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

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

CP 2013 17 September 2013 Uppsala, Sweden

To Encode or to Propagate? The Best Choice for Each Constraint in SAT - p. 1

#### **Overview of the talk**

- Motivation
- SAT encodings
- SMT
- SMT vs SAT Encodings
- Getting the best of both worlds
- Conclusions

#### **Motivation**

- **Goal:** use SAT for problems with complex constraints
  - In this work, cardinality and pseudo-Boolean constraints
- Applications:
  - Many in scheduling, timetabling, planning, MaxSAT, etc.
  - Also in constraint-based program analysis/synthesis

#### **Motivation**

- **Goal:** use SAT for problems with complex constraints
  - In this work, cardinality and pseudo-Boolean constraints
- Applications:
  - Many in scheduling, timetabling, planning, MaxSAT, etc.
  - Also in constraint-based program analysis/synthesis
- Why using SAT?
  - Success in some application areas (e.g. verification)
  - SAT tech outperforms other tools on real-world problems
  - 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
- Solution Variable  $a_{i,t}$  is true if task *i* is active at time *t*
- **Constraint:** There are no more active tasks than machines:

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

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

#### **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:

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

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

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

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

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

#### **Overview of the talk**

#### Motivation

- SAT encodings
- SMT
- SMT vs SAT Encodings
- Getting the best of both worlds
- Conclusions

#### **SAT Encodings**

- Encode the constraint *C* into a (CNF) formula *F* 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**

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

#### **SAT Encodings of Cardinality Constraints**

- Example: for a cardinality constraint  $\sum_{i=1}^{n} x_i \leq k 1$  we have:
  - Naive encoding
    - Variables: the same  $x_1, \ldots, x_n$
    - Clauses:  $\overline{x_{i_1}} \lor \ldots \lor \overline{x_{i_k}}$  for all  $1 \le i_1 < \ldots < i_k \le n$
    - This is  $\binom{n}{k}$  clauses!
  - Sorting network encoding (used in what follows)
     Build a circuit that sorts (say, decreasingly) *n* bits with inputs *x*<sub>1</sub>,...,*x<sub>n</sub>* and outputs new variables *y*<sub>1</sub>,...,*y<sub>n</sub>*
    - 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!
    - Refinements exists: see previous talk

#### **SAT Encodings of PB Constraints**

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

#### **SAT Encodings of PB Constraints**

- Several encodings exist
  - Unary/binary adder circuits
  - Sorting networks
  - BDD's
- Example of encoding  $2x_1 + 3x_2 + 5x_3 \le 6$  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  $\sum_{i=1}^{n} a_i x_i \leq k$  with BDD's:
  - Variables:  $x_1, \ldots, x_n$  and one for each node of the BDD
  - Clauses: standard is to use 6 clauses per node, but only one binary and one ternary clause per node suffice.
  - Linear number of clauses/variables in the size of the BDD

#### **SAT Encodings of PB Constraints (2)**

- In the encoding of  $\sum_{i=1}^{n} a_i x_i \leq k$  with BDD's:
  - Variables:  $x_1, \ldots, x_n$  and one for each node of the BDD
  - Clauses: standard is to use 6 clauses per node, but only one binary and one ternary clause per node suffice.
  - 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
  ... but this rarely occurs in practice

#### **SAT Encodings of PB Constraints (2)**

- In the encoding of  $\sum_{i=1}^{n} a_i x_i \leq k$  with BDD's:
  - Variables:  $x_1, \ldots, x_n$  and one for each node of the BDD
  - Clauses: standard is to use 6 clauses per node, but only one binary and one ternary clause per node suffice.
  - 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
  ... but this rarely occurs in practice
- In the following:
   BDD's used for encoding PB constraints (among most efficient encodings in practice)

#### **SAT Encodings**

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

#### **SAT Encodings**

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

#### **Overview of the talk**

- Motivation
- SAT encodings
- SMT
- SMT vs SAT Encodings
- Getting the best of both worlds
- Conclusions

#### **SMT / LCG**

Instead of eagerly encoding the constraint, deal with it lazily

#### **SMT / LCG**

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





#### ■ SAT: $\bar{x}_1 \lor x_2$ , $\bar{x}_1 \lor x_3$ , $x_1 \lor x_4$ , $\bar{x}_4 \lor x_3$ , $\bar{x}_3 \lor x_2$ Propagator: $x_1 + x_2 + x_3 + x_4 \le 2$



- SAT:  $\bar{x}_1 \lor x_2$ ,  $\bar{x}_1 \lor x_3$ ,  $x_1 \lor x_4$ ,  $\bar{x}_4 \lor x_3$ ,  $\bar{x}_3 \lor x_2$ Propagator:  $x_1 + x_2 + x_3 + x_4 \le 2$
- **SAT-solver finds model**  $x_1 x_2 x_3 \overline{x}_4$

- SAT:  $\bar{x}_1 \lor x_2$ ,  $\bar{x}_1 \lor x_3$ ,  $x_1 \lor x_4$ ,  $\bar{x}_4 \lor x_3$ ,  $\bar{x}_3 \lor x_2$ Propagator:  $x_1 + x_2 + x_3 + x_4 \le 2$
- **SAT-solver finds model**  $x_1 x_2 x_3 \overline{x}_4$
- Propagator refutes it and adds clause  $\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3$

- SAT:  $\bar{x}_1 \lor x_2$ ,  $\bar{x}_1 \lor x_3$ ,  $x_1 \lor x_4$ ,  $\bar{x}_4 \lor x_3$ ,  $\bar{x}_3 \lor x_2$ Propagator:  $x_1 + x_2 + x_3 + x_4 \le 2$
- **SAT-solver finds model**  $x_1 x_2 x_3 \overline{x}_4$
- Propagator refutes it and adds clause  $\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3$
- **SAT:**  $\overline{x}_1 \lor x_2$ ,  $\overline{x}_1 \lor x_3$ ,  $x_1 \lor x_4$ ,  $\overline{x}_4 \lor x_3$ ,  $\overline{x}_3 \lor x_2$  $\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3$ **Propagator:**  $x_1 + x_2 + x_3 + x_4 \le 2$

- SAT:  $\bar{x}_1 \lor x_2$ ,  $\bar{x}_1 \lor x_3$ ,  $x_1 \lor x_4$ ,  $\bar{x}_4 \lor x_3$ ,  $\bar{x}_3 \lor x_2$ Propagator:  $x_1 + x_2 + x_3 + x_4 \le 2$
- **SAT-solver finds model**  $x_1 x_2 x_3 \overline{x}_4$
- **Propagator refutes it and adds clause**  $\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3$
- **SAT:**  $\overline{x}_1 \lor x_2$ ,  $\overline{x}_1 \lor x_3$ ,  $x_1 \lor x_4$ ,  $\overline{x}_4 \lor x_3$ ,  $\overline{x}_3 \lor x_2$  $\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3$ **Propagator:**  $x_1 + x_2 + x_3 + x_4 \le 2$
- **SAT-solver finds model**  $\overline{x}_1 x_2 x_3 x_4$

- SAT:  $\bar{x}_1 \lor x_2$ ,  $\bar{x}_1 \lor x_3$ ,  $x_1 \lor x_4$ ,  $\bar{x}_4 \lor x_3$ ,  $\bar{x}_3 \lor x_2$ Propagator:  $x_1 + x_2 + x_3 + x_4 \le 2$
- **SAT-solver finds model**  $x_1 x_2 x_3 \overline{x}_4$
- Propagator refutes it and adds clause  $\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3$
- **SAT:**  $\overline{x}_1 \lor x_2$ ,  $\overline{x}_1 \lor x_3$ ,  $x_1 \lor x_4$ ,  $\overline{x}_4 \lor x_3$ ,  $\overline{x}_3 \lor x_2$  $\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3$ **Propagator:**  $x_1 + x_2 + x_3 + x_4 \le 2$
- **SAT-solver finds model**  $\overline{x}_1 x_2 x_3 x_4$
- Propagator refutes it and adds clause  $\overline{x}_2 \lor \overline{x}_3 \lor \overline{x}_4$

- SAT:  $\bar{x}_1 \lor x_2$ ,  $\bar{x}_1 \lor x_3$ ,  $x_1 \lor x_4$ ,  $\bar{x}_4 \lor x_3$ ,  $\bar{x}_3 \lor x_2$ Propagator:  $x_1 + x_2 + x_3 + x_4 \le 2$
- **SAT-solver finds model**  $x_1 x_2 x_3 \overline{x}_4$
- **Propagator refutes it and adds clause**  $\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3$
- **SAT:**  $\overline{x}_1 \lor x_2$ ,  $\overline{x}_1 \lor x_3$ ,  $x_1 \lor x_4$ ,  $\overline{x}_4 \lor x_3$ ,  $\overline{x}_3 \lor x_2$  $\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3$ **Propagator:**  $x_1 + x_2 + x_3 + x_4 \le 2$
- **SAT-solver finds model**  $\overline{x}_1 x_2 x_3 x_4$
- **Propagator refutes it and adds clause**  $\overline{x}_2 \lor \overline{x}_3 \lor \overline{x}_4$

**SAT:** 
$$\overline{x}_1 \lor x_2$$
,  $\overline{x}_1 \lor x_3$ ,  $x_1 \lor x_4$ ,  $\overline{x}_4 \lor x_3$ ,  $\overline{x}_3 \lor x_2$   
 $\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3$ ,  $\overline{x}_2 \lor \overline{x}_3 \lor \overline{x}_4$   
**Propagator:**  $x_1 + x_2 + x_3 + x_4 \le 2$ 

- SAT:  $\bar{x}_1 \lor x_2$ ,  $\bar{x}_1 \lor x_3$ ,  $x_1 \lor x_4$ ,  $\bar{x}_4 \lor x_3$ ,  $\bar{x}_3 \lor x_2$ Propagator:  $x_1 + x_2 + x_3 + x_4 \le 2$
- **SAT-solver finds model**  $x_1 x_2 x_3 \overline{x}_4$
- **Propagator refutes it and adds clause**  $\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3$
- **SAT:**  $\overline{x}_1 \lor x_2$ ,  $\overline{x}_1 \lor x_3$ ,  $x_1 \lor x_4$ ,  $\overline{x}_4 \lor x_3$ ,  $\overline{x}_3 \lor x_2$  $\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3$ **Propagator:**  $x_1 + x_2 + x_3 + x_4 \le 2$
- **SAT-solver finds model**  $\overline{x}_1 x_2 x_3 x_4$
- **Propagator refutes it and adds clause**  $\overline{x}_2 \lor \overline{x}_3 \lor \overline{x}_4$

**SAT:** 
$$\overline{x}_1 \lor x_2$$
,  $\overline{x}_1 \lor x_3$ ,  $x_1 \lor x_4$ ,  $\overline{x}_4 \lor x_3$ ,  $\overline{x}_3 \lor x_2$   
 $\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3$ ,  $\overline{x}_2 \lor \overline{x}_3 \lor \overline{x}_4$   
**Propagator:**  $x_1 + x_2 + x_3 + x_4 \le 2$ 

SAT-solver says UNSATISFIABLE

- In practice, propagators also invoked on partial assignments
- This allows to propagate values for unassigned lits
- Propagators for cardinality and PB constraints amount to (incremental) counters
- Explanations of inconsistency (and also of propagations) are clauses of the naive encoding (no auxiliary variables)

#### **Overview of the talk**

- Motivation
- SAT encodings
- SMT

## SMT vs SAT Encodings

- Getting the best of both worlds
- Conclusions

#### **SMT and SAT Encodings Are Complementary**

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

Benchmark suite		SMT at least	SAT enc. at least
		1.5x faster	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	SAT enc. at least
		1.5x faster	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?

#### When is SMT a good choice?

- When, while searching for solutions, constraints only block the current solution candidate very few times (generate very few explanations)
- Generating these explanations can be much more effective than encoding all constraints from the beginning
- This is a well-known fact from the early SMT systems

#### **Cons of SMT**

#### When is SMT a bad choice?

Sometimes some bottleneck constraints end up generating an exponential number of explanations, equivalent to a naive SAT encoding with no auxiliary variables

#### Cons of SMT

#### When is SMT a bad choice?

- Sometimes some **bottleneck constraints** end up generating an exponential number of explanations, equivalent to a naive SAT encoding with no auxiliary variables

• Example: in  $\begin{cases} x_1 + \ldots + x_n < n/2 \\ x_1 + \ldots + x_n \ge n/2 \end{cases}$ 

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

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

#### **Cons of SMT**

#### When is SMT a bad choice?

Sometimes some bottleneck constraints end up generating an exponential number of explanations, equivalent to a naive SAT encoding with no auxiliary variables

$$\begin{array}{rcl} x_1 + \ldots + x_n &< n/2 \\ x_1 + \ldots + x_n &\geq n/2 \end{array}$$

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

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

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

### Cons of SMT (2)



#### **Overview of the talk**

- Motivation
- SAT encodings
- SMT
- SMT vs SAT Encodings

## Getting the best of both worlds

Conclusions

#### **Getting the Best of Both Worlds**

- IDEA: implement an SMT solver equipped with the ability of encoding on the fly:
  - cardinality constraints encoded via cardinality networks
  - PB constraints encoded via BDDs
- Encoding is irreversible (once a constraint is encoded, its propagator is off forever) and not partial (all or nothing)
- When to encode a constraint?

First attempt: only encode active constraints (lots of explanations generated)

#### Getting the best of both worlds (2)

#### Remember, Green: SMT wins, Red: Encoding wins

	Perc. of benchs with this perc. of low-act. constr.							
Suite	0-5%	5-10%	10-20%	20-40%	40-60%	60-80%	80-95%	95-100%
Tomography	100	0	0	0	0	0	0	0
PB evaluation	54	21.6	20.5	0.6	1.1	0.6	1.7	20.5
RCPSP	0	0	2.2	13.2	51.1	31.3	2.2	0
MSU4	74.6	0	0	0	24.9	0.5	0	0
DES	99.9	0	0	0	0	0	0	0.1

- SMT should be the winner only if most constraints are inactive
- This does not explain at all the behavior on Tomography suite
- Whats is happening?

#### Getting the best of both worlds (3)

Remember, Green: SMT wins, Red: Encoding wins

Table below shows % of benchmark instances where at least half the constraints have a given % of repeated explanations

	% 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

- According to the previous table, in Tomography, MSU4 and DES, constraints produced lots of explanation.
- But in Tomography very few different explanations were produced: whole naive encoding not generated

#### Getting the best of both worlds (4)

- 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 likely to produce the whole naive encoding. In our implementation if one of the following conditions holds:

#### Getting the best of both worlds (4)

- 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 likely to produce the whole naive encoding. In our implementation if one of the following conditions holds:
  - If number of different explanations gets close to (> 50 %) the number of clauses of the compact SAT encoding

#### Getting the best of both worlds (4)

- 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 likely to produce the whole naive encoding. In our implementation if one of the following conditions holds:
  - If number of different 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

#### **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
  - **J** 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
  - **J** 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.

#### **Experimental Results**

	<b>No. solved instances within</b> < 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!

#### **Overview of the talk**

- Motivation
- SAT encodings
- SMT
- SMT vs SAT Encodings
- Getting the best of both worlds
- Conclusions

#### **Conclusions and Future Work**

- We can get the best of SMT and Encodings in a single tool
- 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!