PBSugar Compiling Pseudo-Boolean Constraints to SAT with Order Encoding

Naoyuki Tamura, Mutsunori Banbara, Takehide Soh

Kobe University

July 26, 2013
3rd CSPSAT2 Meeting in Sapporo

Outline of this talk

We implemented a Pseudo-Boolean (PB for short) solver named PBSugar by applying the order encoding technique to PB constraints.

Contents

Order Encoding

In [Tamura *et al.* 2006], we proposed the encoding to be applied for finite linear CSPs (Constraint Satisfaction Problems).

- It is implemented in Sugar solver which won in Global categories of 2008 and 2009 CSP solver competitions.
- Boolean variable $p(x \geq c)$ (alternatively $p(x \leq c)$) is assigned for each integer variable x and each domain value c [1].
- Bounds consistency is maintained by Unit Propagation.
- It is the only existing encoding translating tractable CSP to tractable SAT [Petke and Jeavons 2011].
- It showed a good performance for a wide range of problems [2].

[1] [Crawford and Baker 1994] [Bailleux *et al.* 2003] [Ansótegui and Manyà 2004] [Gent and Nightingale 2004] [Ohrimenko *et al.* 2009] [Metodi *et al.* 2011]

[2] Shop scheduling [Inoue *et al.* 2006] [Koshimura *et al.* 2010] 2D packing [Soh *et al.* 2009] Covering arrays [Banbara *et al.* 2010]

Order Encoding: Original Version (1/3)

First, we introduce the order encoding of the original version [Tamura *et al.* 2006].

Let x be an integer variable where $Dom(x) = \{l \mathinner{..} u\}$.

Boolean variables used for $x \in \{l \mathinner{..} u\}$

We introduce $u - l - 1$ Boolean variables for x:

$$p(x \geq c) \quad (\text{for each } c \in \{l+1 \mathinner{..} u\})^{3}$$

Axiom clauses used for $x \in \{l \mathinner{..} u\}$

The axiom clauses enforce x to take exactly one value in $\{l \mathinner{..} u\}$.

$$p(x \geq c) \lor \neg p(x \geq c+1) \qquad (\text{for each } c \in \{l+1 \mathinner{..} u-1\})$$

[3] $p(x \leq c)$ is used instead in [Tamura *et al.* 2006] and Sugar system.

Order Encoding: Original Version (2/3)

Let C be a linear arithmetic constraint $\sum_{i=1}^{n} a_i x_i \geq c$ where each a_i is a non-zero integer constant, c is an integer constant, and each x_i is an integer variable with the domain $Dom(x_i)$.

Original Order Encoding of $\sum_{i=1}^{n} a_i x_i \geq c$

$$\sum_{i=1}^{n} a_i x_i \geq c \iff \bigwedge_{c_i} \bigvee_{i=1}^{n} (a_i x_i \geq c_i)^{\#}$$

- Parameters c_i's range over integers satisfying $\sum_{i=1}^{n} c_i = c + n - 1$ and $\inf(a_i x_i) \leq c_i \leq \sup(a_i x_i) + 1$.
- The translation $(ax \geq c)^{\#}$ is defined by:

$$(ax \geq c)^{\#} = \begin{cases} x \geq \lceil c/a \rceil & (\text{when } a > 0) \\ \neg(x \geq \lfloor c/a \rfloor + 1) & (\text{when } a < 0) \end{cases}$$

Order Encoding: Original Version (3/3)

Straightforward implementation of the original version may generate redundant clauses.

Consider $2x_1 + 3x_2 \geq 20$ where $Dom(x_1) = Dom(x_2) = \{0 \mathinner{\ldotp\ldotp} 8\}$. The parameters c_1 and c_2 range over $\{0 \mathinner{\ldotp\ldotp} 17\}$ and $\{0 \mathinner{\ldotp\ldotp} 25\}$ respectively satisfying $c_1 + c_2 = 21$.

- We obtain 18 clauses including one valid clause.
- Some clauses are appeared twice:

$$\begin{aligned} (2x_1 \geq 1)^{\#} \vee (3x_2 \geq 20)^{\#} &\longrightarrow p(x_1 \geq 1) \vee p(x_2 \geq 7) \\ (2x_1 \geq 2)^{\#} \vee (3x_2 \geq 19)^{\#} &\longrightarrow p(x_1 \geq 1) \vee p(x_2 \geq 7) \end{aligned}$$

- Some clauses can be derived from others:

$$\begin{aligned} (2x_1 \geq 4)^{\#} \vee (3x_2 \geq 17)^{\#} &\longrightarrow p(x_1 \geq 2) \vee p(x_2 \geq 6) \\ (2x_1 \geq 5)^{\#} \vee (3x_2 \geq 16)^{\#} &\longrightarrow p(x_1 \geq 3) \vee p(x_2 \geq 6) \end{aligned}$$

Order Encoding: Optimized Version (1/4)

Now, we revise the original version.

First, we allow non-contiguous domains for variables. Let x be an integer variable where $Dom(x) = \{d_0, d_1, \ldots, d_m\}$ $(d_0 < d_1 < \cdots < d_m)$.

Boolean variables used for $x \in \mathbf{Dom}(x)$

We introduce m Boolean variables:

$$p(x \geq d_j) \quad (\text{for each } j \in \{1 \mathinner{..} m\})$$

Axiom clauses used for $x \in \mathbf{Dom}(x)$

The axiom clauses enforce x to have exactly one value in $Dom(x)$.

$$p(x \geq d_j) \vee \neg p(x \geq d_{j+1}) \qquad (\text{for each } j \in \{1 \mathinner{..} m-1\})$$

Order Encoding: Optimized Version (2/4)

We can show the following proposition.

Proposition

$$\sum_{i=1}^{n} a_i x_i \geq c \iff$$

$$\begin{cases} (x_1 \geq \lceil c/a_1 \rceil) & (n=1, a_1>0) \\ \neg(x_1 \geq \lfloor c/a_1 \rfloor + 1) & (n=1, a_1<0) \\ \displaystyle\bigwedge_{d \in Dom(x_1)} \left((x_1 \geq d+1) \lor \sum_{i=2}^{n} a_i x_i \geq c - a_1 d \right) & (n \geq 2, a_1>0) \\ \displaystyle\bigwedge_{d \in Dom(x_1)} \left(\neg(x_1 \geq d) \lor \sum_{i=2}^{n} a_i x_i \geq c - a_1 d \right) & (n \geq 2, a_1<0) \end{cases}$$

- Now, the big conjunction iterates over the values of x_i instead of $a_i x_i$ in the original version.

Order Encoding: Optimized Version (3/4)

We can derive a recursive encoding algorithm from the proposition. However, there remains some room for further optimization.

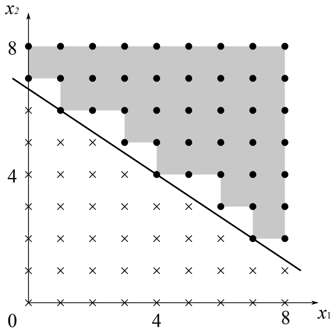
- First, by sorting the coefficients in the descending order, the number of redundant (derived) clauses will be reduced.
- Second, we can remove clauses when the recursive part $\sum_{i=2}^{n} a_i x_i \geq c - a_1 d$ is valid.
- Third, we can remove the literal when the recursive part is inconsistent. Also we only need one clause containing such part because other clauses are derived from it.

These ideas have been implemented in Sugar solver.

Order Encoding: Optimized Version (4/4)

Consider the same example again: $2x_1 + 3x_2 \geq 20$ where $Dom(x_1) = Dom(x_2) = \{0\,..\,8\}$.

- We obtain 6 clauses (instead of 18 in the original version) by applying the optimized algorithm.



$$
\begin{array}{c}
p(x_2 \geq 2) \\
p(x_2 \geq 3) \lor p(x_1 \geq 7) \\
p(x_2 \geq 4) \lor p(x_1 \geq 6) \\
p(x_2 \geq 5) \lor p(x_1 \geq 4) \\
p(x_2 \geq 6) \lor p(x_1 \geq 3) \\
p(x_2 \geq 7) \lor p(x_1 \geq 1)
\end{array}
$$

Boolean Cardinality Constraints

Boolean Cardinality constraints (BC constraints for short) are constraints of the form $x_1 + x_2 + \cdots + x_n \# k$ where

- k is an integer constant,
- x_i's are literals taking a value 0 (false) or 1 (true), and
- $\#$ is one of the relations $\{\geq, \leq, =, \neq\}$.

$x_1 + x_2 + \cdots + x_n$ and k are called the LHS and RHS of the constraint respectively.

Without loss of generality, we can assume the relation $\#$ is $\leq$.

$$\begin{array}{lcl} \sum_{i=1}^{n} x_i \geq k & \Longleftrightarrow & \sum_{i=1}^{n} \overline{x_i} \leq n - k \\ \sum_{i=1}^{n} x_i = k & \Longleftrightarrow & \sum_{i=1}^{n} x_i \leq k \ \wedge \ \sum_{i=1}^{n} x_i \geq k \\ \sum_{i=1}^{n} x_i \neq k & \Longleftrightarrow & \sum_{i=1}^{n} x_i \leq k - 1 \ \vee \ \sum_{i=1}^{n} x_i \geq k + 1 \end{array}$$

Order Encoding of BC (1/3)

Now, we consider to apply the optimized order encoding to BC constraint $\sum_{i=1}^{n} x_i \leq k$.

$(k+1)$-wise tuple encoding

When the order encoding is directly applied to the BC constraint, we will obtain $(k+1)$-wise tuple encoding (pairwise when $k = 1$).

$$\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}} \qquad (1 \leq i_1 < \cdots < i_{k+1} \leq n)$$

Order Encoding of BC (2/3)

We can derive Totalizer encoding [Bailleux and Boufkhad 2003] by introducing new integer variables for each node in the dichotomic decomposition of the summation.

Totalizer encoding

When we introduce new integer variables.

$$\begin{array}{rcl}
s_{1\,..\,2} & \geq & x_1 + x_2 \\
s_{3\,..\,4} & \geq & x_3 + x_4 \\
s_{1\,..\,4} & \geq & s_{1\,..\,2} + s_{3\,..\,4} \\
 & \cdots & \\
s_{1\,..\,n} & \geq & s_{1\,..\,n/2} + s_{n/2+1\,..\,n} \\
s_{1\,..\,n} & \leq & k
\end{array}$$

Order Encoding of BC (3/3)

We can derive Sequential Counter encoding [Sinz 2005] by introducing new integer variables for each node in the sequential decomposition of the summation.

Totalizer encoding

When we introduce new integer variables.

$$\begin{aligned} s_1 &\geq x_1 \\ s_2 &\geq x_2 + s_1 \\ s_3 &\geq x_3 + s_2 \\ &\cdots \\ s_n &\geq x_n + s_{n-1} \\ s_n &\leq k \end{aligned}$$

Pseudo-Boolean Constraints

Pseudo-Boolean constraints (PB constraints for short) are constraints of the form $a_1x_1 + a_2x_2 + \cdots + a_nx_n \# k$ where

- a_i's and k are integer constants,
- x_i's are literals taking a value 0 (false) or 1 (true), and
- $\#$ is one of the relations $\{\geq, \leq, =, \neq\}$.

$a_1x_1 + a_2x_2 + \cdots + a_nx_n$ and k are called the LHS and RHS of the constraint respectively.

Without loss of generality, we can assume:

- Literals x_i's are all distinct and no complementary pairs.
- The coefficients a_i's are all positive and no common divisors except 1.
- The RHS value k is non-negative.

Pseudo-Boolean Satisfaction

- PB is a generalization of SAT.

$$x_1 \vee x_2 \vee \cdots \vee x_n \iff x_1 + x_2 + \cdots + x_n \geq 1$$

- Arithmetic constraints on integers can be represented as a PB constraint by using binary number representation.

$$\textstyle\sum_{i=0}^{m-1} 2^i x_i$$

- Disjunctions of PB constraints can be represented by a conjunction of PB constraints.

$$\begin{aligned} & \textstyle\sum_{i=1}^{m} a_i x_i \geq k \vee \sum_{j=1}^{n} b_j y_j \geq l \\ \iff \quad & p + q \geq 1 \wedge k\overline{p} + \textstyle\sum_{i=1}^{m} a_i x_i \geq k \wedge l\overline{q} + \sum_{j=1}^{n} b_j y_j \geq l \end{aligned}$$

- Satisfiability testing of one PB constraint $\sum a_i x_i = k$ is already NP-complete (subset sum problem).

PB Solvers

PB solver is a program to solve the given PB instance. It finds an assignment satisfying all given PB constraints and sometimes also minimizing/maximizing the given objective function.

- Many good PB solvers can be found at PB Competition annually held since 2005.
  - Sat4j [Le Berre and Parrain 2010]
  - clasp [Gebser et al. 2007]
  - bsolo [Manquinho and Silva 2006]
  - wbo [Manquinho et al. 2010]
  - MiniSat+ [Eén and Sörensson 2006], etc.
- Also, there are many attempts of encoding PB constraints to SAT including:
  - [Eén and Sörensson 2006] (Adder+BDD+Sorter)
  - [Bailluex *et al.* 2006] (BDD-like)
  - [Bailluex *et al.* 2009] (WatchDog)
  - [Abio *et al.* 2012] (BDD)

Order Encoding of PB (1/3)

Now, we consider to apply the optimized order encoding to PB constraint $\sum_{i=1}^{n} a_i x_i \# k$.

However, its direct application generates too many clauses, that is, we will obtain 2^{n-1} clauses in general.

Order Encoding of PB (2/3)

Introducing auxiliary variables of partial sums

We introduce integer variables s_i's representing the partial sum of the first i terms of the LHS, that is:

$$s_1 = a_1 x_1, \qquad s_i = a_i x_i + s_{i-1} \quad (i = 2 \mathinner{..} n).$$

Each domain $Dom(s_i)$ can be defined by $\{\sum A \mid A \subset \{a_1, \ldots, a_i\}\}$.

Example when the LHS is $3x_1 + 2x_2$

$$\begin{aligned} s_1 &= 3x_1 \\ s_2 &= 2x_2 + s_1 \\ Dom(s_1) &= \{0, 3\} \\ Dom(s_2) &= \{0, 2, 3, 5\} \end{aligned}$$

Order Encoding of PB (3/3)

Then, we obtain the followings by applying the order encoding.

$$\begin{array}{rll}
x_1 \lor \neg p(s_1 \ge a_1) & & (1) \\
\neg x_1 \lor p(s_1 \ge a_1) & & (2) \\
p(s_{i-1} \ge d - a_i) \lor \neg p(s_i \ge d) & (d \in Dom(s_i)) & (3) \\
x_i \lor p(s_{i-1} \ge d) \lor \neg p(s_i \ge d) & (d \in Dom(s_i)) & (4) \\
\neg p(s_{i-1} \ge d) \lor p(s_i \ge d) & (d \in Dom(s_{i-1})) & (5) \\
\neg x_i \lor \neg p(s_{i-1} \ge d) \lor p(s_i \ge d + a_i) & (d \in Dom(s_{i-1})) & (6)
\end{array}$$

- (1) and (2) correspond to $s_1 \le a_1 x_1$ and $s_1 \ge a_1 x_1$ respectively
- (3) and (4) correspond to $s_i \le a_1 x_1 + s_{i-1}$
- (5) and (6) correspond to $s_i \ge a_1 x_1 + s_{i-1}$

The obtained encoding is equivalent to the one in [Bailleux *et al.* 2006] except the definitions of $Dom(s_i)$.

Counter Matrix (1/2)

Boolean variables $p(s_i \geq j)$ compose a sparse Boolean matrix $(s_{i,j})$ named *Counter Matrix* (CM for short).

Each component $s_{i,j}$ satisfies:

$$s_{i,j} \iff a_1x_1 + \cdots + a_ix_i \geq j$$

Therefore, PB constraint $\sum_{i=1}^{n} a_ix_i \mathbin{\#} k$ can be represented by:

$$\begin{array}{cl}
s_{n,k} & \text{(when } \# \text{ is } \geq) \\
\neg s_{n,k+1} & \text{(when } \# \text{ is } \leq) \\
s_{n,k} \land \neg s_{n,k+1} & \text{(when } \# \text{ is } =) \\
\neg s_{n,k} \lor s_{n,k+1} & \text{(when } \# \text{ is } \neq)
\end{array}$$

Counter Matrix (2/2)

Features of Counter Matrix

- The column size of the CM can be limited to $k+1$.
  - The numbers of aux. variables and clauses are $O(nk)$.
- CM can be reused for other constraints with the same LHS.
  - We need only one CM for $\sum_{i=1}^{n} a_i x_i \geq k$ and $\sum_{i=1}^{n} a_i x_i \leq k$.
- It is also possible to reuse a part of CM in another PB constraint having the same sub-terms in the head.

$$\begin{array}{lll} C_1 : & a_1 x_1 + \cdots + a_i x_i + \cdots & \# \; k \\ C_2 : & a_1 x_1 + \cdots + a_i x_i + \cdots & \#' \; k' \end{array}$$

  - We can reused the first i rows of the CM when $k \geq k'$.

- Compared with the encoding in [Bailleux *et al.* 2006], the proposed encoding enables more sharings of CMs and will contribute to decrease the size of generated CNF (and hopefully improve the efficiency).

Implementation

We implemented the encoding program as PBSugar.

- PB constraint with at most three literals is directly encoded to CNF by the order encoding.
- Otherwise, the CM is constructed for the constraint.
  - Reusing and Sharing caches contain at most 1000 CMs and is managed by LRU strategy.

- PBSugar is written in Java.
- Glucose version 2.1 with SatELite preprocessor is used as the default back-end SAT solver.
- For calculating the domains of s_i's, balanced Diet (discrete interval encoding tree) structure proposed in [Ohnishi *et al.* 2003] is used.

Experimental Results

We made a comparison with the following 5 solvers.

Solver	Reference
Sat4j PB Res//CP v2.3.2	[Le Berre and Parrain 2010]
clasp v2.1.1 (64-bit code)	[Gebser et al. 2007]
bsolo v3.2 (32-bit code)	[Manquinho and Silva 2006]
wbo v1.4a (32-bit code)	[Manquinho et al. 2010]
MiniSat+ v1.0 (64-bit code)	[Eén and Sörensson 2006]

- Sat4j PB Res//CP v2.3.2 was ranked first in 2012.
- clasp v2.1.1 is a newer version of clasp v2.0.6 which was ranked second in 2012.
- bsolo v3.2 was ranked third in 2010.
- wbo v1.4a was ranked fourth in 2010.
- MiniSat+ v1.0 was ranked third in 2006.

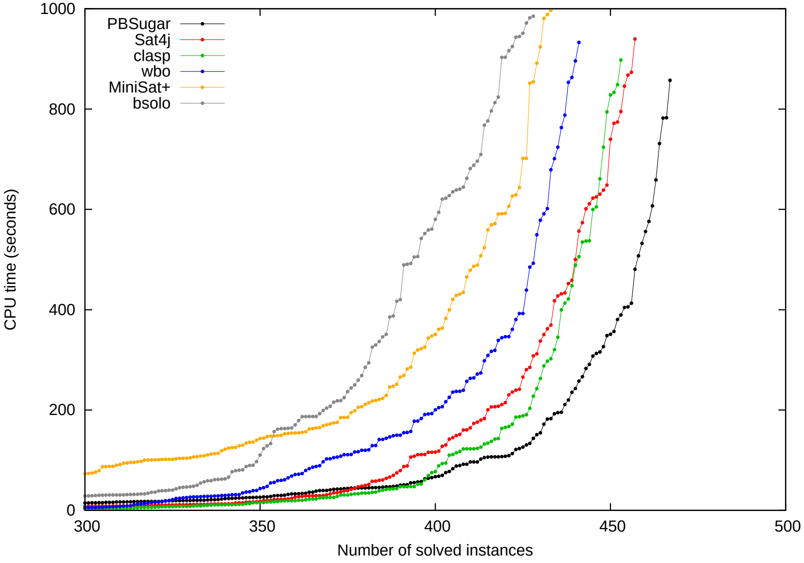
Experimental Results

- We used 669 instances of DEC-SMALLINT-LIN (decision problems, small integers, linear constraints) category of 2011 PB competition benchmark set.
  - classified to 22 families as done in [Abio *et al.* 2012]
- We compared
  - the number of solved instances within the time limit of 1000 seconds
  - on Intel Dual Core Xeon 3.33GHz machine.

Number of solved instances

Family	VBS	PBSugar	Sat4j	clasp	bsolo	wbo	MiniSat+
lopes (200)	86	**81**	44	67	36	65	64
army (12)	11	8	5	**10**	6	5	6
blast (8)	8	**8**	**8**	**8**	**8**	**8**	**8**
cache (9)	9	**9**	6	**9**	6	6	8
chnl (21)	21	6	**21**	3	**21**	3	3
dbstv (15)	15	**15**	**15**	**15**	**15**	**15**	**15**
dlx (3)	3	**3**	**3**	**3**	**3**	**3**	**3**
elf (5)	5	**5**	**5**	**5**	**5**	**5**	**5**
fpga (36)	36	**36**	**36**	34	**36**	**36**	33
j30-120 (80)	67	66	66	**67**	64	66	63
neos (4)	2	**2**	**2**	**2**	**2**	**2**	**2**
ooo (19)	18	15	15	**17**	12	16	16
pig-card (20)	20	2	**20**	2	19	2	2
pig-cl (20)	2	**2**	**2**	**2**	**2**	**2**	**2**
ppp (6)	5	**4**	**4**	3	**4**	3	**4**
robin (6)	4	3	**4**	**4**	3	3	3
13queen (100)	100	**100**	**100**	**100**	**100**	**100**	**100**
11tsp11 (100)	100	**100**	**100**	**100**	85	**100**	95
vdw (5)	3	**2**	1	**2**	1	1	1
Total (669)	515	**467**	457	453	428	441	433

Cactus plot


PBSugar
Sat4j
clasp
wbo
MiniSat+
bsolo
CPU time (seconds)
0
200
400
600
800
1000
300
350
400
450
500
Number of solved instances

Conclusion

We presented:

- the optimized version of the order encoding applicable to non-contiguous domain variables,
- the encoding of PB constraints by using the optimized version,
- the idea of Counter Matrix which allows reusing/sharing of clauses among different PB constraints,
- the implementation named PBSugar, and
- the experimental results for 669 DEC-SMALLINT-LIN instances.

Order Encoding: Optimized Version

Function Encode($a_1x_1 + \cdots + a_nx_n \geq c$)

```
1: if $n = 1$ then
2:     if $a_1 > 0$ then
3:         $\Psi \leftarrow \{\{P(x_1 \geq \lceil c/a_1 \rceil)\}\}$
4:     else $\{a_1 < 0\}$
5:         $\Psi \leftarrow \{\{\neg P(x_1 \geq \lfloor c/a_1 \rfloor + 1)\}\}$
6:     end if
7: else $\{n > 1\}$
8:     $D \leftarrow \{d \in Dom(x) \mid c - \sup(\sum_{i=2}^n a_ix_i) \leq a_1d < c - \inf(\sum_{i=2}^n a_ix_i)\}$
9:     $D' \leftarrow \{d \in Dom(x) \mid a_1d < c - \sup(\sum_{i=2}^n a_ix_i)\}$
10:    if $a_1 > 0$ then
11:        $\Psi \leftarrow \{\{P(x_1 \geq d+1)\} \cup \theta \mid d \in D,\ \theta \in \mathrm{Encode}(a_2x_2 + \cdots + a_nx_n \geq c - a_1d)\}$
12:        if $D' \neq \emptyset$ then
13:            $\Psi \leftarrow \{\{P(x_1 \geq \max(D') + 1\}\} \cup \Psi$
14:        end if
15:    else $\{a_1 < 0\}$
16:        $\Psi \leftarrow \{\{\neg P(x_1 \geq d)\} \cup \theta \mid d \in D,\ \theta \in \mathrm{Encode}(a_2x_2 + \cdots + a_nx_n \geq c - a_1d)\}$
17:        if $D' \neq \emptyset$ then
18:            $\Psi \leftarrow \{\{\neg P(x_1 \geq \min(D')\}\} \cup \Psi$
19:        end if
20:    end if
21: end if
22: return $\Psi$
```

Number of auxiliary variables (in thousands)

Family	PBSugar	Adder	WD1	WD2	BDD1	BDD2	BDD3
lopes (200)	**54.07**	1744.05	3566.11	5478.81	2393.97	2394.00	7734.65
army (12)	4.66	**4.63**	10.96	245.49	6.36	6.36	479.83
blast (8)	2738.13	**27.77**	62.22	1394.50	36.73	39.67	761.29
cache (9)	729.06	**145.66**	339.18	2393.97	201.18	210.83	1503.19
chnl (21)	**4.52**	8.39	24.55	1007.90	6.76	6.76	184.59
dbstv30 (5)	**39.99**	219.82	709.73	–	441.86	1695.87	–
dbstv40 (5)	**90.98**	2468.45	6564.44	–	4282.16	7225.63	–
dbstv50 (5)	**187.40**	6135.13	16365.39	–	11111.06	19723.37	–
dlx (3)	41.53	**10.40**	21.62	247.79	12.40	13.89	126.81
elf (5)	103.15	**20.37**	42.78	571.38	24.62	28.13	306.76
fpga (36)	**1.82**	21.15	53.96	1074.03	13.27	13.27	242.03
j120 (28)	698.46	**159.43**	540.00	5713.75	612.13	806.08	22246.82
j30 (17)	70.16	**18.15**	50.80	1190.59	44.96	59.82	1153.51
j60 (18)	218.58	**37.03**	112.35	4775.72	157.92	180.05	7285.69
j90 (17)	770.31	**65.41**	217.01	6543.52	553.80	553.76	19793.50
neos (4)	**62.69**	73.74	185.59	3542.94	79.33	122.53	2003.95
ooo (19)	818.80	**118.15**	273.54	2248.35	162.25	168.61	1315.77
pig-card (20)	**7.97**	15.26	50.75	2966.58	11.94	11.94	632.33
pig-cl (20)	**0.00**	7.68	25.25	1984.06	4.01	4.01	310.03
ppp (6)	**26.00**	57.13	141.49	623.59	81.57	82.86	382.68
robin (6)	**123.37**	171.67	628.45	3634.13	158.55	158.55	16565.75
13queen (100)	5.87	**2.20**	6.18	461.54	5.63	7.08	791.43
11tsp11 (100)	**2.35**	3.37	8.83	170.59	5.71	6.51	221.21
vdw (5)	**2.23**	1895.39	3356.94	12818.51	1391.65	1391.65	5875.58
Average (669)	**145.44**	602.15	1289.35	2324.57	896.64	1002.94	4068.87

- Comparison with the results reported in [Abio *et al.* 2012]

Number of clauses (in thousands)

Family	PBSugar	Adder	WD1	WD2	BDD1	BDD2	BDD3
lopes (200)	**1397.73**	10901.95	7730.20	22171.19	3436.72	3436.67	10004.63
army (12)	24.03	26.29	34.69	2155.89	11.07	**11.06**	925.12
blast (8)	12316.27	191.46	108.58	2114.99	77.41	**71.97**	1271.34
cache (9)	3437.40	1161.61	731.78	3833.01	**453.37**	456.36	2599.87
chnl (21)	20.22	56.82	116.97	4936.16	**11.23**	**11.23**	285.67
dbstv30 (5)	**522.96**	1823.61	3693.95	–	1183.59	3608.42	–
dbstv40 (5)	**1434.74**	18169.82	17901.80	–	6511.80	12244.49	–
dbstv50 (5)	**3480.32**	45349.38	46862.83	–	16952.14	33831.20	–
dlx (3)	204.18	86.15	56.59	398.67	43.94	**43.49**	229.83
elf (5)	505.38	175.49	117.73	927.47	92.75	**92.51**	553.90
fpga (36)	**8.69**	139.45	175.66	3615.21	15.65	15.65	278.37
j120 (28)	3530.43	**1207.00**	3907.42	26283.89	1290.70	1674.87	44157.85
j30 (17)	348.67	135.25	178.47	3903.27	**103.22**	129.73	2257.71
j60 (18)	1097.27	**283.99**	525.65	22873.88	341.98	381.84	14386.03
j90 (17)	3851.46	**501.05**	1336.72	34184.53	1156.78	1145.67	39163.47
neos (4)	288.47	472.92	596.17	12411.55	**140.71**	221.90	3682.71
ooo (19)	3751.32	888.58	537.69	3473.38	**314.62**	322.82	2221.64
pig-card (20)	35.75	104.68	367.03	20711.11	**19.86**	**19.86**	958.65
pig-cl (20)	**161.21**	213.56	341.48	14802.42	165.22	165.22	475.20
ppp (6)	147.10	422.45	301.51	1834.14	**130.73**	133.08	684.32
robin (6)	562.68	1185.92	6916.35	19694.86	**281.42**	**281.42**	28875.61
13queen (100)	28.31	14.74	38.92	5068.36	**10.85**	13.74	1573.90
11tsp11 (100)	13.98	25.97	28.01	1338.27	**10.43**	12.01	436.26
vdw (5)	**287.18**	10894.94	6573.12	24283.19	1671.71	1671.71	7271.86
Average (669)	**1070.76**	3864.52	3146.49	10316.25	1341.46	1548.42	6339.21

- Comparison with the results reported in [Abio *et al.* 2012]