Short Presentation

Saurabh Joshi

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Venue: Second SAT+SMT School, Mysore

December 7, 2017

Generalized Totalizer Encoding for Pseudo-Boolean Constraints

Saurabh Joshi⋆, Ruben Martins⋆, Vasco Manquinho†

⋆University of Oxford, UK †INESC-ID / IST, University of Lisbon, Portugal

CP 2015

Types of Constraints over Boolean variables

Clauses: $l_1 \vee l_2 \vee \ldots \vee l_n$

$l_1 \vee l_2 \vee l_3 \vee l_4$

Types of Constraints over Boolean variables

Clauses: $l_1 \vee l_2 \vee \ldots \vee l_n$

$$l_1 \vee l_2 \vee l_3 \vee l_4$$

Cardinality: $\sum_{j=1}^{n} l_j \leq k$

$$l_1 + l_2 + l_3 + l_4 + l_5 \leq 3$$

Types of Constraints over Boolean variables

Clauses: $l_1 \vee l_2 \vee \ldots \vee l_n$

$l_1 \vee l_2 \vee l_3 \vee l_4$

Cardinality: $\sum_{j=1}^{n} l_j \leq k$

$l_1 + l_2 + l_3 + l_4 + l_5 \leq 3$

Pseudo-Boolean: $\sum_{j=1}^{n} w_j . l_j \leq k$

$3l_1 + 2l_2 + 5l_3 \leq 5$

Pseudo-Boolean Constraints

Pseudo-Boolean constraints are used in several application domains:

- Computational Biology
- Upgradeability of Software Systems
- Scheduling
- Automated Test Pattern Generation
- ...
  Problem instances available at the Pseudo-Boolean Solver Evaluation website (http://pbeva.computational-logic.org/).

Pseudo-Boolean Constraints

Pseudo-Boolean constraints are used in several application domains:

- Computational Biology
- Upgradeability of Software Systems
- Scheduling
- Automated Test Pattern Generation
- ...
  Problem instances available at the Pseudo-Boolean Solver Evaluation website (http://pbeva.computational-logic.org/).

Also arise in different algorithmic frameworks, e.g. weighted Maximum Satisfiability (MaxSAT) algorithms.

How to deal with Pseudo-Boolean Constraints

There are two typical approaches in solving formulas with Pseudo-Boolean constraints.

1. Use a generalized SAT solver to deal directly with Pseudo-Boolean Constraints
   - pueblo, bsolo, wbo, sat4jPB

How to deal with Pseudo-Boolean Constraints

There are two typical approaches in solving formulas with Pseudo-Boolean constraints.

1. Use a generalized SAT solver to deal directly with Pseudo-Boolean Constraints
   - pueblo, bsolo, wbo, sat4jPB
2. Encode the Pseudo-Boolean Constraints into CNF and use an off-the-shelf SAT solver
   - minisat+, oree

How to deal with Pseudo-Boolean Constraints

There are two typical approaches in solving formulas with Pseudo-Boolean constraints.

1. Use a generalized SAT solver to deal directly with Pseudo-Boolean Constraints
   - pueblo, bsolo, wbo, sat4jPB
2. Encode the Pseudo-Boolean Constraints into CNF and use an off-the-shelf SAT solver
   - minisat+, oree

The latter approach results in a larger formula, but able to take advantage of powerful SAT solver technology.

Encodings of Pseudo-Boolean Constraints

Several encodings of Pseudo-Boolean constraints into CNF have been proposed that use additional auxiliary variables and clauses:

- BDD [Eén and Sorensson, JSAT 2006]
- Sorters
- Adders

Encodings of Pseudo-Boolean Constraints

Several encodings of Pseudo-Boolean constraints into CNF have been proposed that use additional auxiliary variables and clauses:

- BDD [Eén and Sorensson, JSAT 2006]
- Sorters
- Adders
- WatchDog [Bailleux et al. SAT 2009]
- Binary Merger [Manthey et al. KI 2014]

Encodings of Pseudo-Boolean Constraints

Several encodings of Pseudo-Boolean constraints into CNF have been proposed that use additional auxiliary variables and clauses:

- BDD [Eén and Sorensson, JSAT 2006]
- Sorters
- Adders
- WatchDog [Bailleux et al. SAT 2009]
- Binary Merger [Manthey et al. KI 2014]
- Sequential Weighted Counter [Holldobler et al. KI 2012]

Generalized Totalizer Encoding (GTE)

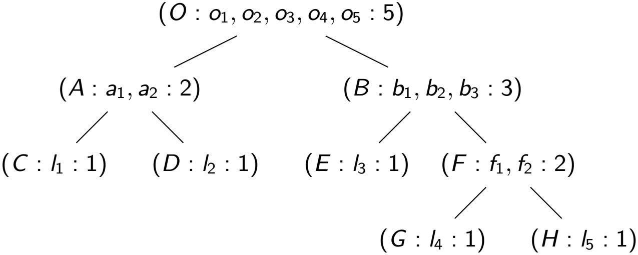
- New encoding of Pseudo-Boolean Constraints into CNF
- Not dependent on the magnitude of the variable coefficients
- Preserves Arc-Consistency
- Exponential in the worst case, but compact and effective in practice
- Based on the Totalizer encoding [Bailleux and Boufkhad, CP 2003]

Totalizer Encoding for Cardinality Constraints

Totalizer Encoding

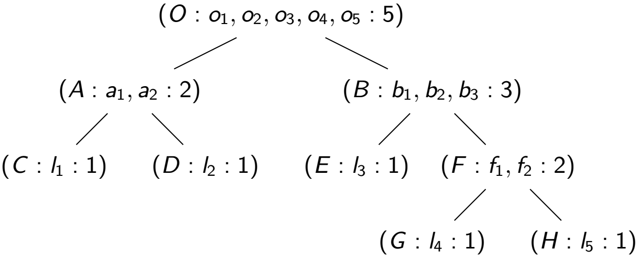
- CNF encoding for cardinality constraints $\sum_{i=1}^{n} l_i \leq k$
- Count in unary how many of the n literals $(l_1 \dots l_n)$ is set to *true*
- $O(n\ lg\ n)$ new variables
- $O(n^2)$ new clauses
  - Can be improved to $O(n\ k)$

Totalizer Encoding for Cardinality Constraints



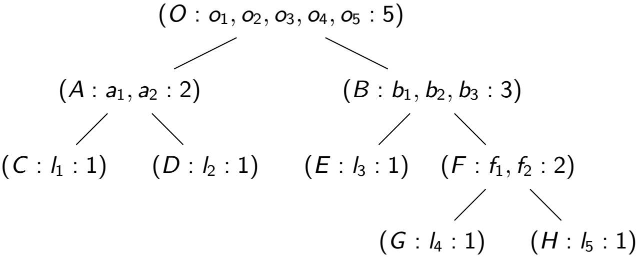
- Visualize the encoding as a tree
  - Each node is $(\textit{name} : \textit{variables} : \textit{sum})$
  - Literals are at the leaves
  - Each node counts in unary how many leaves are set to *true* in its subtree
  - Example: if at least 2 of the leaves (l_3, l_4, l_5) are assigned to *true* then b_2 must be *true*.

Totalizer Encoding for Cardinality Constraints



- Root node has the output variables $(o_1 \dots o_5)$ that count how many literals are set to *true*
- To encode $l_1 + l_2 + l_3 + l_4 + l_5 \leq 3$ just set $o_4 = 0$ and $o_5 = 0$

Totalizer Encoding for Cardinality Constraints



- Root node has the output variables ($o_1 \dots o_5$) that count how many literals are set to *true*
- To encode $l_1 + l_2 + l_3 + l_4 + l_5 \leq 3$ just set $o_4 = 0$ and $o_5 = 0$
- For this constraint, variable o_5 is not necessary (k-simplification technique)

GTE for Pseudo-Boolean Constraints

- The goal of the Generalized Totalizer Encoding (GTE) is also to account for the possible values of the left-hand side

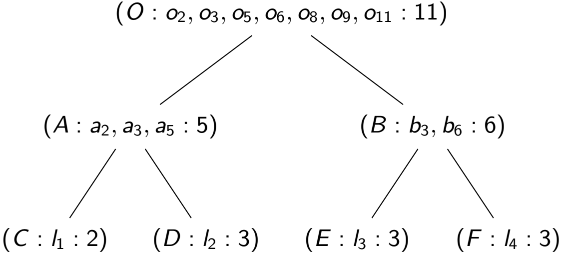
GTE for Pseudo-Boolean Constraints

- The goal of the Generalized Totalizer Encoding (GTE) is also to account for the possible values of the left-hand side
- GTE only considers the possible sums generated from the weights in the constraint

GTE for Pseudo-Boolean Constraints

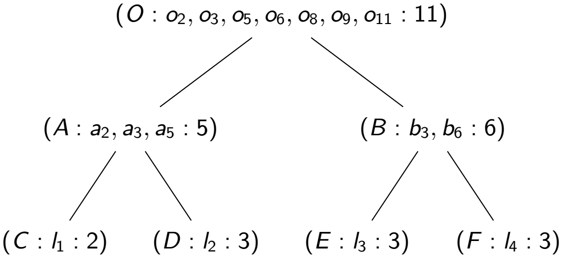
- The goal of the Generalized Totalizer Encoding (GTE) is also to account for the possible values of the left-hand side
- GTE only considers the possible sums generated from the weights in the constraint
- For example, in $2l_1 + 3l_2 + 3l_3 + 3l_4 \leq 5$ it is not possible for the weighted sum to have value 1, 4 or 7

GTE for Pseudo-Boolean Constraints



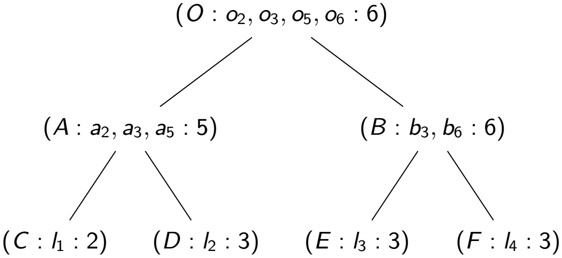
- The GTE encoding is also a tree with literals at the leaves
  - Each node is (*name* : *variables* : *sum*)
  - *sum* represents the maximum possible weighted sum of the subtree rooted at that node
  - A node variable a_w represents if there is a weighted sum of the subtree rooted at that node equal to w
  - Example: if one of the leaves (l_3, l_4) is assigned to *true* then b_3 must be *true*

GTE for Pseudo-Boolean Constraints



- Root node has the output variables $(o_2, o_3, o_5, o_6, o_8, o_9, o_{11})$ that encode the possible value of the weighted sums of the subtree
- To encode $2l_1 + 3l_2 + 3l_3 + 3l_4 \leq 5$ just set variables o_6, o_8, o_9 and o_{11} to *false*

GTE for Pseudo-Boolean Constraints



- Root node has the output variables $(o_2, o_3, o_5, o_6, o_8, o_9, o_{11})$ that encode the possible value of the weighted sums of the subtree
- To encode $2l_1 + 3l_2 + 3l_3 + 3l_4 \leq 5$ just set variables o_6, o_8, o_9 and o_{11} to *false*
- For this constraint, variables o_8, o_9 and o_{11} are not necessary (k-simplification technique)

GTE for Pseudo-Boolean Constraints - Analysis

- GTE maintains arc-consistency

GTE for Pseudo-Boolean Constraints - Analysis

- GTE maintains arc-consistency
- GTE does not depend on the magnitude of the weights

GTE for Pseudo-Boolean Constraints - Analysis

- GTE maintains arc-consistency
- GTE does not depend on the magnitude of the weights
- The number of variables depends solely on the unique weighted sums can be generated from the constraint weights

GTE for Pseudo-Boolean Constraints - Analysis

- GTE maintains arc-consistency
- GTE does not depend on the magnitude of the weights
- The number of variables depends solely on the unique weighted sums can be generated from the constraint weights
- In the worst case, the GTE encoding is exponential
    - $\sum_{j=1}^{n} 2^{j-1} l_j \leq k$

GTE for Pseudo-Boolean Constraints - Analysis

- GTE maintains arc-consistency
- GTE does not depend on the magnitude of the weights
- The number of variables depends solely on the unique weighted sums can be generated from the constraint weights
- In the worst case, the GTE encoding is exponential
  - $\sum_{j=1}^{n} 2^{j-1} l_j \leq k$
- If the number of distinct weighted sum combinations is low, GTE should perform better

Experimental Results

GTE was implemented on top of the open source PBLib library and compared with a large set of Pseudo-Boolean encodings available at PBLib.

- BDD
- Sorters
- Adders
- WatchDog
- Binary Merger
- Sequential Weighted Counter

GTE encoding should be available soon in PBLib.
http://tools.computational-logic.org/content/pblib.php

Experimental Results

Benchmarks:

- Decision instances from the 2012 Pseudo-Boolean Evaluation (PB'12)
- Pedigree instances used in MaxSAT Evaluation

Experimental setup:

- AMD Opteron 6276 processors (2.3 GHz) running Fedora Core 18;
- Timeout: 1,800 seconds
- Memory Limit: 16GB

Experimental Results

Characteristics of Pseudo-Boolean benchmarks

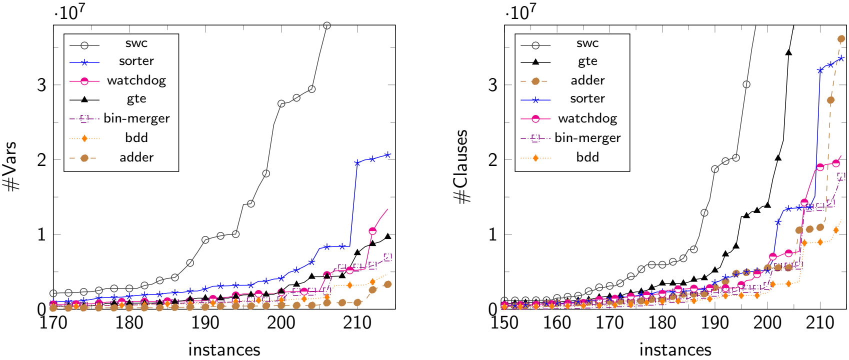
Benchmark	#PB	#lits	k	max w_i	$\sum w_i$	#diff w_i
PB'12	164.31	32.25	27.94	12.55	167.14	6.72
pedigree	1.00	10,794.13	11,106.69	456.28	4,665,237.38	2.00

Number of solved instances

Benchmark	Result	sorter	swc	adder	watchdog	bin-merger	bdd	gte
PB'12	SAT	72	74	73	79	79	**81**	**81**
(214)	UNSAT	74	77	83	**85**	**85**	84	84
pedigree	SAT	2	7	6	25	43	82	**83**
(172)	UNSAT	0	7	6	23	35	72	**75**
Total	SAT/UNSAT	146	165	172	212	242	319	**323**

Experimental Results

Encoding growth on PB'12 Benchmarks

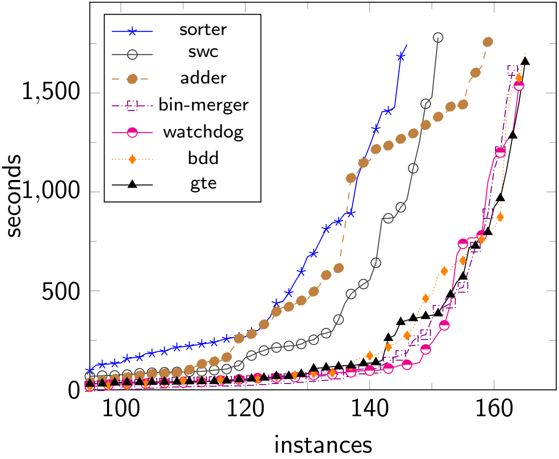


(a) # Variables on PB'12 benchmarks

(b) # Clauses on PB'12 benchmarks

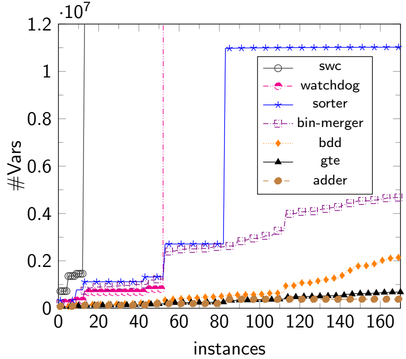
Experimental Results

Running times on PB'12 Benchmarks

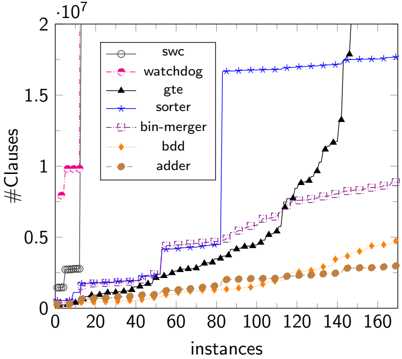


Experimental Results

Encoding growth on pedigree Benchmarks



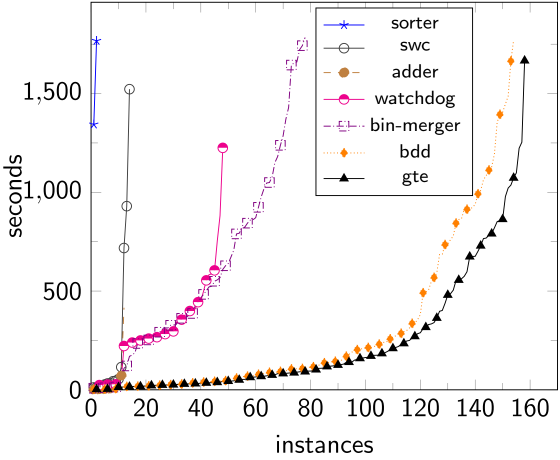
(a) # Variables on pedigree benchmarks



(b) # Clauses on pedigree benchmarks

Experimental Results

Running times on pedigree Benchmarks



Conclusions

- GTE: A new encoding of Pseudo-Boolean Constraints into CNF
  - Maintains Arc-Consistency
  - Does not depend on the magnitude of the weights
  - More effective when the number of distinct weights is small
- Results show that GTE is competitive with state of the art encodings for different sets of benchmark instances
  - 3rd rank in Pseudo Boolean 2016 competition

Future Work

- GTE will become available on the open source PBLib library ?
- GTE ~~is being~~ has been integrated into the open source OPEN-WBO solver
    - Available for decision and optimization problems (MaxSAT and Pseudo-Boolean Optimization)
    - New release of OPEN-WBO ~~should be~~ is available ~~soon~~ now
    - https://github.com/sat-group/open-wbo