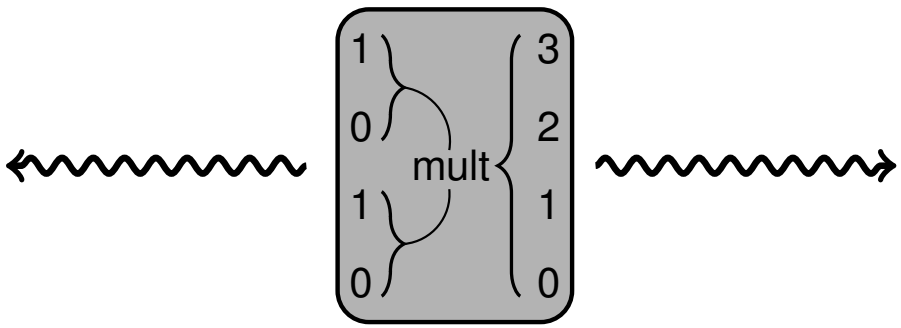
$$\cdots \quad v_1 \quad v_2 \quad v_3 \quad v_4 \quad v_5 \quad v_6 \quad v_7 \quad v_8 \quad v_9 \quad \cdots$$

SAT Constraints



Good Encodings

```
1 -5 0
3 -5 0
2 -8 0
4 -8 0
5 -8 0
-6 -8 0
4 -7 0
-5 -7 0
2 -7 0
-3 6 -7 0
3 4 -6 0
2 4 -6 0
-1 -3 5 0
1 3 -6 0
1 2 -6 0
-1 6 -7 0
-2 -3 6 8 0
-1 -4 6 8 0
-2 -4 7 8 0
```




```
1 -5 0
3 -5 0
2 -8 0
4 -8 0
-7 -8 0
4 -7 0
2 -7 0
2 4 -6 0
-1 -3 5 0
1 3 -6 0
-2 -3 6 8 0
-1 -4 6 8 0
-2 -4 5 7 0
-2 3 -6 7 0
-3 -4 5 -6 7 0
-1 -2 -3 -4 -6 0
```

Common properties of good encodings

- Few clauses
- Makes best use of unit propagation

Propagation Completeness

Propagation completeness

A CNF encoding ψ over a set of variables $\mathcal{V}$ is propagation complete iff for every partial valuation $p : \mathcal{V} \rightharpoonup \{\textbf{false}, \textbf{true}\}$ that implies a literal value l, we have that l can be derived from p by unit propagation.

Propagation Completeness

Propagation completeness

A CNF encoding ψ over a set of variables $\mathcal{V}$ is propagation complete iff for every partial valuation $p : \mathcal{V} \rightharpoonup \{\textbf{false}, \textbf{true}\}$ that implies a literal value l, we have that l can be derived from p by unit propagation.

Example

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge (x_4 \vee x_2 \vee x_3)$$

Propagation Completeness

Propagation completeness

A CNF encoding ψ over a set of variables $\mathcal{V}$ is propagation complete iff for every partial valuation $p : \mathcal{V} \rightharpoonup \{\textbf{false}, \textbf{true}\}$ that implies a literal value l, we have that l can be derived from p by unit propagation.

Example

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge (x_4 \vee x_2 \vee x_3)$$

Partial valuation $p = \{x_4 \mapsto \textbf{false}\}$ implies $\neg x_1$, but this is not detected by unit propagation.

Propagation Completeness

Propagation completeness

A CNF encoding ψ over a set of variables $\mathcal{V}$ is propagation complete iff for every partial valuation $p : \mathcal{V} \rightharpoonup \{\textbf{false}, \textbf{true}\}$ that implies a literal value l, we have that l can be derived from p by unit propagation.

Example

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge (x_4 \vee x_2)$$

Propagation Completeness

Propagation completeness

A CNF encoding ψ over a set of variables $\mathcal{V}$ is propagation complete iff for every partial valuation $p : \mathcal{V} \rightharpoonup \{\textbf{false}, \textbf{true}\}$ that implies a literal value l, we have that l can be derived from p by unit propagation.

Example

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge (x_4 \vee x_2)$$

Partial valuation $p = \{x_4 \mapsto \textbf{false}\}$ implies x_2 by UP.

Propagation Completeness

Propagation completeness

A CNF encoding ψ over a set of variables $\mathcal{V}$ is propagation complete iff for every partial valuation $p : \mathcal{V} \rightharpoonup \{\textbf{false}, \textbf{true}\}$ that implies a literal value l, we have that l can be derived from p by unit propagation.

Example

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge (x_4 \vee x_2)$$

Partial valuation $p = \{x_4 \mapsto \textbf{false}\}$ implies x_2, $\neg x_1$ and $\neg x_3$ by unit propagation.

Generating Propagation Complete CNF Encodings

Bordeaux and Marques-Silva, 2012

- Showed that we can make a CNF encoding propagation complete by **adding empowering clauses**.

Brain et al., 2016

- Introduced abstract satisfaction as a theoretical foundation to reason about propagation strength of encodings.
- Give a procedure to compute *minimal* propagation complete CNF encodings **from scratch**.

Propagation-Complete vs. Smallest Encodings

Problem

Minimally-sized propagation complete CNF encodings can be much larger than (minimally-sized) arbitrary encodings
→ **more book-keeping effort for the solver**

Some examples

Constraint	**P.C. CNF**	**arbitrary CNF**
2×3 bit multiplier (5 output bits)	304	45
3-object all-different constraint	24	12
8 bit square root function	209	52
5 bit square function	273	42
5 bit adder	1080	78

Research Question

Is there a way to **balance**
the **size** of CNF encodings
of constraints against their
propagation strength?

Contributions of This Paper

New concepts

- **Approximate** Propagation Completeness
- **Approximate** Conflict Propagation

Technical results

- An efficient approach to compute minimal approximately propagation complete and conflict propagating CNF encodings

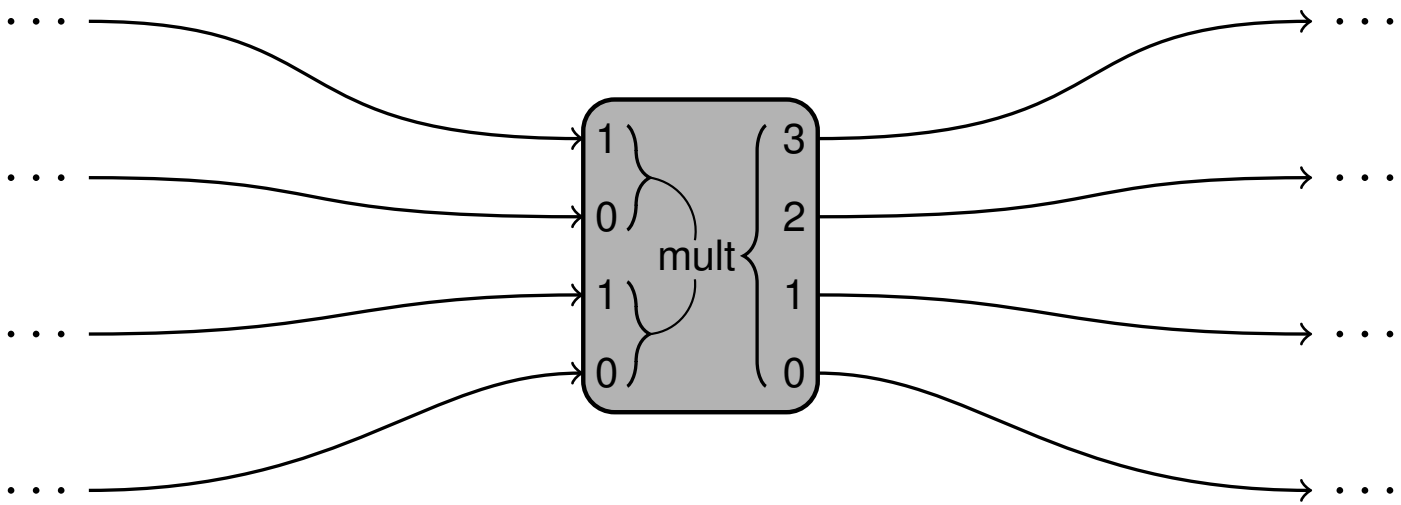
Approximate Propagation Completeness

Approximate propagation completeness with level n

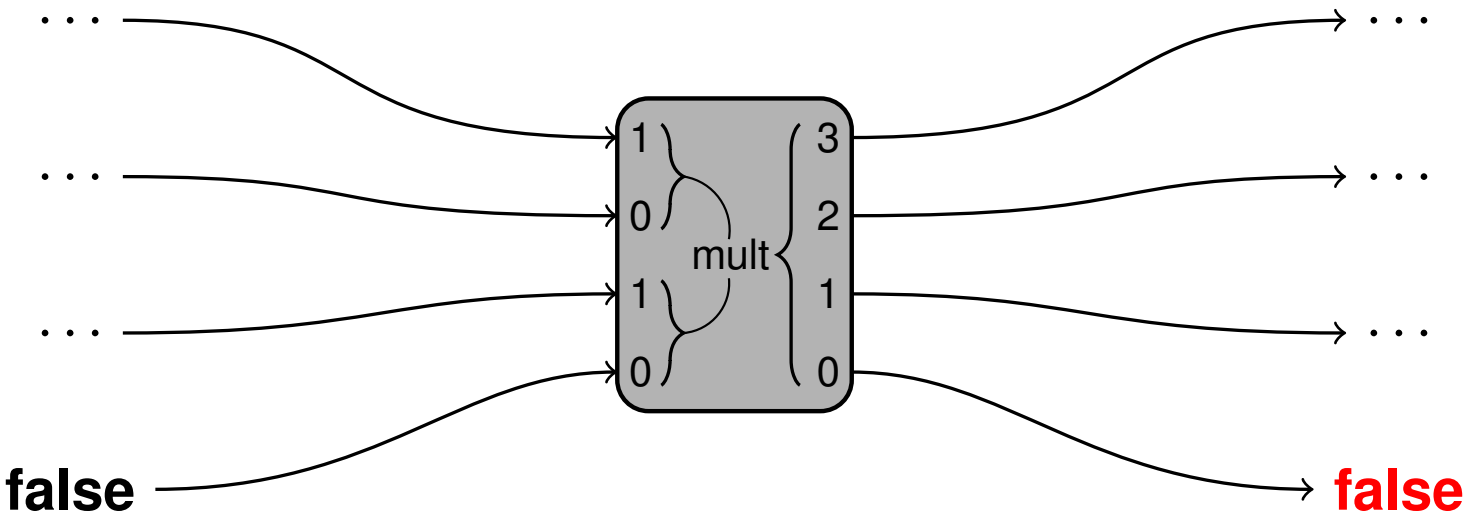
A CNF encoding ψ over a set of variables $\mathcal{V}$ is approximately propagation complete with quality level of n iff

- for every partial valuation $p : \mathcal{V} \rightharpoonup \{\textbf{false}, \textbf{true}\}$ that
  - can be extended to a valuation that satifies ψ and
  - that implies at least n different literals/variable values not defined in p,
- we have that
  - at least one implied literal can be derived from p and ψ by unit propagation.

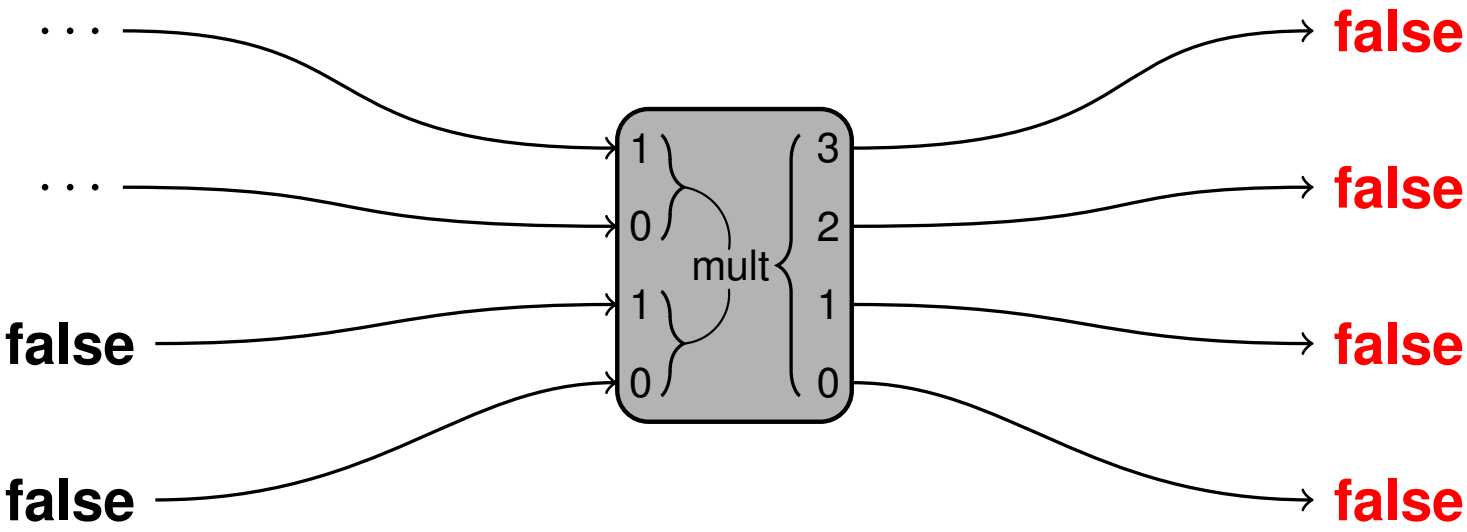
Example – Multiplier



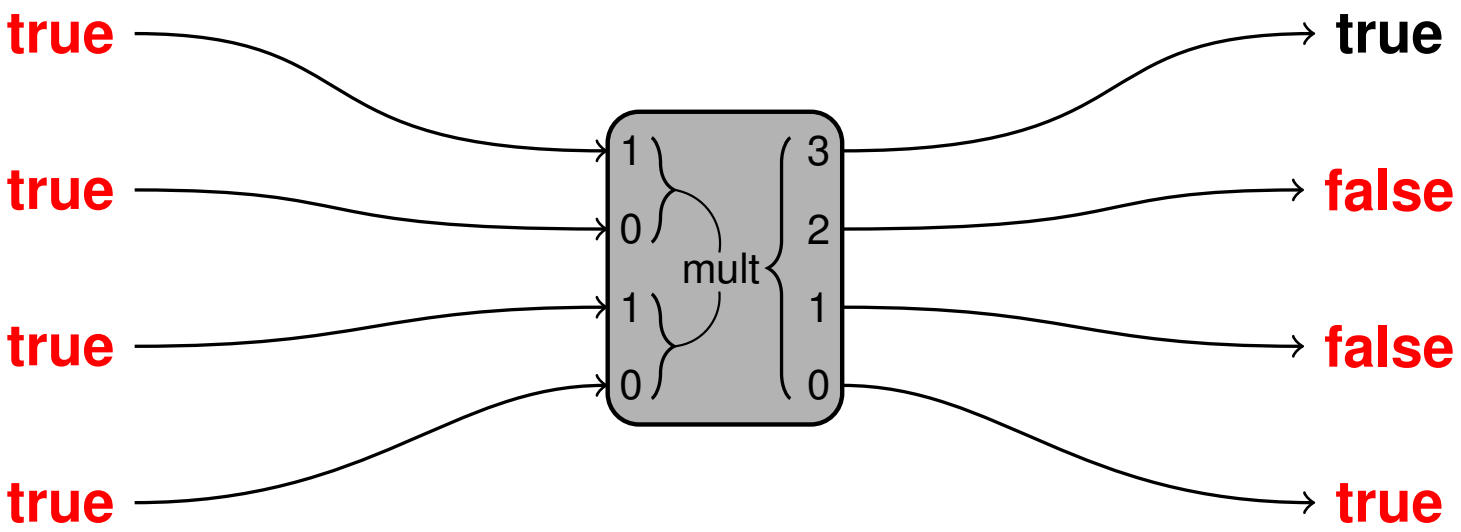
Example – Multiplier



Example – Multiplier



Example – Multiplier



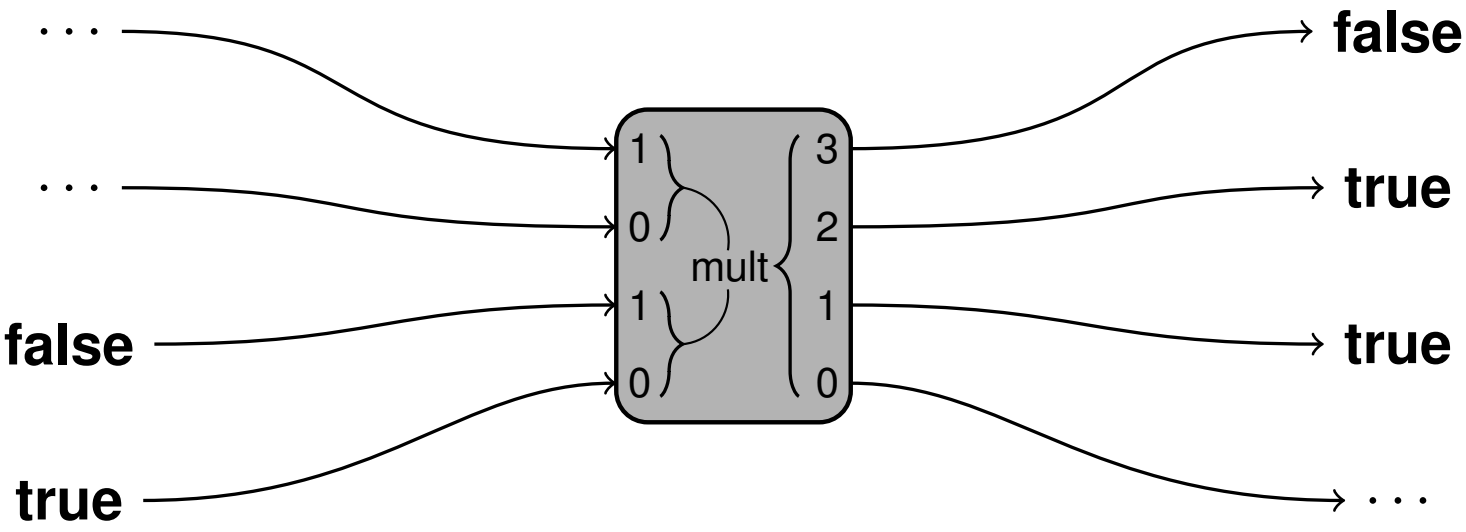
Approximate Conflict Propagation

Approximate conflict propagation with quality level *n*

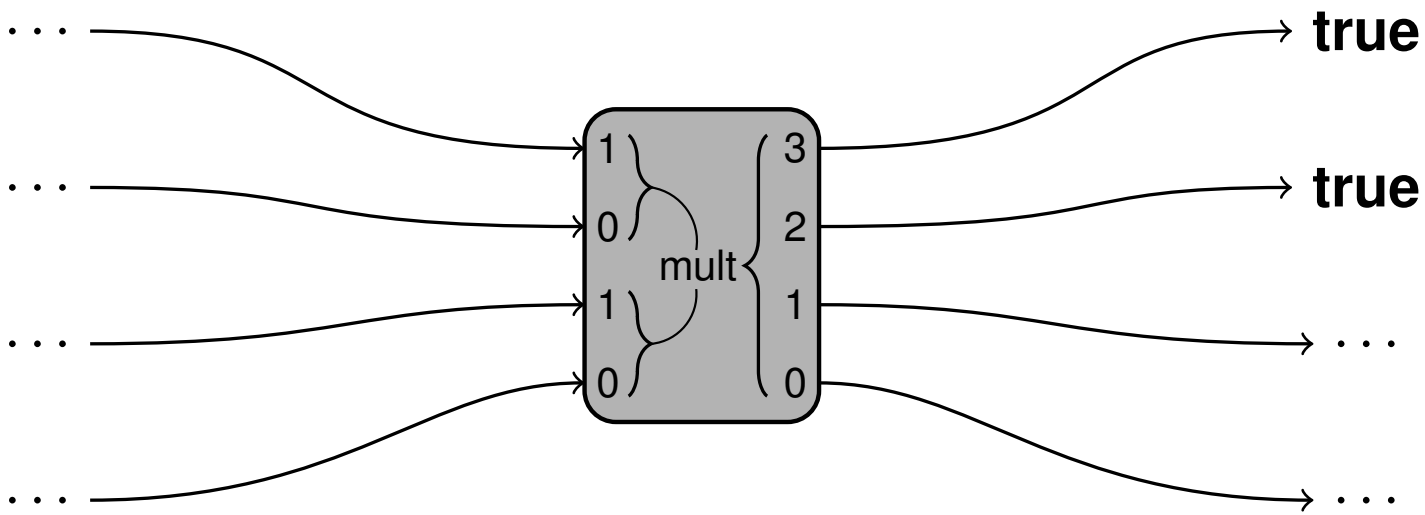
A CNF encoding ψ over a set of variables $\mathcal{V}$ is approximately conflict propagating with quality level of *n* iff

- for every partial valuation $p : \mathcal{V} \rightharpoonup \{\textbf{false}, \textbf{true}\}$ that
  - **cannot** be extended to a valuation that satifies ψ and
  - for which at most *n* variables do not have values assigned
- we have that
  - at least one implied literal can be derived from *p* and ψ by unit propagation **or**
  - there is some clause in ψ that is already violated by *p*

Example – Multiplier

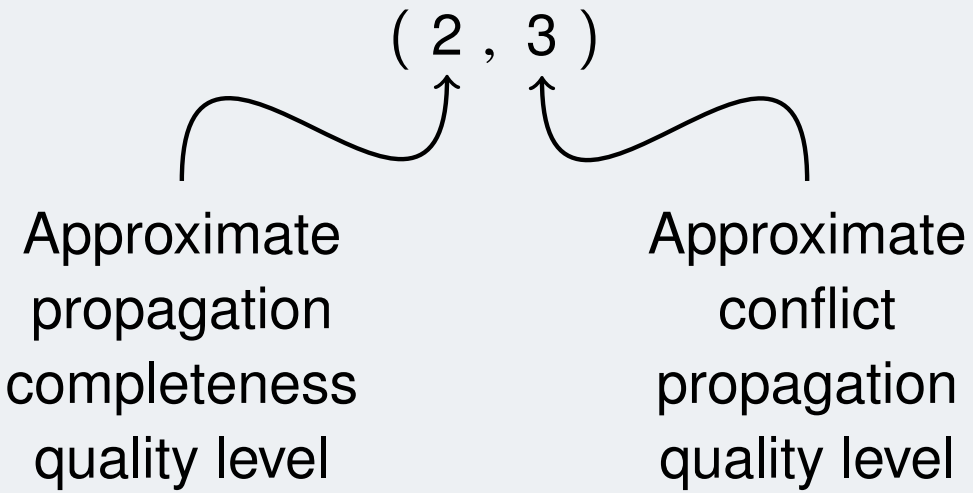


Example – Multiplier



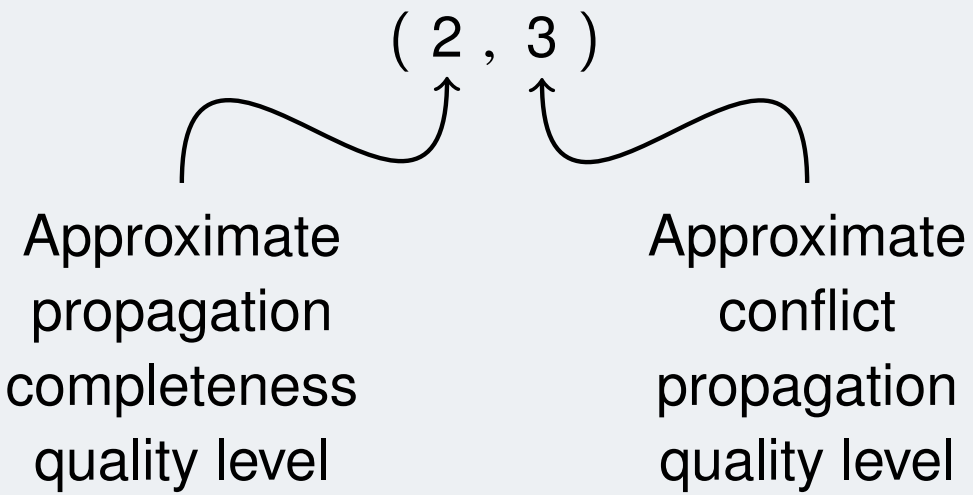
Combining the Concepts

Propagation quality tuples



Combining the Concepts

Propagation quality tuples



Important special cases

- $(1, \infty)$ – Propagation complete encodings
- $(\infty, 1)$ – Arbitrary encodings

(where ∞ denotes arbitrary numbers greater than $|\mathcal{V}|$.)

Computing Smallest Constraint Encodings

Clause Selection

Starting observation

Without loss of generality, we can assume that smallest encodings consist of *shortest* clauses implied by the constraint.

Clause Selection

Starting observation

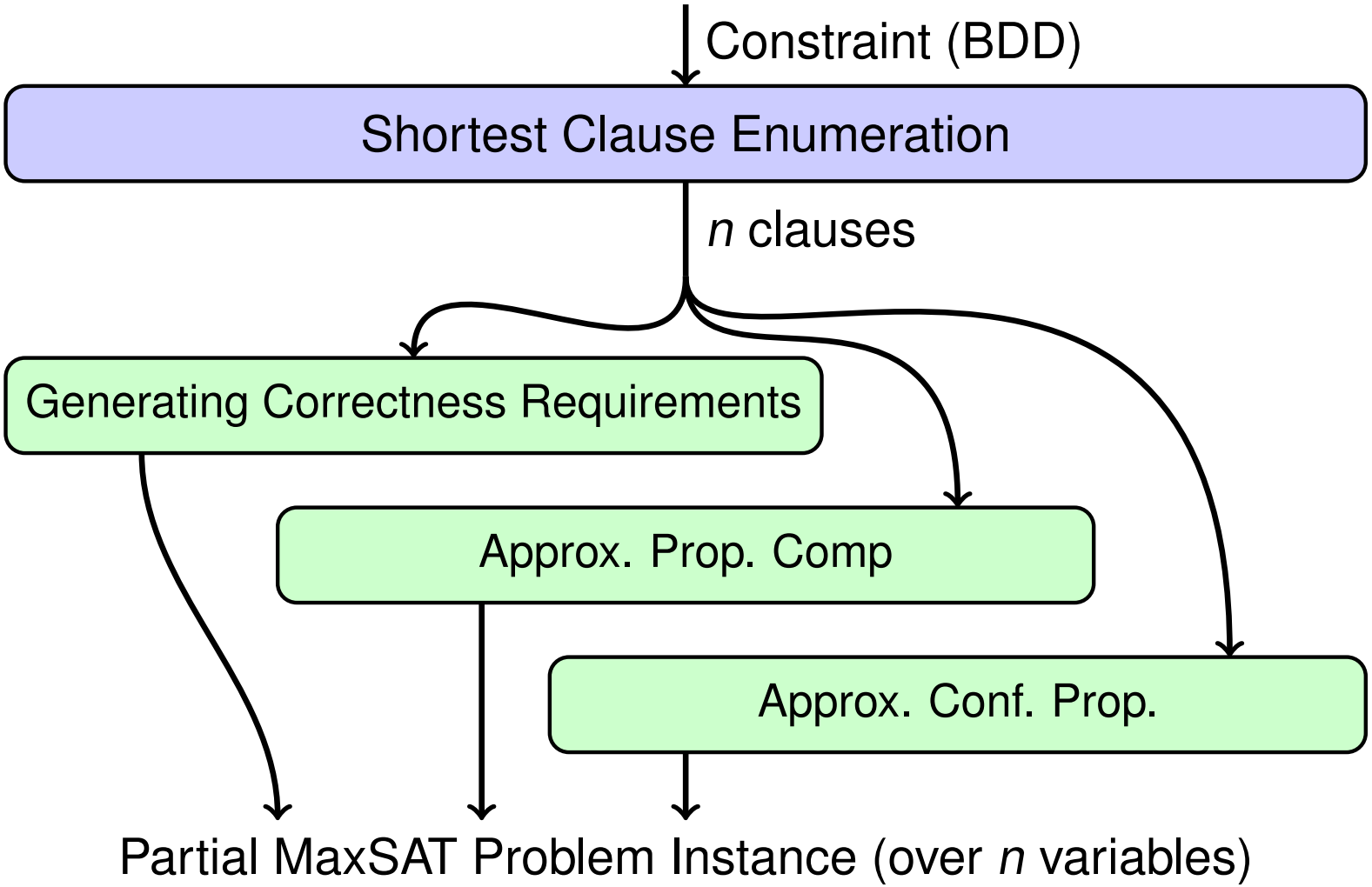
Without loss of generality, we can assume that smallest encodings consist of *shortest* clauses implied by the constraint.

Idea

We enumerate all shortest clauses from which we select a smallest subset of them such that the selected clauses

- form a *correct* encoding
- are sufficient to achieve the requested level of approximate propagation completeness
- are sufficient to achieve the requested level of approximate conflict propagation

Overall Approach



Shortest Clause Enumeration

Steps:

- Build a BDD of the constraint to be encoded
- Encode the search for a clause whose induced paths in the BDD all lead to the **false** leaf as a SAT instance
- Employ incremental SAT solving to find all *shortest* clauses

Enforcing Approximately Optimal Propagation

Steps:

- Iterate over all variables $\mathcal{V} = \{v_1, \ldots, v_m\}$ and for every $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, m\}$, build BDDs $B_{i,j}$ encoding (satisfiable) partial valuations that induce at least j literal values for the first i variables.
  $\rightarrow$ Uses an additional variable set $\mathcal{V} = \{v'_1, \ldots, v'_m\}$ to encode which variables in $\mathcal{V}$ should have concrete values
- Build the valuation BDD $B = \bigvee_{j=q_p}^{|\mathcal{V}|} B_{m,j}$
- While B is not equivalent to **false**:
  - Find a satisfying partial assignment to B
  - Compute the subset of shortest clauses that give rise to unit propagation (or a conflict) over the assignment and add a MaxSAT clause that enforces (at least) one of them to be selected
  - Remove all partial valuations from B for which all clauses in the subset give rise to unit propagation

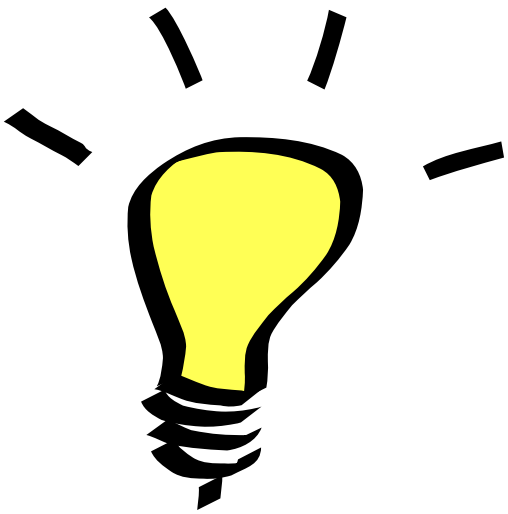
Enforcing Approximate Conflict Propagation

Steps:

- Build a BDD B for all partial valuations that cannot be extended to a satisfying valuation $\rightarrow$ Uses an additional variable set $\mathcal{V} = \{v'_1, \ldots, v'_m\}$ to encode which variables in $\mathcal{V}$ should have concrete values
- Restrict B to partial valuations for which at least $|\mathcal{V}| - q_c$ variables have defined values
- While B is not equivalent to **false**:
  - Find a satisfying partial assignment to B
  - Compute the subset of shortest clauses that give rise to unit propagation (or a conflict) over the assignment and add a MaxSAT clause that enforces (at least) one of them to be selected
  - Remove all partial valuations from B for which all clauses in the subset give rise to unit propagation or a conflict

Experiments

Questions to be Answered



- How quickly can we compute approximately propagation complete and approximately conflict propagating CNF encodings?
- How large do the encodings get?
- How much use are such encodings in practice?

Setup

Our implementation: `optic`

- Written in C++ using the SAT solver `lingeling` (Biere, 2010) and using a MaxSAT solver as an external tool, which is `LMHS` (Saikko et al., 2016) in the following.
- Available at `https://www.github.com/progirep/optic`

Comparison basis

- Greedy constraint encodings
- The `GenPCE` tool (Brain et al., 2016) for computing propagation complete CNF encodings

Constraints used for the evaluation

- The ones used for evaluating `GenPCE`
- Adders, Multipliers, Square Root, Squaring functions, All-difference constraints, ...

Example Input for `optic`

```
// Adder, 2 bits, carry output
var a0
var a1
var b0
var b1
var x0
var x1
var x2
ob0 = a0 ^ b0
cb0 = a0 & b0
ob1 = cb0 ^ a1 ^ b1
cb1 = cb0 & (a1 | b1) | (a1 & b1)
encode = true & (x0 <-> ob0) & (x1 <-> ob1) & (x2 <-> cb1)
```

Encoding Computation

Benchmark	Quality	Time	Clauses	Real Quality
Adder, 5 bits, carry output	Greedy	0.38 s	343	$(11, 11)$
	(∞, ∞)	76.9 s	90	$(9,\infty)$
	$(1, \infty)$	64.5 s	1086	$(1,\infty)$
	$(2, \infty)$	76.1 s	948	$(2,\infty)$
	$(3, \infty)$	106 s	560	$(3,\infty)$
	$(3, 3)$	timeout		
	$(\infty, 1)$	376 s	80	$(11, 13)$
	GenPCE	848 s	1086	$(1,\infty)$

Encoding Computation

Benchmark	Quality	Time	Clauses	Real Quality
cvc-add3-bw3.cnf	Greedy	1.74 s	824	$(6, 8)$
	(∞, ∞)	2.31 s	144	$(4,\infty)$
	$(1, \infty)$	1.64 s	1536	$(1,\infty)$
	$(2, \infty)$	8.30 s	808	$(2,\infty)$
	$(3, \infty)$	3.64 s	512	$(3,\infty)$
	$(3, 3)$	47.5 s	500	$(3, 4)$
	$(\infty, 1)$	10.4 s	122	$(6, 8)$
	GenPCE	19.8 s	1536	$(1,\infty)$

Encoding Computation

Benchmark	Quality	Time	Clauses	Real Quality
All-Difference 3 obj.	Greedy	0.00 s	26	$(7, 6)$
	(∞, ∞)	1.15 s	15	$(4,\infty)$
	$(1, \infty)$	0.24 s	24	$(1,\infty)$
	$(2, \infty)$	0.35 s	20	$(2,\infty)$
	$(3, \infty)$	0.39 s	15	$(3,\infty)$
	$(3, 3)$	0.34 s	15	$(3,\infty)$
	$(\infty, 1)$	0.10 s	12	$(4, 6)$
	GenPCE	0.00 s	26	$(1,\infty)$

Encoding Computation

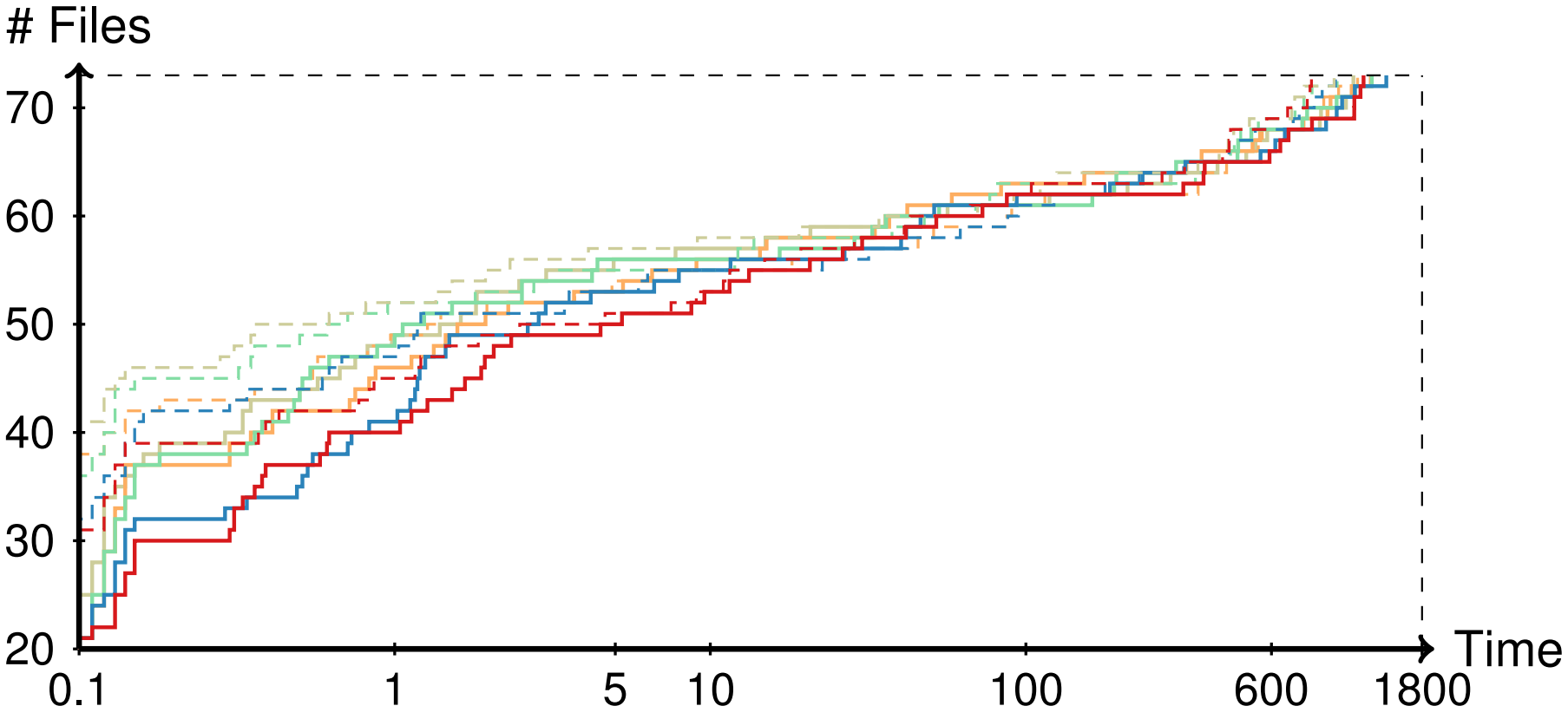
Benchmark	Quality	Time	Clauses	Real Quality
All-Difference 4 obj.	Greedy	0.10 s	86	$(13, 12)$
	(∞, ∞)	timeout		
	$(1, \infty)$	timeout		
	$(2, \infty)$	timeout		
	$(3, \infty)$	timeout		
	$(3, 3)$	timeout		
	$(\infty, 1)$	74.8 s	28	$(8, 12)$
	GenPCE	0.00 s	203	$(1,\infty)$

Encoding Computation

Benchmark	Quality	Time	Clauses	Real Quality
All-Difference 4 obj.	Greedy	0.10 s	86	$(13, 12)$
	(∞, ∞)	timeout		
	$(1, \infty)$	18.1 h	200	$(1, \infty)$
	$(2, \infty)$	28.4 h	156	$(2, \infty)$
	$(3, \infty)$	timeout		
	$(3, 3)$	timeout		
	$(\infty, 1)$	74.8 s	28	$(8, 12)$
	GenPCE	0.00 s	203	$(1,\infty)$

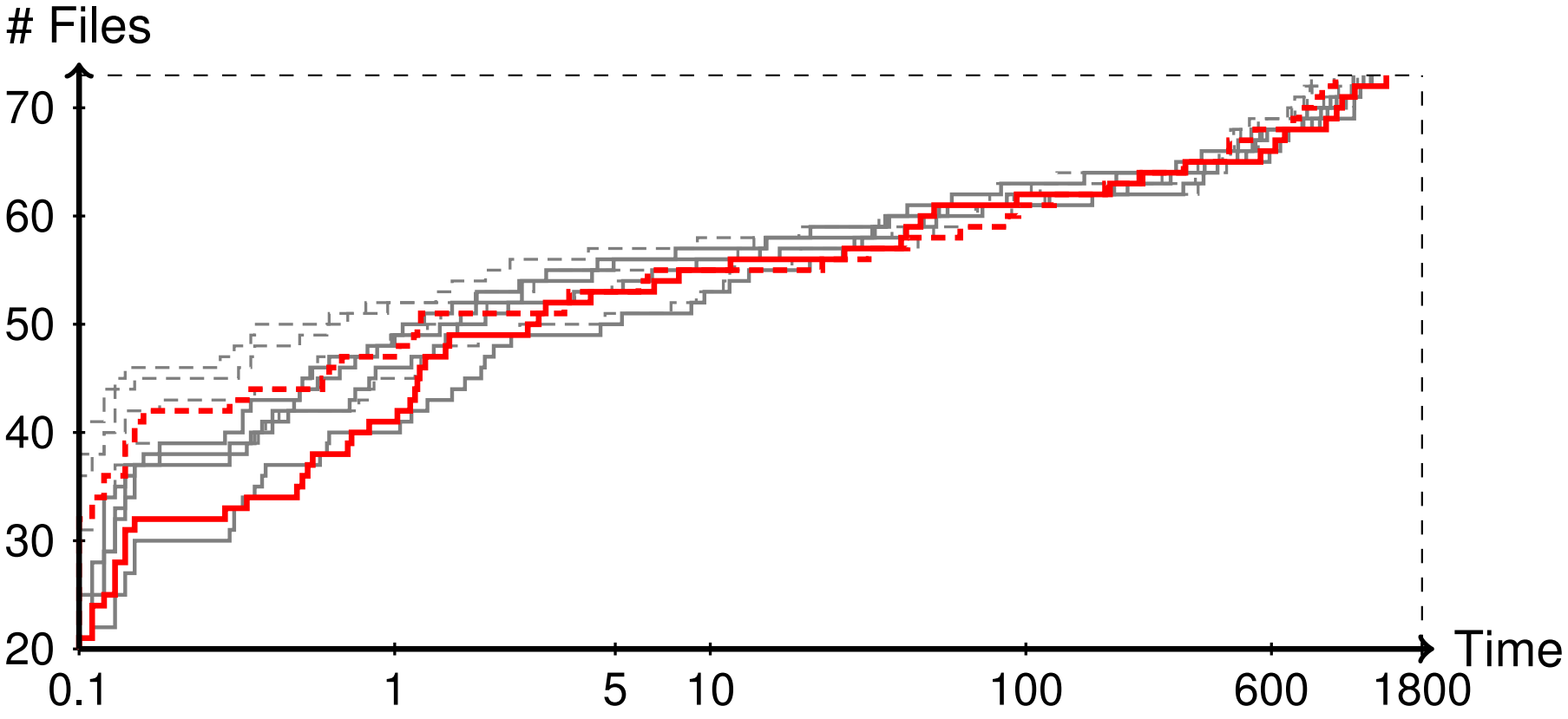
MaxSAT solving times for the $(1, \infty)$ and $(2, \infty)$ cases:
11.2 minutes and 621.3 minutes

SAT Solving Performance – Integer Factoring



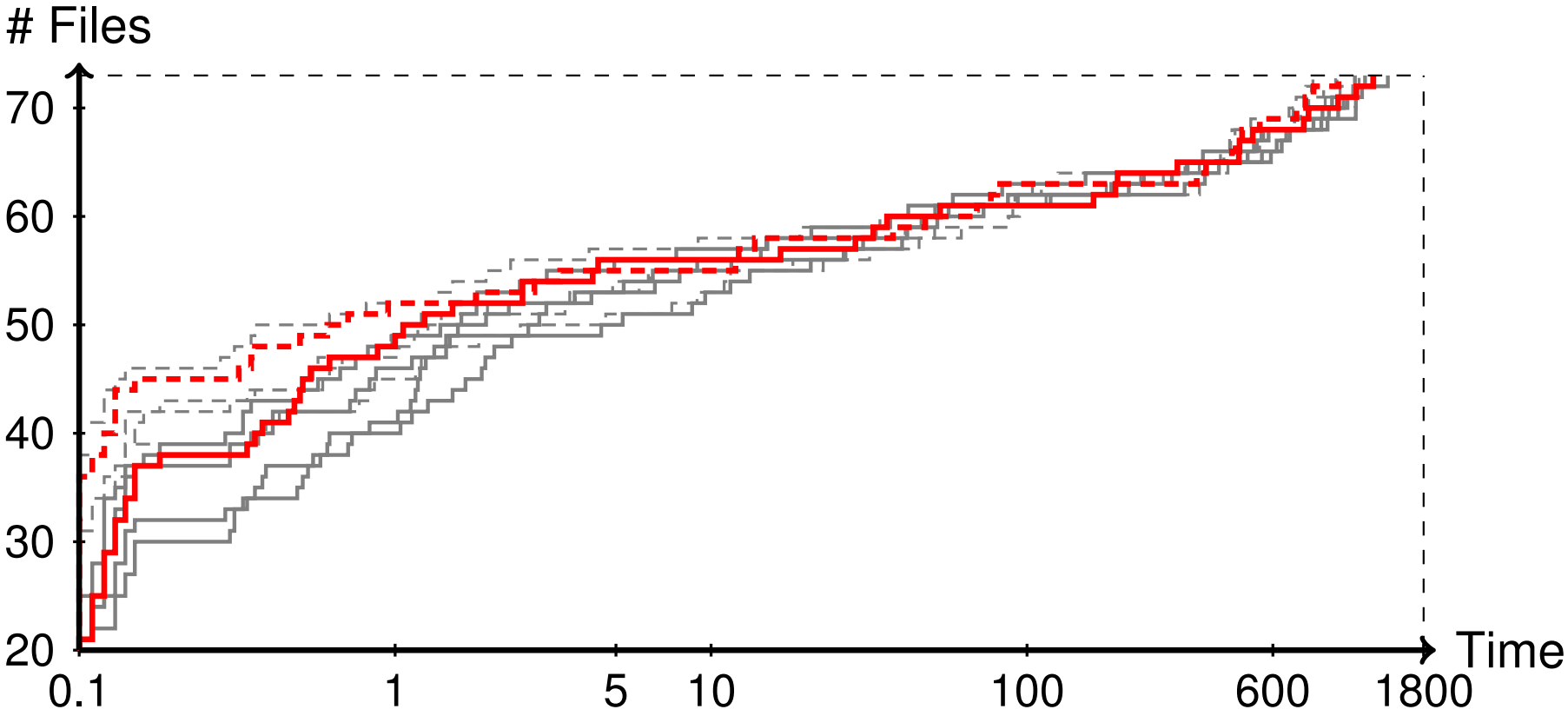
Propagation quality levels: $(\infty, 1)$, $(\infty, 1)$, $(3, 3)$, (∞, ∞), Greedy.
Time is in seconds.
`MapleSAT_LRB`: solid, `lingeling bbc 9230380`: dashed

SAT Solving Performance – Integer Factoring



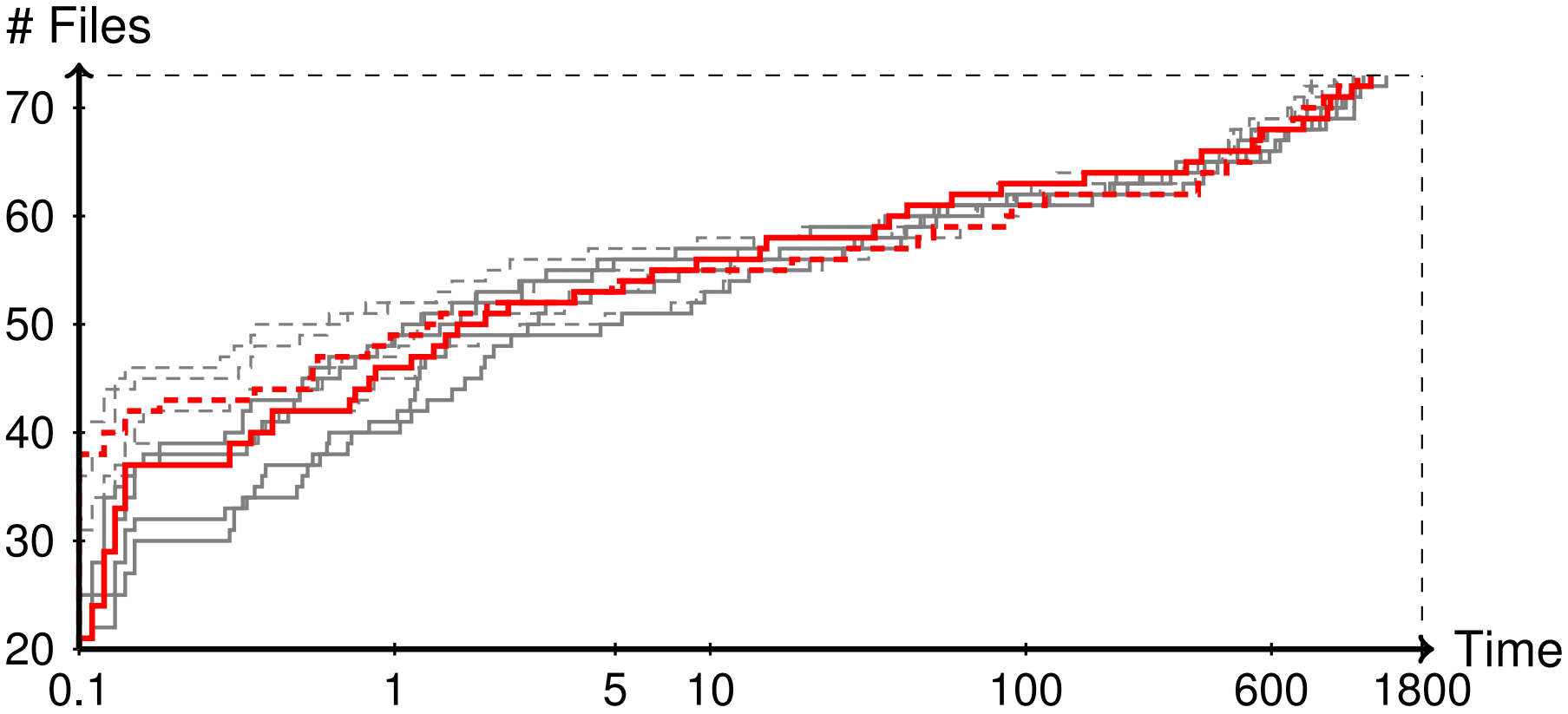
Propagation quality level: **Greedy**
Time is in seconds.
`MapleSAT_LRB`: solid, `lingeling bbc 9230380`: dashed

SAT Solving Performance – Integer Factoring



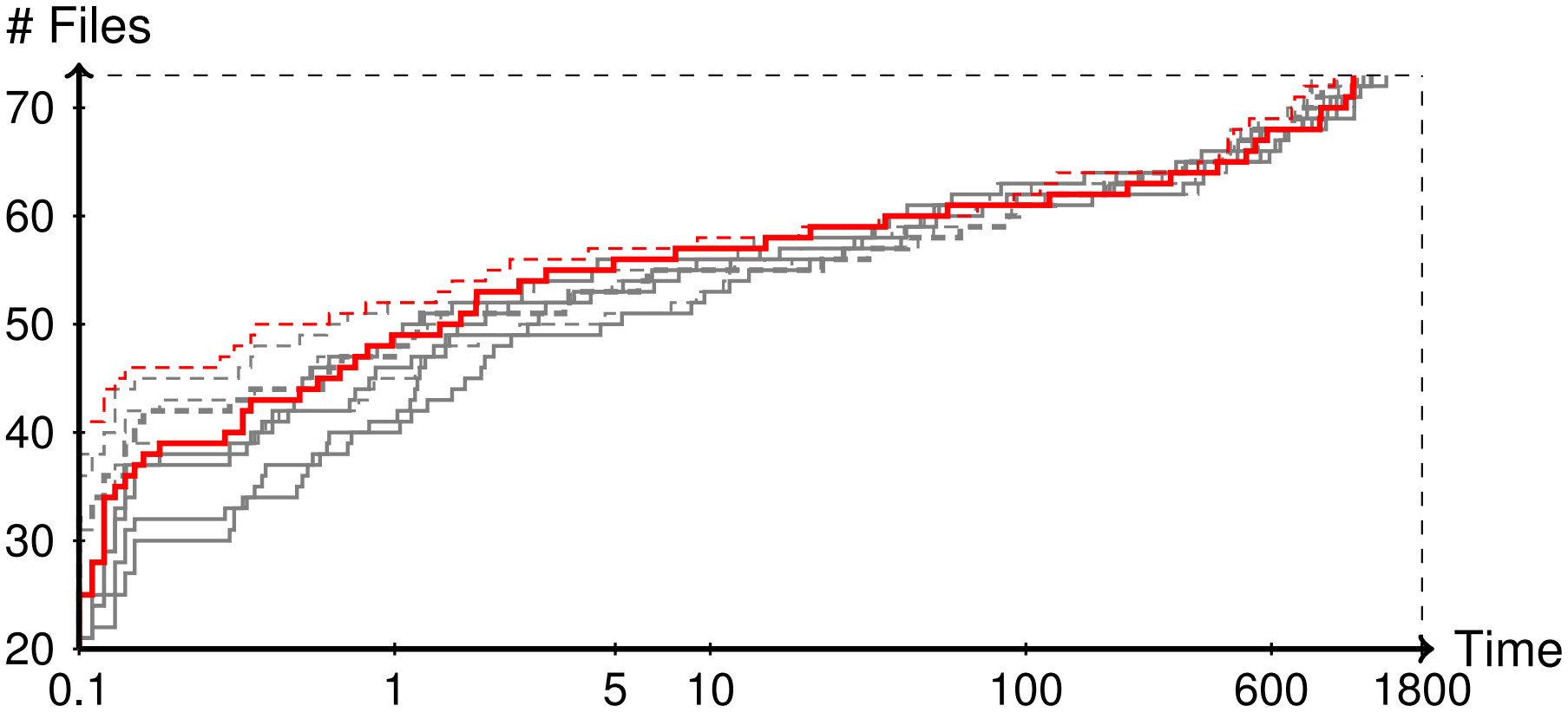
Propagation quality level: $(3, 3)$
Time is in seconds.
`MapleSAT_LRB`: solid, `lingeling bbc 9230380`: dashed

SAT Solving Performance – Integer Factoring



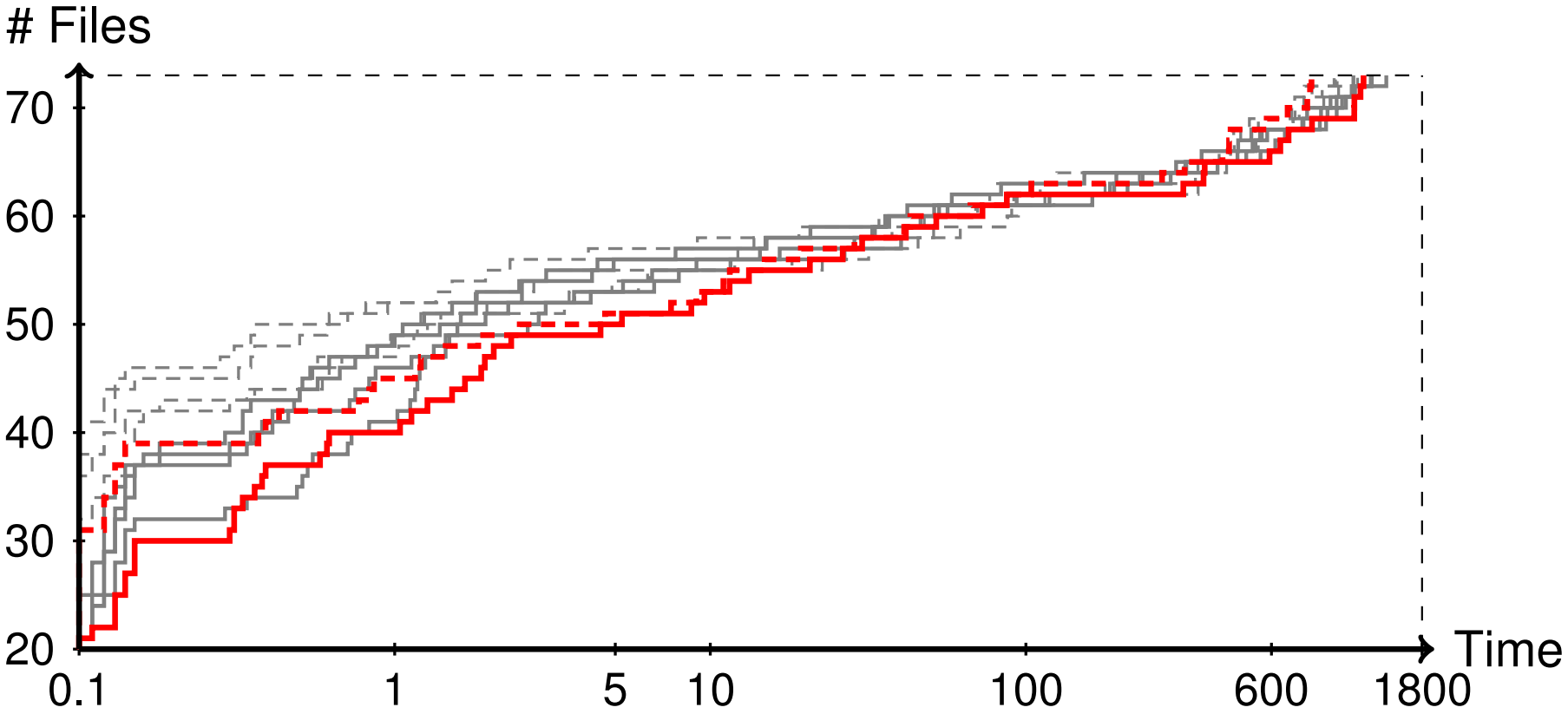
Propagation quality level: $(1, \infty)$
Time is in seconds.
`MapleSAT_LRB`: solid, `lingeling bbc 9230380`: dashed

SAT Solving Performance – Integer Factoring



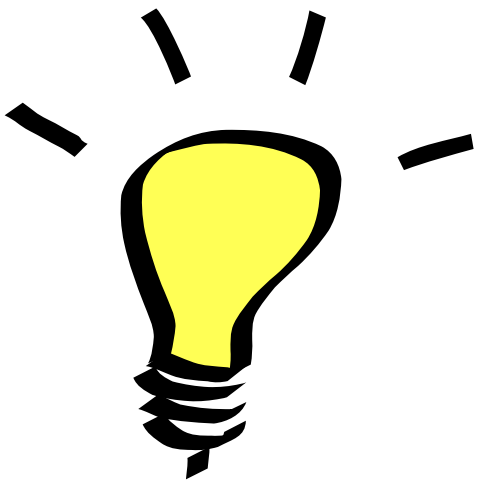
Propagation quality level: (∞, ∞)
Time is in seconds.
`MapleSAT_LRB`: solid, `lingeling bbc 9230380`: dashed

SAT Solving Performance – Integer Factoring



Propagation quality level: $(\infty, 1)$
Time is in seconds.
`MapleSAT_LRB`: solid, `lingeling bbc 9230380`: dashed

Conclusion



New concepts

- **Approximate** Propagation Completeness
- **Approximate** Conflict Propagation

Technical contributions

- An efficient approach to compute minimal approximately propagation complete and conflict propagating CNF encodings
- Encoding tool available on:
  `http://www.github.com/progirep/optic`
- New MaxSAT instances for the evaluation

References I

Armin Biere. Lingeling, Plingeling, PicoSAT and PrecoSAT at SAT race 2010. FMV Report Series Technical Report 10/1, Johannes Kepler University, Linz, Austria, 2010.

Lucas Bordeaux and João Marques-Silva. Knowledge compilation with empowerment. In *SOFSEM 2012: Theory and Practice of Computer Science - 38th Conference on Current Trends in Theory and Practice of Computer Science, Špindlerův Mlýn, Czech Republic, January 21-27, 2012*, pages 612–624, 2012. doi: 10.1007/978-3-642-27660-6_50. URL https://doi.org/10.1007/978-3-642-27660-6_50.

Martin Brain, Liana Hadarean, Daniel Kroening, and Ruben Martins. Automatic generation of propagation complete SAT encodings. In *Verification, Model Checking, and Abstract Interpretation - 17th International Conference, VMCAI 2016, St. Petersburg, FL, USA, January 17-19, 2016*, pages 536–556, 2016. doi: 10.1007/978-3-662-49122-5_26. URL http://dx.doi.org/10.1007/978-3-662-49122-5_26.

Paul Saikko, Jeremias Berg, and Matti Järvisalo. LMHS: A SAT-IP hybrid MaxSAT solver. In *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016*, pages 539–546, 2016. doi: 10.1007/978-3-319-40970-2_34. URL https://doi.org/10.1007/978-3-319-40970-2_34.