To Encode or to Propagate?
The Best Choice for Each Constraint in SAT

I. Abío, R. Nieuwenhuis, A. Oliveras, E. Rodríguez-Carbonell, P. Stuckey

Rich Model Toolkit COST Action Meeting
16 June 2013
Malta

Overview

- Motivation: solving constraints with SAT technology
  - Eager Approach: SAT encodings
  - Lazy Approach: SMT/propagators
- Choosing Right: Related Work and Contributions
- Experimental Results
- Conclusions and Future Work

Motivation

- Goal: solving systems of constraints with SAT tools
- Applications:
  - Many in scheduling, timetabling, planning, etc.
  - Also in constraint-based program analysis/synthesis

Motivation

- Goal: solving systems of constraints with SAT tools
- Applications:
  - Many in scheduling, timetabling, planning, etc.
  - Also in constraint-based program analysis/synthesis
- Why using SAT? (cf. Linear/Constraint Programming)
  - SAT tech outperforms other tools on real-world problems with a single, fully automatic variable selection strategy!
  - Hence problem solving is essentially declarative

Motivation

- Goal: solving systems of constraints with SAT tools
- Applications:
  - Many in scheduling, timetabling, planning, etc.
  - Also in constraint-based program analysis/synthesis
- Why using SAT? (cf. Linear/Constraint Programming)
  - SAT tech outperforms other tools on real-world problems with a single, fully automatic variable selection strategy!
  - Hence problem solving is essentially declarative
- However, propositional logic is a very low-level language for complex constraints

Cardinality and PB Constraints

Example: limited-resource problems

- Some tasks $\{1, 2, \ldots, n\}$ must be carried out
- Tasks require some (limited) resources
- Variable $a_{i,t}$ is true if task i is active at time t

Cardinality and PB Constraints

Example: limited-resource problems

- Some tasks $\{1,2,\ldots,n\}$ must be carried out
- Tasks require some (limited) resources
- Variable $a_{i,t}$ is true if task i is active at time t

- **Constraint:** There are no more active tasks than machines:

$$a_{1,t} + a_{2,t} + \ldots + a_{n,t} \leq 20$$

In general, cardinality cons. are of the form $\Sigma_{i=1}^{n} x_i \leq k$

Cardinality and PB Constraints

Example: limited-resource problems

- Some tasks $\{1,2,\ldots,n\}$ must be carried out
- Tasks require some (limited) resources
- Variable $a_{i,t}$ is true if task i is active at time t

- **Constraint:** There are no more active tasks than machines:

$$a_{1,t} + a_{2,t} + \ldots + a_{n,t} \leq 20$$

In general, cardinality cons. are of the form $\Sigma_{i=1}^{n} x_i \leq k$

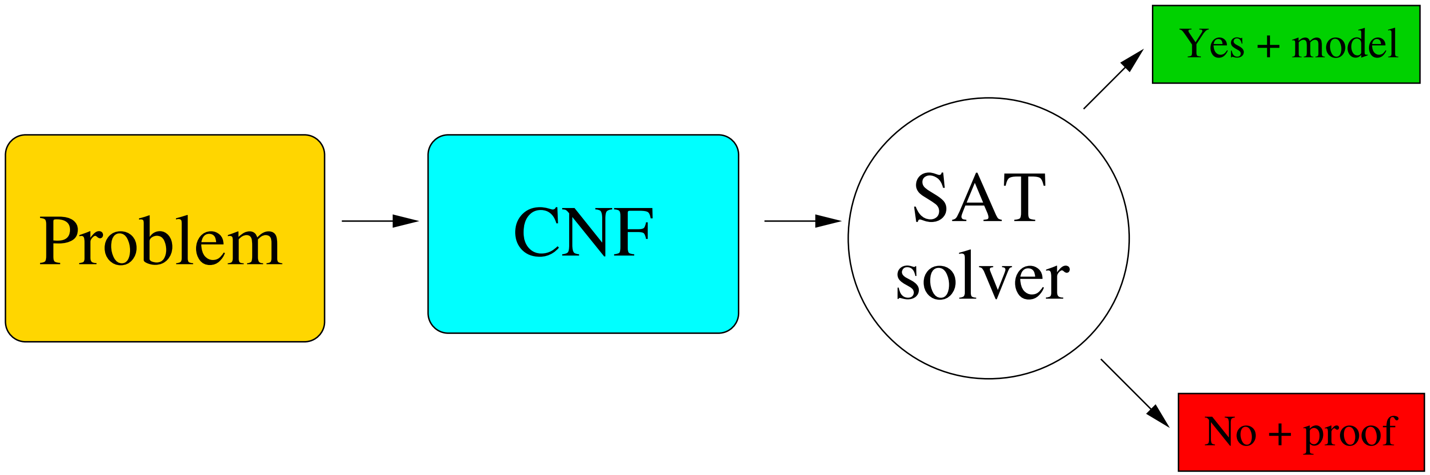
- **Constraint:** The max number of workers is not exceeded:

$$3a_{1,t} + 4a_{2,t} + \ldots + 10a_{n,t} \leq 50$$

In general, pseudo-Boolean (PB) cons. are of the form $\Sigma_{i=1}^{n} a_i x_i \leq k$

SAT Encodings

- Express constraint C with (CNF) formula F (the encoding) s.t.
  - For each solution to C there is a model of F
  - For each model of F there is a solution to C



SAT Encodings of Cardinality Constraints (1)

- Example: for a cardinality constraint $\Sigma_{i=1}^{n} x_i < k$ we have:
  - Naive encoding.
    - Variables: the same $x_1, \ldots, x_n$
    - Clauses: $\overline{x_{i_1}} \vee \ldots \vee \overline{x_{i_k}}$ for all $1 \leq i_1 < \ldots < i_k \leq n$
    - This is $\binom{n}{k}$ clauses!

SAT Encodings of Cardinality Constraints (1)

- Example: for a cardinality constraint $\Sigma_{i=1}^{n} x_i < k$ we have:
  - Naive encoding.
    - Variables: the same $x_1, \ldots, x_n$
    - Clauses: $\overline{x_{i_1}} \vee \ldots \vee \overline{x_{i_k}}$ for all $1 \leq i_1 < \ldots < i_k \leq n$
    - This is $\binom{n}{k}$ clauses!
  - Sorting network encoding.

    Build a circuit that sorts (say, decreasingly) n bits with inputs $x_1, \ldots, x_n$ and outputs new variables $y_1, \ldots, y_n$

    - Variables: $x_1, \ldots, x_n$ and gates of the circuit
    - Clauses: Tseitin encoding of the circuit + unit clause $\overline{y_k}$
    - Can be done with $O(n \log^2(n))$ clauses and new vars!

SAT Encodings of Cardinality Constraints (2)

- Only first k outputs suffice:
  cardinality networks just use $O(n \log^2(k))$ clauses, vars

SAT Encodings of Cardinality Constraints (2)

- Only first k outputs suffice:
  cardinality networks just use $O(n \log^2(k))$ clauses, vars

- In the following:
  cardinality networks used for encoding cardinality constraints
  (among most robust, efficient encodings for these constraints)

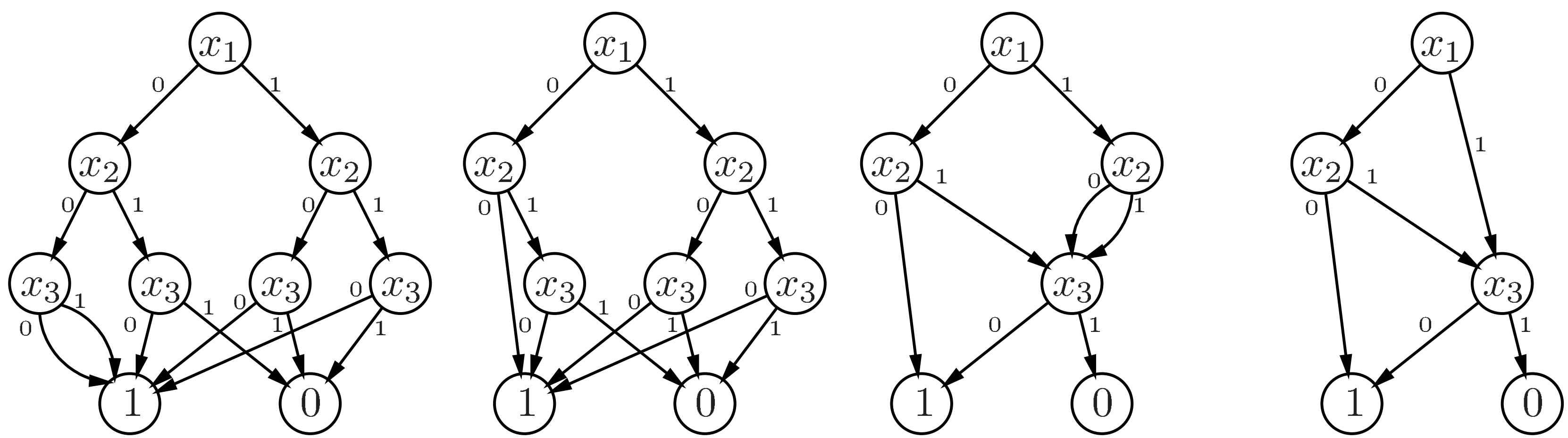
SAT Encodings of PB Constraints (1)

- Several encodings exist
  - Unary/binary adder circuits
  - Sorting networks
  - BDD's

SAT Encodings of PB Constraints (1)

- Several encodings exist
  - Unary/binary adder circuits
  - Sorting networks
  - BDD's
- Example of encoding $2x_1 + 3x_2 + 5x_3 \leq 6$ with a BDD:

  Construct the (RO)BDD wrt. ordering $x_1 \succ x_2 \succ x_3$...



... and relate truth values of parents and children according to selector variables

SAT Encodings of PB Constraints (2)

- In the encoding of $\Sigma_{i=1}^{n} a_i x_i \leq k$ with BDD's:
  - Variables: $x_1, \ldots, x_n$ and one for each node of the BDD
  - Clauses: if n is a node with selector variable x and true and false children t and f, express

$$x \rightarrow (n \leftrightarrow t) \qquad \overline{x} \rightarrow (n \leftrightarrow f)$$

  - Linear number of clauses/variables in the size of the BDD

SAT Encodings of PB Constraints (2)

- In the encoding of $\Sigma_{i=1}^{n} a_i x_i \leq k$ with BDD's:
  - Variables: $x_1, \ldots, x_n$ and one for each node of the BDD
  - Clauses: if n is a node with selector variable x and true and false children t and f, express

$$x \rightarrow (n \leftrightarrow t) \qquad \overline{x} \rightarrow (n \leftrightarrow f)$$

  - Linear number of clauses/variables in the size of the BDD

- There are families of PB constraints for which no ordering of variables yields polynomial-size BDD-based encodings
... but this rarely occurs in practice

SAT Encodings of PB Constraints (2)

- In the encoding of $\Sigma_{i=1}^{n} a_i x_i \leq k$ with BDD's:
  - Variables: $x_1, \ldots, x_n$ and one for each node of the BDD
  - Clauses: if n is a node with selector variable x and true and false children t and f, express

$$x \rightarrow (n \leftrightarrow t) \qquad \overline{x} \rightarrow (n \leftrightarrow f)$$

  - Linear number of clauses/variables in the size of the BDD

- There are families of PB constraints for which no ordering of variables yields polynomial-size BDD-based encodings
  ... but this rarely occurs in practice

- In the following:
  BDD's used for encoding PB constraints
  (among most efficient encodings in practice)

Pros and Cons of SAT Encodings

- Encodings introduce auxiliary variables that:
  - ✓ yield smaller formulations,
  - ✓ may produce more general/shorter lemmas,
  - ✓ can be used for case splitting,
  - ✗ but make search space larger

Pros and Cons of SAT Encodings

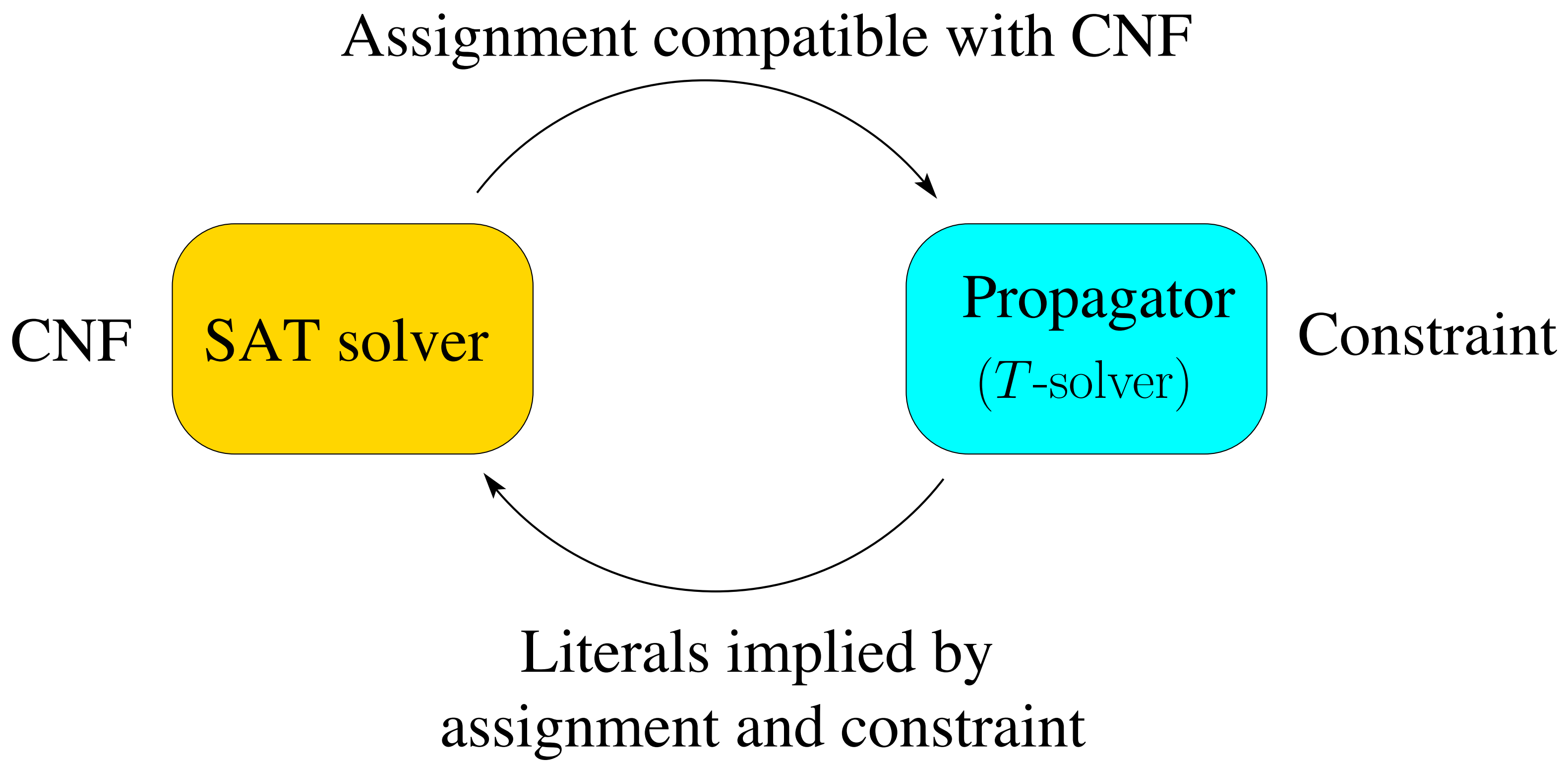
- Encodings introduce auxiliary variables that:
  - ✓ yield smaller formulations,
  - ✓ may produce more general/shorter lemmas,
  - ✓ can be used for case splitting,
  - ✗ but make search space larger
- ✗ Encodings impractical if problem has many/large constraints

SMT/propagators (1)

- Instead of eagerly encoding the constraint, deal with it lazily

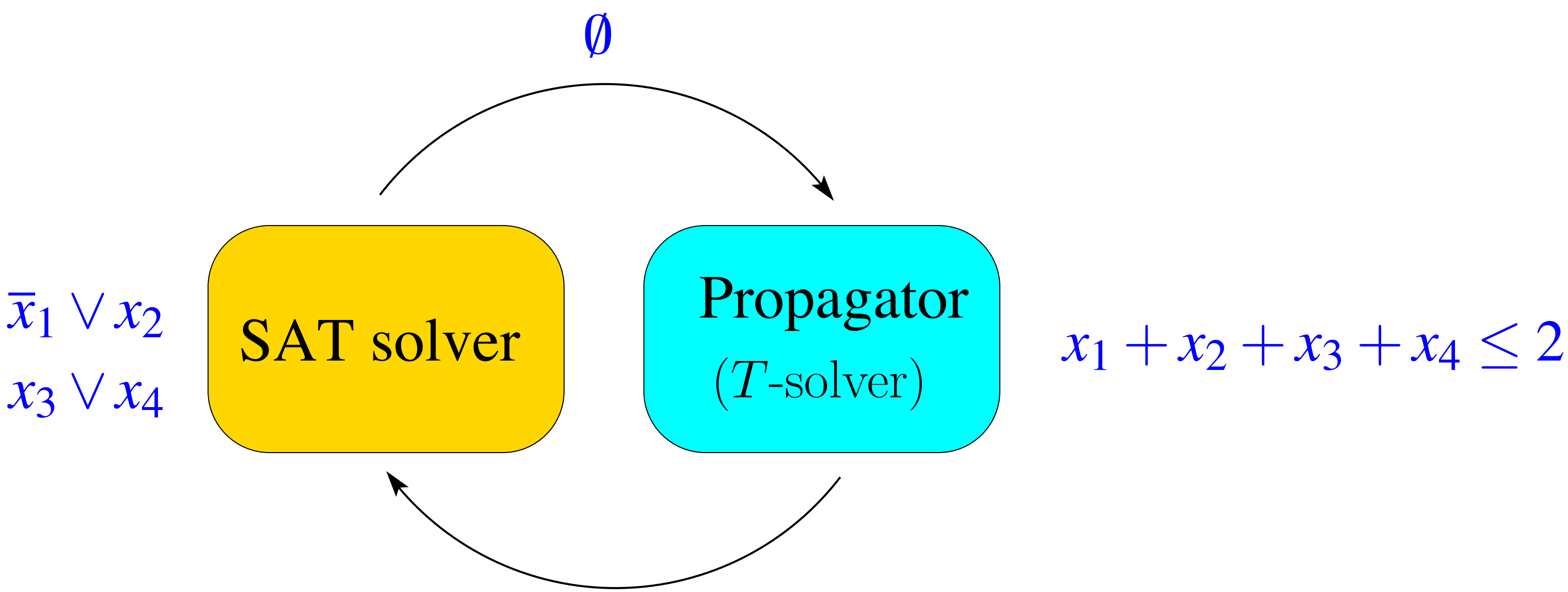
SMT/propagators (1)

- Instead of eagerly encoding the constraint, deal with it lazily
- DPLL(T) approach for solving CNF $\wedge$ Constraint :



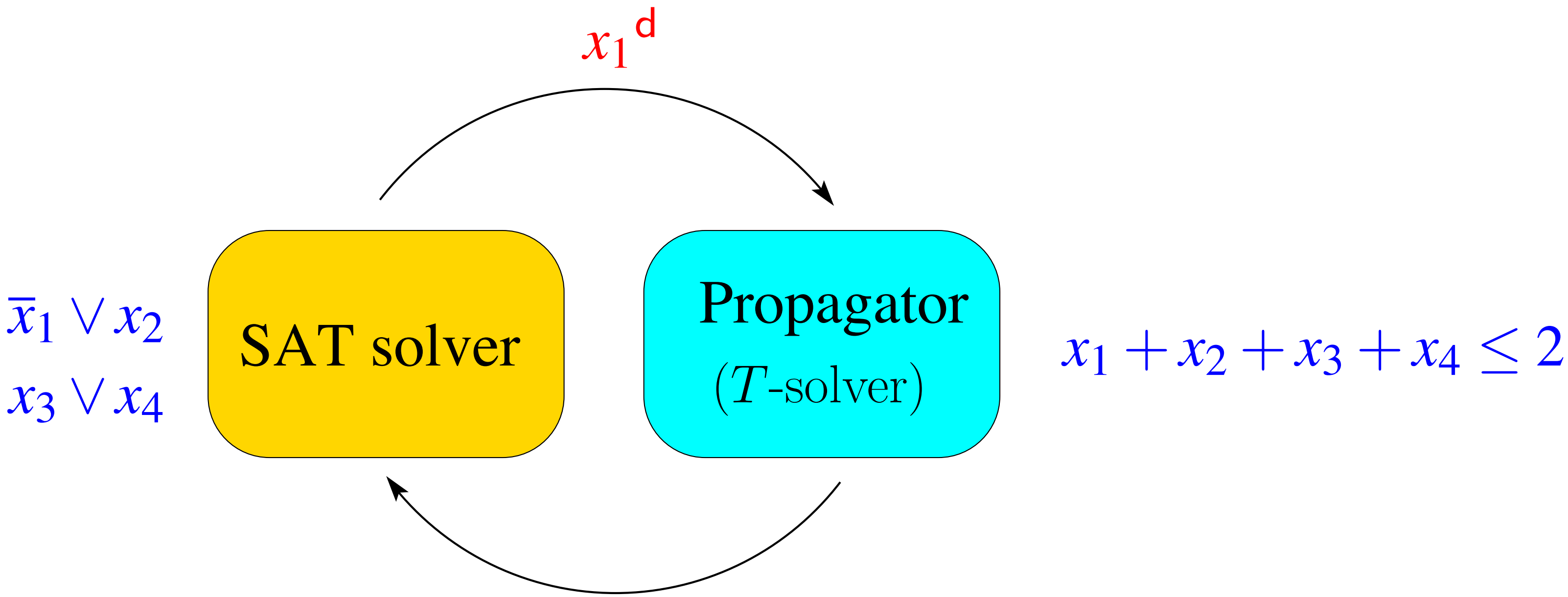
SMT/propagators (2)

- Example: $\overline{x}_1 \vee x_2, \quad x_3 \vee x_4, \quad x_1 + x_2 + x_3 + x_4 \leq 2$



SMT/propagators (2)

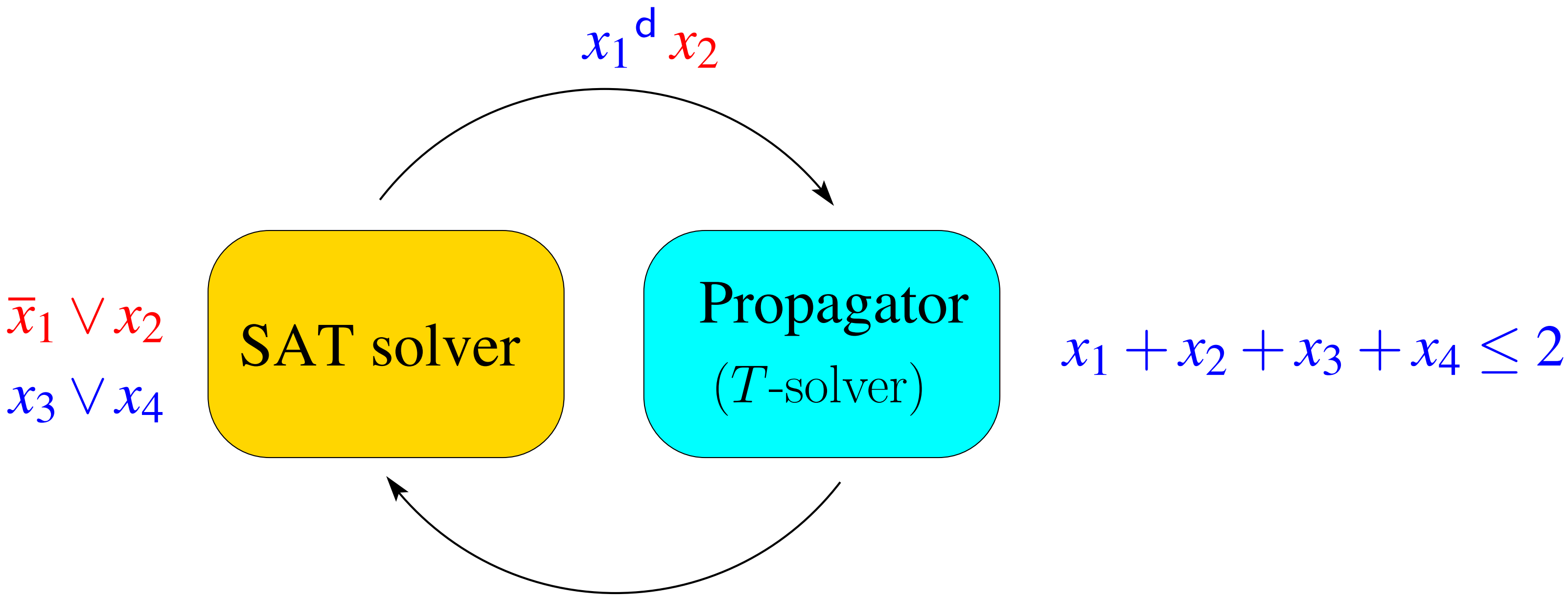
- Example: $\bar{x}_1 \vee x_2, \quad x_3 \vee x_4, \quad x_1 + x_2 + x_3 + x_4 \leq 2$



Decide

SMT/propagators (2)

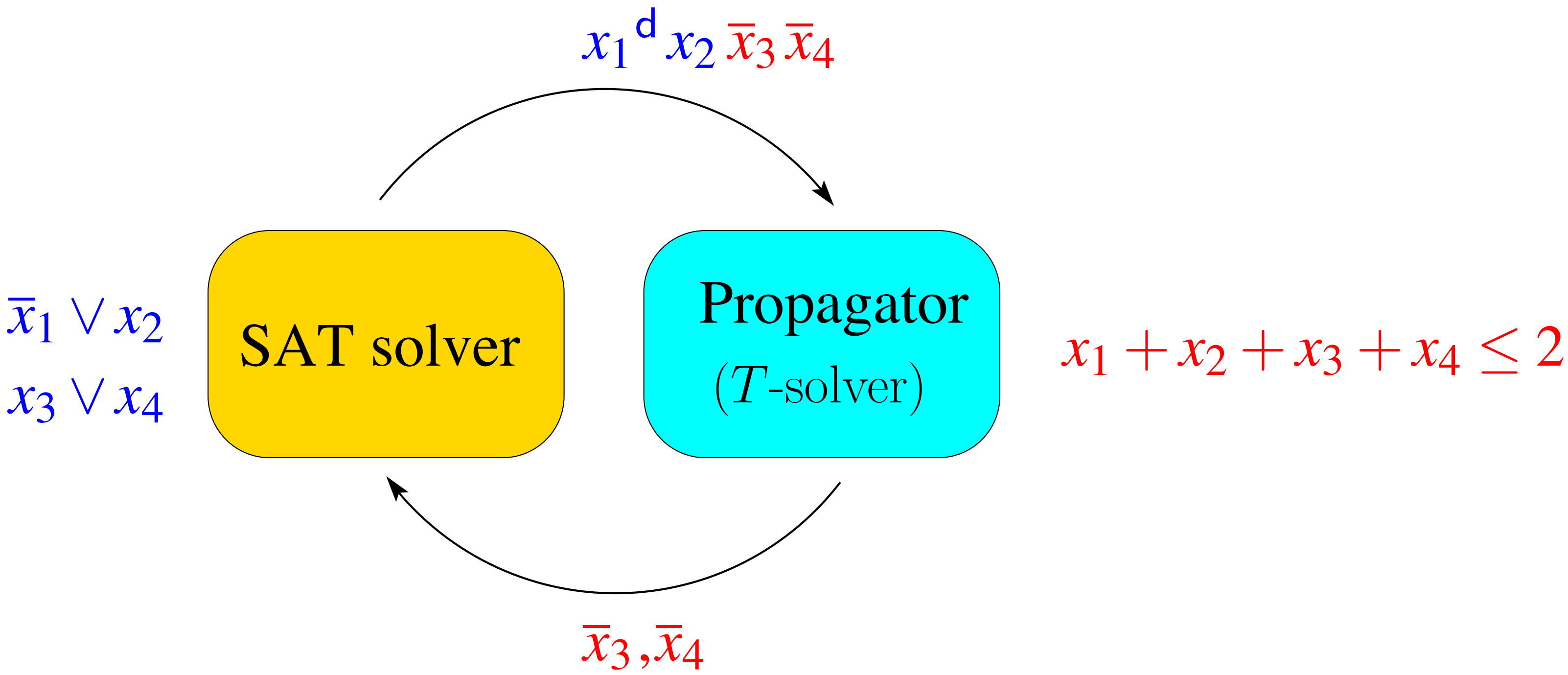
- Example: $\bar{x}_1 \vee x_2, \quad x_3 \vee x_4, \quad x_1 + x_2 + x_3 + x_4 \leq 2$



UnitPropagate

SMT/propagators (2)

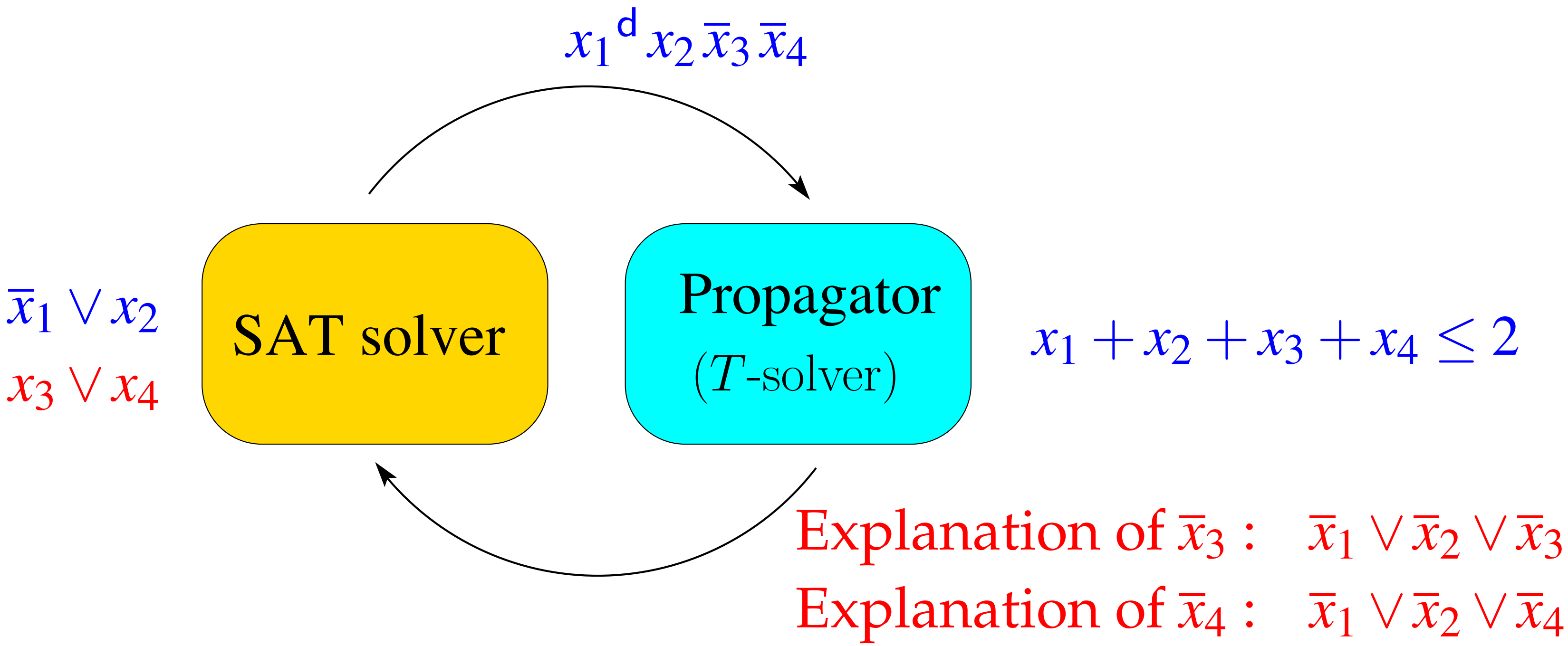
- Example: $\bar{x}_1 \vee x_2, \quad x_3 \vee x_4, \quad x_1 + x_2 + x_3 + x_4 \leq 2$



T-Propagate

SMT/propagators (2)

- Example: $\bar{x}_1 \lor x_2, \quad x_3 \lor x_4, \quad x_1 + x_2 + x_3 + x_4 \leq 2$

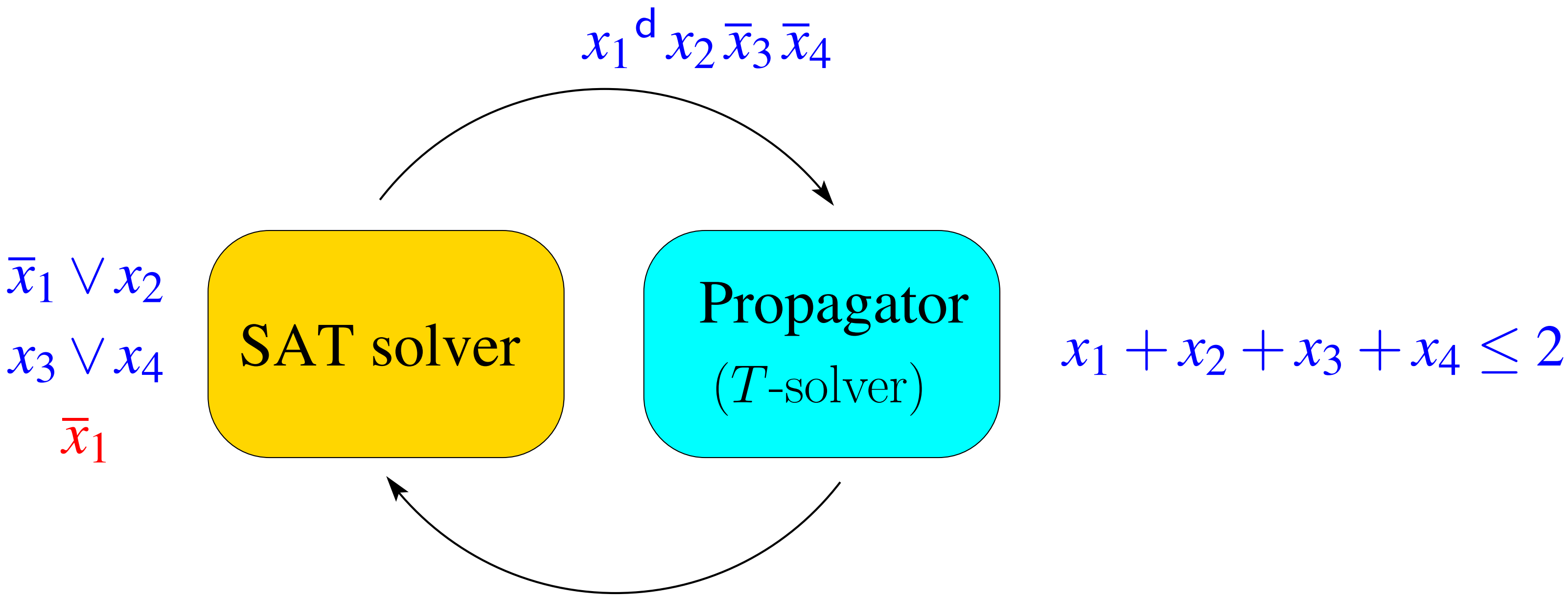


Conflict!

$$\dfrac{\dfrac{\dfrac{x_3 \lor x_4 \qquad \bar{x}_1 \lor \bar{x}_2 \lor \bar{x}_4}{\bar{x}_1 \lor \bar{x}_2 \lor x_3} \qquad \bar{x}_1 \lor \bar{x}_2 \lor \bar{x}_3}{\bar{x}_1 \lor \bar{x}_2} \qquad \bar{x}_1 \lor x_2}{\bar{x}_1}$$

SMT/propagators (2)

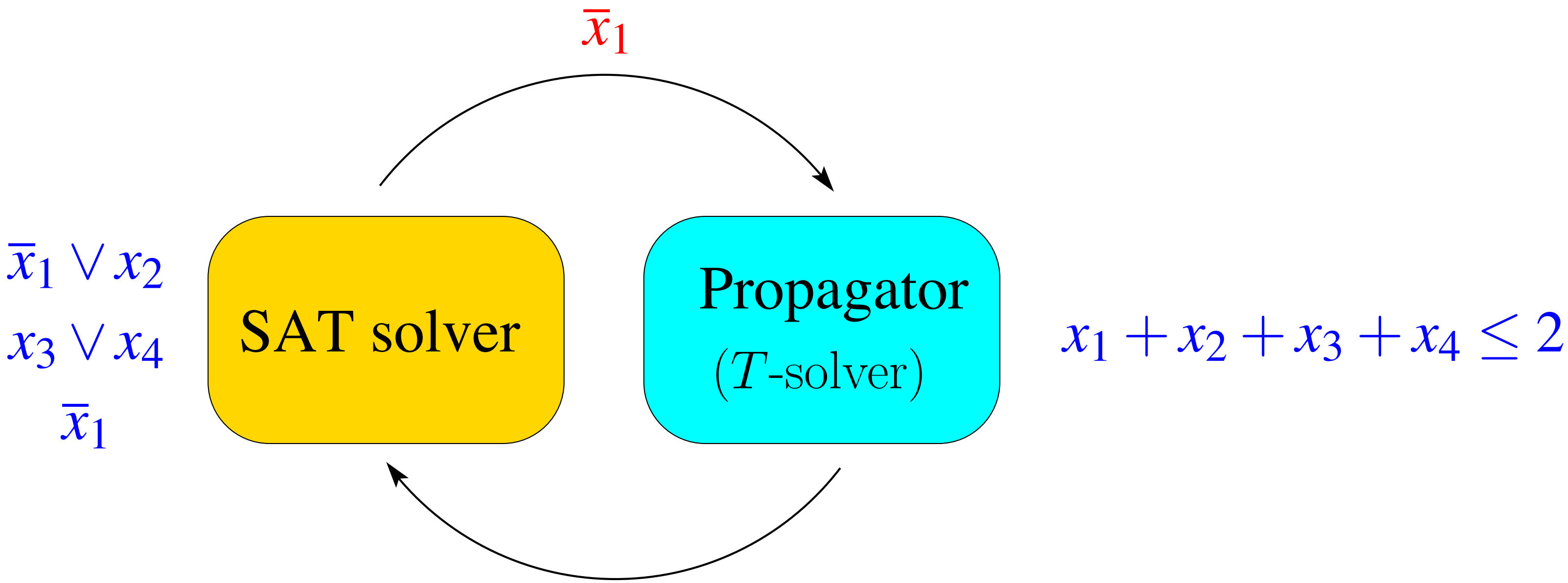
- Example: $\bar{x}_1 \vee x_2, \quad x_3 \vee x_4, \quad x_1 + x_2 + x_3 + x_4 \leq 2$



Learn

SMT/propagators (2)

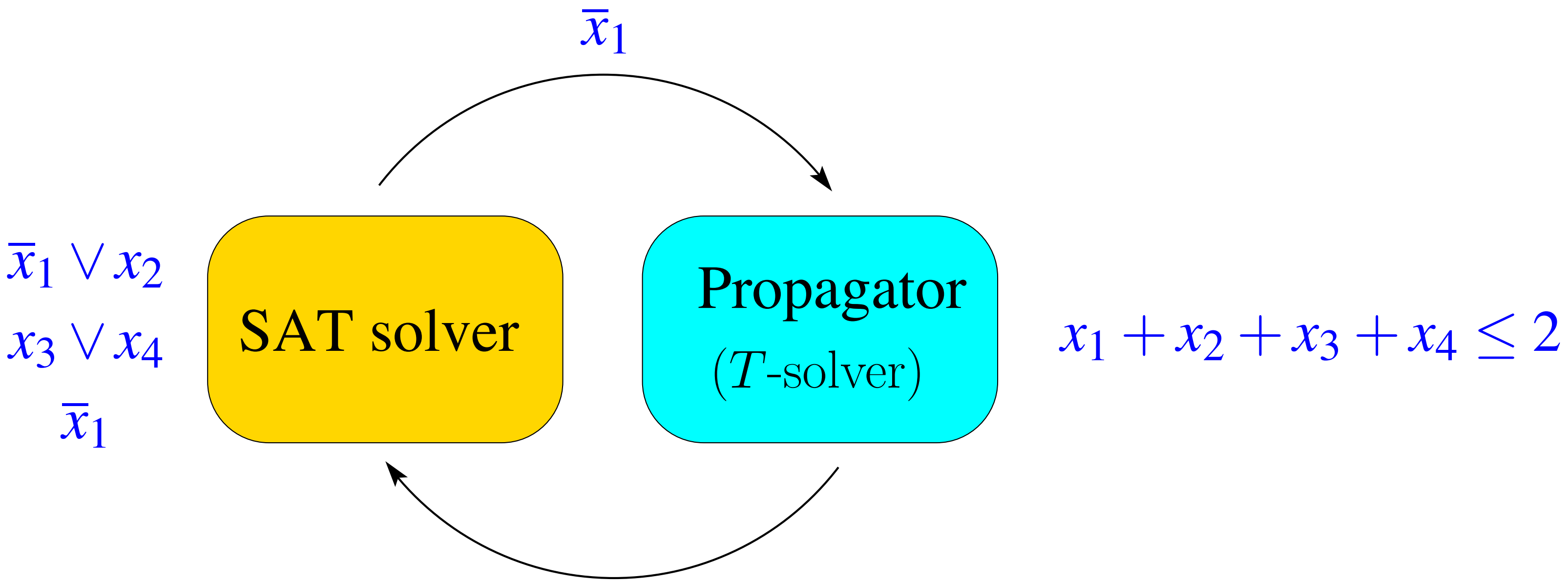
- Example: $\bar{x}_1 \vee x_2, \quad x_3 \vee x_4, \quad x_1 + x_2 + x_3 + x_4 \leq 2$



Backjump

SMT/propagators (2)

- Example: $\bar{x}_1 \vee x_2, \quad x_3 \vee x_4, \quad x_1 + x_2 + x_3 + x_4 \leq 2$



...

- SAT solver requires that the propagator:
  - Detects lits implied by partial assignment and constraint
  - Gives explanations of propagated lits for conflict analysis

Propagator for Cardinality Constraints

- Consider the constraint $x_1 + \ldots + x_n \leq k$
- Let us count no. of true literals, i.e., the size of $A_1 = \{i \mid x_i = 1\}$

Propagator for Cardinality Constraints

- Consider the constraint $x_1 + \ldots + x_n \leq k$
- Let us count no. of true literals, i.e., the size of $A_1 = \{i \mid x_i = 1\}$
- If $|A_1| \geq k$, let $E \subseteq A_1$ such that $|E| = k$
- For any $j \notin E$, literal $\overline{x}_j$ can be propagated
- Explanation: clause

$$\bigvee_{i_s \in E} \overline{x}_{i_s} \vee \overline{x}_j$$

Propagator for Cardinality Constraints

- Consider the constraint $x_1 + \ldots + x_n \leq k$
- Let us count no. of true literals, i.e., the size of $A_1 = \{i \mid x_i = 1\}$
- If $|A_1| \geq k$, let $E \subseteq A_1$ such that $|E| = k$
- For any $j \notin E$, literal $\overline{x}_j$ can be propagated
- Explanation: clause

$$\bigvee_{i_s \in E} \overline{x}_{i_s} \vee \overline{x}_j$$

- Note that explanations are the clauses of the naive encoding
- In general, SMT can be seen as lazily producing an encoding (without auxiliary variables)

Propagator for PB Constraints

- Consider the constraint $a_1 x_1 + \ldots + a_n x_n \leq k$ with $a_i \geq 0$
- Let us count the weighted sum $a_1 x_1 + \ldots + a_n x_n$ for true lits, i.e. in $A_1 = \{i \mid x_i = 1\}$

Propagator for PB Constraints

- Consider the constraint $a_1x_1 + \ldots + a_nx_n \leq k$ with $a_i \geq 0$
- Let us count the weighted sum $a_1x_1 + \ldots + a_nx_n$ for true lits, i.e. in $A_1 = \{i \mid x_i = 1\}$
- Assume there are $E \subseteq A_1$ and $j \notin E$ s.t. $a_j + \Sigma_{i \in E}\, a_i > k$
  Literal $\overline{x}_j$ can then be propagated
- Explanation: clause

$$\bigvee_{i_s \in E} \overline{x}_{i_s} \vee \overline{x}_j$$

Propagator for PB Constraints

- Consider the constraint $a_1x_1 + \ldots + a_nx_n \leq k$ with $a_i \geq 0$
- Let us count the weighted sum $a_1x_1 + \ldots + a_nx_n$ for true lits, i.e. in $A_1 = \{i \mid x_i = 1\}$
- Assume there are $E \subseteq A_1$ and $j \notin E$ s.t. $a_j + \Sigma_{i \in E}\, a_i > k$
  Literal $\overline{x}_j$ can then be propagated
- Explanation: clause

$$\bigvee_{i_s \in E} \overline{x}_{i_s} \vee \overline{x}_j$$

- Again, explanations correspond to clauses of a naive encoding (generalization of the case of cardinality constraints)

SMT and SAT Encodings Are Complementary

- Comparison of SMT / SAT encoding
  (using same underlying SAT solver Barcelogic)

Benchmark suite		SMT at least 1.5x faster	SAT enc. at least 1.5x faster
Tomography	(many card. cons.)	86.49%	5.93%
PB evaluation	(many PB/card. cons.)	43.49%	7.02%
RCPSP	(many PB cons.)	46.62%	0.69%
MSU4	(few card. cons.)	15.39%	39.37%
DES	(1 large card. cons.)	0.28%	92.06%

SMT and SAT Encodings Are Complementary

- Comparison of SMT / SAT encoding
  (using same underlying SAT solver Barcelogic)

Benchmark suite		SMT at least 1.5x faster	SAT enc. at least 1.5x faster
Tomography	(many card. cons.)	86.49%	5.93%
PB evaluation	(many PB/card. cons.)	43.49%	7.02%
RCPSP	(many PB cons.)	46.62%	0.69%
MSU4	(few card. cons.)	15.39%	39.37%
DES	(1 large card. cons.)	0.28%	92.06%

- **Can we get the best of both worlds?**

Related Work

Conflict-Directed Lazy Decomposition: [Abío & Stuckey, CP'12]

- Goal: to get the best of SAT encodings and SMT
- Basic idea:
  - Start off with a full SMT approach for each constraint
  - On the fly, partially encode only *active* parts of constraints
  - Active = would appear in explanations in conflict analysis

Related Work

Conflict-Directed Lazy Decomposition: [Abío & Stuckey, CP'12]

- Goal: to get the best of SAT encodings and SMT
- Basic idea:
  - Start off with a full SMT approach for each constraint
  - On the fly, partially encode only *active* parts of constraints
  - Active = would appear in explanations in conflict analysis
  - Thus:
    - Very active constraints end up completely encoded
    - Little active constraints are handled with SMT

Related Work

Conflict-Directed Lazy Decomposition: [Abío & Stuckey, CP'12]

- Goal: to get the best of SAT encodings and SMT
- Basic idea:
  - Start off with a full SMT approach for each constraint
  - On the fly, partially encode only *active* parts of constraints
  - Active = would appear in explanations in conflict analysis
  - Thus:
    - Very active constraints end up completely encoded
    - Little active constraints are handled with SMT
  - So far only available for encodings allowing partial decomposition (non-trivial):
    - cardinality network encoding for cardinality cons.
    - BDD encoding for PB cons.

Our Contribution: Pros of SMT (1)

- When is SMT effective?
- Often, while searching for solutions, constraints only
  - block the current solution candidate very few times (generate very few explanations)

    or

  - they do it almost always in the same way (generate few different explanations)
- Generating these explanations can be much more effective than encoding all constraints from the beginning

Our Contribution: Pros of SMT (2)

- Table below shows % of benchmark instances where at least half the constraints have a given % of repeated explanations
- Recall: in Tomography, PB evaluation, RCPSP better is SMT; in MSU4, DES better are SAT encodings

	Benchs with >50% of the constraints with this % of repeated explanations							
Suite	0-5%	5-10%	10-20%	20-40%	40-60%	60-80%	80-95%	95-100%
Tomography	0	0	0	0	0	0	100	0
PB evaluation	6.2	0	0	0	0	0.6	14.2	51.7
RCPSP	0	0	0	0	0	5.5	54.4	1.6
MSU4	66.9	11.0	19.9	12.4	2.8	0.9	0.2	0
DES	21.4	29.8	35.2	13.6	0	0	0	0

Our Contribution: Cons of SMT

- When is SMT not so effective?
- Sometimes some bottleneck constraints end up generating an exponential number of explanations,
  equivalent to a naive SAT encoding with no auxiliary variables

Our Contribution: Cons of SMT

- When is SMT not so effective?
- Sometimes some bottleneck constraints end up generating an exponential number of explanations,
  equivalent to a naive SAT encoding with no auxiliary variables
- Example: in $\begin{cases} x_1 + \ldots + x_n & < & n/2 \\ x_1 + \ldots + x_n & \geq & n/2 \end{cases}$

  SMT forced to produce all explanations of the form

  $$\overline{x_{i_1}} \vee \overline{x_{i_2}} \vee \ldots \vee \overline{x_{i_{n/2}}}$$

  and

  $$x_{i_1} \vee x_{i_2} \vee \ldots$$

Our Contribution: Cons of SMT

- When is SMT not so effective?
- Sometimes some bottleneck constraints end up generating an exponential number of explanations,
equivalent to a naive SAT encoding with no auxiliary variables
- Example: in $\begin{cases} x_1 + \ldots + x_n & < & n/2 \\ x_1 + \ldots + x_n & \geq & n/2 \end{cases}$

  SMT forced to produce all explanations of the form

$$\overline{x_{i_1}} \vee \overline{x_{i_2}} \vee \ldots \vee \overline{x_{i_{n/2}}}$$

  and

$$x_{i_1} \vee x_{i_2} \vee \ldots$$

- A polynomial-sized encoding for such a bottleneck constraint (possibly with auxiliary variables) may be better

Our Contribution: Getting the Best

- We implemented an SMT solver
  equipped with the ability of encoding on the fly:
  - cardinality constraints with cardinality networks
  - PB constraints with BDD's
- Encoding is irreversible (once a constraint is encoded, its propagator is off forever) and not partial (all or nothing)
- When to encode a constraint?
  When SMT is producing too many different explanations:

Our Contribution: Getting the Best

- We implemented an SMT solver
  equipped with the ability of encoding on the fly:
  - cardinality constraints with cardinality networks
  - PB constraints with BDD's
- Encoding is irreversible (once a constraint is encoded, its propagator is off forever) and not partial (all or nothing)
- When to encode a constraint?
  When SMT is producing too many different explanations:
  - If number of generated explanations gets close to (> 50 %) the number of clauses of the compact SAT encoding

Our Contribution: Getting the Best

- We implemented an SMT solver
  equipped with the ability of encoding on the fly:
  - cardinality constraints with cardinality networks
  - PB constraints with BDD's
- Encoding is irreversible (once a constraint is encoded, its propagator is off forever) and not partial (all or nothing)
- When to encode a constraint?
  When SMT is producing too many different explanations:
  - If number of generated explanations gets close to ($>$ 50 %) the number of clauses of the compact SAT encoding
  - More than X % of the explanations are new and more than Y explanations have already been generated;
    for us, $X = 70$ and $Y = 5000$

Experimental Results

	No. solved instances within < 600 secs.			
Suite	SMT	Encoding	LD	New
Tomography	2021	1932	1918	**2021**
PB evaluation	414	414	**416**	415
RCPSP	**272**	175	228	271
MSU4	4767	5677	5674	**5679**
DES	1452	**4228**	4019	4166

- No. of problems New solves close to best option for each suite
- Comparable,often better, results than lazy decomposition (LD) but much simpler and more widely applicable!

Conclusions and Future Work

- Paper accepted at CP'13. To appear soon.
- It is unnecessary to consider partial encodings:
  just encode on the fly the few really active constraints entirely
- The method is widely applicable: unlike lazy decomposition,
  not just for constraints for which partial encodings are known
- Future work:
  - Consider other kinds of constraints (alldifferent, ...)
  - Explore other adaptive strategies

Thank you!