Effective Auxiliary Variables via Structured Reencoding

Andrew Haberlandt*, Harrison Green*, Marijn Heule

*equal contribution
Potential of auxiliary variables...
Potential of auxiliary variables...

...in encodings:

<table>
<thead>
<tr>
<th>Encoding</th>
<th>#clauses</th>
<th>#aux. vars</th>
<th>decided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve</td>
<td>( \binom{n}{k+1} )</td>
<td>0</td>
<td>immediately</td>
</tr>
<tr>
<td>Sequential unary counter (LT\textsuperscript{n,k}\textsubscript{SEQ})</td>
<td>( \mathcal{O}(n \cdot k) )</td>
<td>( \mathcal{O}(n \cdot k) )</td>
<td>by unit prop.</td>
</tr>
<tr>
<td>Parallel binary counter (LT\textsuperscript{n,k}\textsubscript{PAR})</td>
<td>( 7n - 3 \lfloor \log n \rfloor - 6 )</td>
<td>2n - 2</td>
<td>by search</td>
</tr>
<tr>
<td>Bailleux &amp; Boufkhad [3]</td>
<td>( \mathcal{O}(n^2) )</td>
<td>( \mathcal{O}(n \cdot \log n) )</td>
<td>by unit prop.</td>
</tr>
<tr>
<td>Warners [4]</td>
<td>8n</td>
<td>2n</td>
<td>by search</td>
</tr>
</tbody>
</table>

Potential of auxiliary variables...

...in encodings:

<table>
<thead>
<tr>
<th>Encoding</th>
<th>#clauses</th>
<th>#aux. vars</th>
<th>decided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve</td>
<td>$\binom{n}{k+1}$</td>
<td>0</td>
<td>immediately</td>
</tr>
<tr>
<td>Sequential unary counter (LT$_{SEQ}^{n,k}$)</td>
<td>$\mathcal{O}(n \cdot k)$</td>
<td>$\mathcal{O}(n \cdot k)$</td>
<td>by unit prop.</td>
</tr>
<tr>
<td>Parallel binary counter (LT$_{PAR}^{n,k}$)</td>
<td>$7n - 3 \lfloor \log n \rfloor - 6$</td>
<td>$2n - 2$</td>
<td>by search</td>
</tr>
<tr>
<td>Bailleux &amp; Boufkhad [3]</td>
<td>$\mathcal{O}(n^2)$</td>
<td>$\mathcal{O}(n \cdot \log n)$</td>
<td>by unit prop.</td>
</tr>
<tr>
<td>Warners [4]</td>
<td>$8n$</td>
<td>$2n$</td>
<td>by search</td>
</tr>
</tbody>
</table>


...for proofs:

A SHORT PROOF OF THE PIGEON HOLE PRINCIPLE USING EXTENDED RESOLUTION

Stephen A. Cook
Potential of auxiliary variables...

...in encodings:

<table>
<thead>
<tr>
<th>Encoding</th>
<th>#clauses</th>
<th>#aux. vars</th>
<th>decided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve</td>
<td>(\binom{n}{k+1})</td>
<td>0</td>
<td>immediately</td>
</tr>
<tr>
<td>Sequential unary counter (LT(n,k)(_{SEQ}))</td>
<td>(\mathcal{O}(n \cdot k))</td>
<td>(\mathcal{O}(n \cdot k))</td>
<td>by unit prop.</td>
</tr>
<tr>
<td>Parallel binary counter (LT(n,k)(_{PAR}))</td>
<td>(7n - 3\lfloor \log n \rfloor - 6)</td>
<td>2n - 2</td>
<td>by search</td>
</tr>
<tr>
<td>Bailleux &amp; Bouffkhad [3]</td>
<td>(\mathcal{O}(n^2))</td>
<td>(\mathcal{O}(n \cdot \log n))</td>
<td>by unit prop.</td>
</tr>
<tr>
<td>Warners [4]</td>
<td>8n</td>
<td>2n</td>
<td>by search</td>
</tr>
</tbody>
</table>

| Table 1. Comparison of different encodings for \(\leq k(x_1, \ldots, x_n)\). |


...for proofs:

A SHORT PROOF OF THE PIGEON HOLE PRINCIPLE USING EXTENDED RESOLUTION

Stephen A. Cook

Bounded Variable Addition (Manthey et al., 2012) reduces formulas by adding auxiliary variables

\[ a \lor p \]
\[ a \lor q \]
\[ a \lor r \]
\[ b \lor p \]
\[ b \lor q \]
\[ b \lor r \]

Bounded Variable Addition (Manthey et al., 2012) reduces formulas by adding auxiliary variables

\[ \begin{align*}
  a \lor p & \quad \rightarrow \quad x \lor p \\
  a \lor q & \quad \rightarrow \quad x \lor q \\
  a \lor r & \quad \rightarrow \quad x \lor r \\
  b \lor p & \quad \rightarrow \quad \overline{x} \lor a \\
  b \lor q & \quad \rightarrow \quad \overline{x} \lor b \\
  b \lor r &
\end{align*} \]
Bounded Variable Addition (Manthey et al., 2012) reduces formulas by adding auxiliary variables

\[ a \lor p \]
\[ a \lor q \]
\[ a \lor r \]
\[ b \lor p \]
\[ b \lor q \]
\[ b \lor r \]

\[ x \lor p \]
\[ x \lor q \]
\[ x \lor r \]

\[ \overline{x} \lor a \]
\[ \overline{x} \lor b \]

\[ a \lor p \]
\[ a \lor q \]
\[ a \lor r \]
\[ b \lor p \]
\[ b \lor q \]
\[ b \lor r \]
Bounded Variable Addition (Manthey et al., 2012) reduces formulas by adding auxiliary variables.

\[
\begin{align*}
    a \lor p & \quad \rightarrow \quad x \lor p \\
    a \lor q & \quad \rightarrow \quad x \lor q \\
    a \lor r & \quad \rightarrow \quad x \lor r \\
    b \lor p & \quad \rightarrow \quad \overline{x} \lor a \\
    b \lor q & \quad \rightarrow \quad \overline{x} \lor b \\
    b \lor r & \quad \rightarrow \quad \overline{x} \lor b
\end{align*}
\]

\[
(\overline{x} \lor a) \otimes (x \lor p) = a \lor p
\]
Bounded Variable Addition greedily constructs auxiliary variables
Bounded Variable Addition greedily constructs auxiliary variables

Packing k-coloring Problem - auxiliary variables represent regions
Bounded Variable Addition greedily constructs auxiliary variables

Packing k-coloring Problem - auxiliary variables represent regions

Randomization causes BVA to generate ineffective auxiliary variables
**Bounded Variable Addition** greedily constructs auxiliary variables

Packing k-coloring Problem - **auxiliary variables represent regions**

**Randomization** causes BVA to generate ineffective auxiliary variables

**SBVA** improves BVA with a **structure-based tiebreaking heuristic**
Bounded Variable Addition greedily constructs auxiliary variables

Packing k-coloring Problem - auxiliary variables represent regions

Randomization causes BVA to generate ineffective auxiliary variables

SBVA improves BVA with a structure-based tiebreaking heuristic

SBVA is effective on a diverse competition-style benchmark
BVA finds grids of clauses in an iterative, greedy manner

<table>
<thead>
<tr>
<th></th>
<th>$x \lor p$</th>
<th>$x \lor q$</th>
<th>$x \lor r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{x} \lor a$</td>
<td>$a \lor p$</td>
<td>$a \lor q$</td>
<td>$a \lor r$</td>
</tr>
<tr>
<td>$\bar{x} \lor b$</td>
<td>$b \lor p$</td>
<td>$b \lor q$</td>
<td>$b \lor r$</td>
</tr>
</tbody>
</table>
BVA finds grids of clauses in an iterative, greedy manner

<table>
<thead>
<tr>
<th></th>
<th>( p )</th>
<th>( q )</th>
<th>( r )</th>
<th>( s \lor t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>( a \lor p )</td>
<td>( a \lor q )</td>
<td>( a \lor r )</td>
<td>( a \lor s \lor t )</td>
</tr>
<tr>
<td>( b )</td>
<td>( b \lor p )</td>
<td>( b \lor q )</td>
<td>( b \lor r )</td>
<td></td>
</tr>
<tr>
<td>( c )</td>
<td></td>
<td>( c \lor q )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BVA finds grids of clauses in an iterative, greedy manner

\[
\begin{array}{cccc}
\varphi & a & \eta & s \lor t \\
\hline
a & \bullet & \bullet & \bullet \\
\hline
b & \bullet & \bullet & \bullet \\
\hline
c & & \bullet & \\
\end{array}
\]
BVA finds grids of clauses in an iterative, greedy manner

<table>
<thead>
<tr>
<th></th>
<th>( \varphi )</th>
<th>( \alpha )</th>
<th>( \gamma )</th>
<th>( s \lor t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>( b )</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>( c )</td>
<td></td>
<td></td>
<td>●</td>
<td></td>
</tr>
</tbody>
</table>
BVA finds grids of clauses in an iterative, greedy manner

<table>
<thead>
<tr>
<th></th>
<th>$\varphi$</th>
<th>$\alpha$</th>
<th>$\gamma$</th>
<th>$\delta \lor \iota$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td><img src="bullet.png" alt="bullet" /></td>
<td><img src="bullet.png" alt="bullet" /></td>
<td><img src="bullet.png" alt="bullet" /></td>
<td><img src="bullet.png" alt="bullet" /></td>
</tr>
<tr>
<td>$b$</td>
<td><img src="bullet.png" alt="bullet" /></td>
<td><img src="bullet.png" alt="bullet" /></td>
<td><img src="bullet.png" alt="bullet" /></td>
<td><img src="bullet.png" alt="bullet" /></td>
</tr>
<tr>
<td>$c$</td>
<td></td>
<td></td>
<td><img src="bullet.png" alt="bullet" /></td>
<td></td>
</tr>
</tbody>
</table>
BVA finds grids of clauses in an iterative, greedy manner

<table>
<thead>
<tr>
<th></th>
<th>$\varphi$</th>
<th>$\alpha$</th>
<th>$\gamma$</th>
<th>$\delta \lor \tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>⬜</td>
<td>⬜</td>
<td>⬜</td>
<td>⬜</td>
</tr>
<tr>
<td>$b$</td>
<td>⬜</td>
<td>⬜</td>
<td>⬜</td>
<td>⬜</td>
</tr>
<tr>
<td>$c$</td>
<td></td>
<td></td>
<td>⬜</td>
<td></td>
</tr>
</tbody>
</table>
BVA finds grids of clauses in an iterative, greedy manner

<table>
<thead>
<tr>
<th></th>
<th>$\varphi$</th>
<th>$\varrho$</th>
<th>$\tau$</th>
<th>$s \lor t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Andrew Haberlandt, Harrison Green, and Marijn Heule
Example: The packing k-coloring of the square grid

Example: The packing k-coloring of the square grid

Example: The packing k-coloring of the square grid

Direct encoding:

\[ \nu(x,y), k \] denotes whether location \((x, y)\) has color \(k\).

Example: The packing k-coloring of the square grid

Direct encoding:

\( v(x,y),k \) denotes whether location \((x, y)\) has color \(k\).

1. At-Least-One-Color clauses

\( v(0,0),1 \lor v(0,0),2 \lor \ldots \lor v(0,0),6 \)

---

Example: The packing k-coloring of the square grid

Direct encoding:

\( v(x,y),k \) denotes whether location \((x, y)\) has color \(k\)

1. At-Least-One-Color clauses

\[ v(0,0),1 \lor v(0,0),2 \lor \ldots \lor v(0,0),6 \]

2. At-Most-One-Distance clauses

\[
\begin{align*}
\overline{v(0,0),1} & \lor \overline{v(0,1),1} \\
\overline{v(0,0),1} & \lor \overline{v(1,0),1} \\
\ldots & \\
\overline{v(0,0),2} & \lor \overline{v(0,1),2} \\
\overline{v(1,0),2} & \lor v(0,0),2
\end{align*}
\]
Example: Auxiliary variables describe regions of variables

Example: Different ‘regions’ may be more effective

<table>
<thead>
<tr>
<th></th>
<th>direct encoding</th>
<th>bva encoding</th>
<th>plus encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$D_{5,10,5}$</td>
<td>$D_{5,10,5}$</td>
<td>$D_{5,10,5}$</td>
</tr>
<tr>
<td>Number of variables</td>
<td>610</td>
<td>973</td>
<td>673</td>
</tr>
<tr>
<td>Number of clauses</td>
<td>10688</td>
<td>2313</td>
<td>4063</td>
</tr>
<tr>
<td>CDCL runtime (s)</td>
<td>255.12</td>
<td>39.88</td>
<td>15.90</td>
</tr>
</tbody>
</table>

*Subercaseaux and Heule (2023) studied variables added by BVA to derive an efficient manual encoding.*

### BVA is sensitive to randomization

<table>
<thead>
<tr>
<th></th>
<th>Vars</th>
<th>Clauses</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>610</td>
<td>10688</td>
<td>590</td>
</tr>
<tr>
<td>BVA</td>
<td>973</td>
<td>2313</td>
<td>38</td>
</tr>
<tr>
<td>Randomize + BVA</td>
<td>971</td>
<td>2305</td>
<td>107</td>
</tr>
</tbody>
</table>

- Original: 590 seconds
- BVA: 38 seconds
- Randomize + BVA: 107 seconds

The table shows that Randomize + BVA is about 2x slower than BVA.
Individual variable additions have a disproportionate effect

<table>
<thead>
<tr>
<th></th>
<th>Vars</th>
<th>Clauses</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>610</td>
<td>10688</td>
<td>590</td>
</tr>
<tr>
<td>BVA</td>
<td>973</td>
<td>2313</td>
<td>38</td>
</tr>
<tr>
<td>Randomize + BVA</td>
<td>971</td>
<td>2305</td>
<td>107</td>
</tr>
<tr>
<td>1x BVA</td>
<td>611</td>
<td>9819</td>
<td>105</td>
</tr>
<tr>
<td>1x Randomize + BVA</td>
<td>611</td>
<td>9819</td>
<td>429</td>
</tr>
</tbody>
</table>

BVA: >5x faster
Randomize + BVA: >4x slower
Tiebreaking in BVA leads to different auxiliary variables

<table>
<thead>
<tr>
<th></th>
<th>p ∨ q</th>
<th>p ∨ r</th>
<th>r ∨ s</th>
<th>t</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>b</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>
Tiebreaking in BVA leads to different auxiliary variables

<table>
<thead>
<tr>
<th></th>
<th>p \lor q</th>
<th>p \lor r</th>
<th>r \lor s</th>
<th>t</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>b</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>
Tiebreaking in BVA leads to different auxiliary variables

<table>
<thead>
<tr>
<th></th>
<th>p ∨ q</th>
<th>p ∨ r</th>
<th>r ∨ s</th>
<th>t</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tiebreaking in BVA leads to different auxiliary variables
Using formula structure to break ties
Using formula structure to break ties
Using formula structure to break ties
Using formula structure to break ties
Using formula structure to break ties
Breaking ties: Counting paths in the Variable Incidence Graph (VIG)

Variable incidence graph (VIG): a weighted graph of variables, where two variables share an edge for each clause between them.

A: adjacency matrix of the VIG

H(x,y): number of paths of length 3 in the VIG

\[ H(x, y) = (A^3)_{x,y} \]

“Nearby” variables have more paths of length 3 in the VIG between them.
SBVA: Break ties using the “3-HOP” heuristic

Compare $H(a, b)$ and $H(a, c)$
SBVA constructs effective auxiliary variables, even on randomized formulas

<table>
<thead>
<tr>
<th></th>
<th>Vars</th>
<th>Clauses</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>610</td>
<td>10688</td>
<td>590</td>
</tr>
<tr>
<td>BVA</td>
<td>973</td>
<td>2313</td>
<td>38</td>
</tr>
<tr>
<td>Randomize + BVA</td>
<td>971</td>
<td>2305</td>
<td>107</td>
</tr>
<tr>
<td>Randomize + SBVA</td>
<td>972</td>
<td>2290</td>
<td>55</td>
</tr>
</tbody>
</table>
Evaluation Questions
Evaluation Questions

1. Is SBVA worth running as a preprocessor?
   a. Is it worth it to spend time preprocessing formulas before solving?
   b. How does SBVA compare to BVA (and to no preprocessor)?
Evaluation Questions

1. Is SBVA worth running as a preprocessor?
   a. Is it worth it to spend time preprocessing formulas before solving?
   b. How does SBVA compare to BVA (and to no preprocessor)?

2. What types of formulas does SBVA work well on?
   a. Are there common structures?
   b. Does reduction factor correlate with speedup?
Four Experimental Variants

<table>
<thead>
<tr>
<th>Preprocessor</th>
<th>Formula</th>
<th>Baseline</th>
<th>CaDiCaL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BVA-ORIG</td>
<td></td>
<td>Formula</td>
<td>BVA</td>
</tr>
<tr>
<td>BVA-RAND-ORIG</td>
<td></td>
<td>Formula</td>
<td>BVA</td>
</tr>
<tr>
<td>BVA-RAND-3HOP</td>
<td></td>
<td>Formula</td>
<td>SBVA</td>
</tr>
</tbody>
</table>

Time Limit (BVA + CaDiCaL) 5000s
BVA Timeout 200s

= scranfilize (-p -P -f 0.5)
## Dataset

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Full</strong></td>
<td>29,402</td>
</tr>
<tr>
<td><strong>ANNI-2022</strong></td>
<td>5,355</td>
</tr>
</tbody>
</table>

\(^1\)https://benchmark-database.de/
SBVA solved the most formulas in both datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Variant</th>
<th>ALL PAR-2</th>
<th>#</th>
<th>UNSAT PAR-2</th>
<th>#</th>
<th>SAT PAR-2</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>FULL</td>
<td>Baseline</td>
<td>1077.91</td>
<td>21602</td>
<td>756.14</td>
<td>6495</td>
<td>1196.99</td>
<td>15107</td>
</tr>
<tr>
<td></td>
<td>BVA-orig</td>
<td>867.04</td>
<td>22140</td>
<td>635.71</td>
<td>6562</td>
<td>948.85</td>
<td>15578</td>
</tr>
<tr>
<td></td>
<td>BVA-rand-orig</td>
<td>870.20</td>
<td>22077</td>
<td>673.58</td>
<td>6533</td>
<td>953.25</td>
<td>15544</td>
</tr>
<tr>
<td></td>
<td>BVA-rand-3hop</td>
<td><strong>862.29</strong></td>
<td><strong>22173</strong></td>
<td>650.41</td>
<td><strong>6568</strong></td>
<td><strong>935.38</strong></td>
<td><strong>15605</strong></td>
</tr>
<tr>
<td>ANNI-2022</td>
<td>Baseline</td>
<td>1262.18</td>
<td>3953</td>
<td>1164.61</td>
<td>2048</td>
<td><strong>1309.41</strong></td>
<td>1905</td>
</tr>
<tr>
<td></td>
<td>BVA-orig</td>
<td><strong>1174.80</strong></td>
<td>3987</td>
<td><strong>967.85</strong></td>
<td>2085</td>
<td>1338.31</td>
<td>1902</td>
</tr>
<tr>
<td></td>
<td>BVA-rand-orig</td>
<td>1193.27</td>
<td>3958</td>
<td>1053.75</td>
<td>2060</td>
<td>1350.09</td>
<td>1898</td>
</tr>
<tr>
<td></td>
<td>BVA-rand-3hop</td>
<td><strong>1188.63</strong></td>
<td><strong>3995</strong></td>
<td>982.84</td>
<td><strong>2088</strong></td>
<td>1350.98</td>
<td><strong>1907</strong></td>
</tr>
</tbody>
</table>

All times include BVA runtime
## Pigeonhole Problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Baseline</th>
<th>BVA-ORIG</th>
<th>BVA-RAND-ORIG</th>
<th>BVA-RAND-3HOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>fphp-010-008.shuffled-as.sat05-1213</td>
<td>0.5</td>
<td>0.2</td>
<td>0.3</td>
<td>0.0</td>
</tr>
<tr>
<td>fphp-010-009.shuffled-as.sat05-1227</td>
<td>5.4</td>
<td>4.3</td>
<td>4.2</td>
<td>0.0</td>
</tr>
<tr>
<td>fphp-012-010.shuffled-as.sat05-1214</td>
<td>113.4</td>
<td>45.1</td>
<td>36.2</td>
<td>0.1</td>
</tr>
<tr>
<td>fphp-012-011.shuffled-as.sat05-1228</td>
<td>1722.9</td>
<td>1525.2</td>
<td>1844.1</td>
<td>0.7</td>
</tr>
<tr>
<td>fphp-014-012.shuffled-as.sat05-1215</td>
<td>T.O.</td>
<td>T.O.</td>
<td>T.O.</td>
<td>1.9</td>
</tr>
<tr>
<td>fphp-014-013.shuffled-as.sat05-1229</td>
<td>T.O.</td>
<td>T.O.</td>
<td>T.O.</td>
<td>564.5</td>
</tr>
<tr>
<td>fphp-016-015.shuffled-as.sat05-1230</td>
<td>T.O.</td>
<td>T.O.</td>
<td>T.O.</td>
<td>1027.5</td>
</tr>
<tr>
<td>fphp-018-014.shuffled-as.sat05-1217</td>
<td>T.O.</td>
<td>T.O.</td>
<td>T.O.</td>
<td>549.1</td>
</tr>
<tr>
<td>fphp-020-015.shuffled-as.sat05-1218</td>
<td>T.O.</td>
<td>T.O.</td>
<td>T.O.</td>
<td>944.7</td>
</tr>
<tr>
<td>php-010-008.shuffled-as.sat05-1171</td>
<td>0.6</td>
<td>0.2</td>
<td>0.3</td>
<td>0.0</td>
</tr>
<tr>
<td>php-010-009.shuffled-as.sat05-1185</td>
<td>6.6</td>
<td>3.1</td>
<td>3.6</td>
<td>0.0</td>
</tr>
<tr>
<td>php-012-010.shuffled-as.sat05-1172</td>
<td>138.3</td>
<td>31.6</td>
<td>24.0</td>
<td>3.0</td>
</tr>
<tr>
<td>php-012-011.shuffled-as.sat05-1186</td>
<td>2696.0</td>
<td>1006.3</td>
<td>755.8</td>
<td>26.9</td>
</tr>
<tr>
<td>php-014-012.shuffled-as.sat05-1173</td>
<td>T.O.</td>
<td>T.O.</td>
<td>T.O.</td>
<td>237.5</td>
</tr>
</tbody>
</table>

SBVA solves >2,500x faster!
### Puzzle

<table>
<thead>
<tr>
<th>Problem Name</th>
<th>Baseline</th>
<th>BVA-ORIG</th>
<th>BVA-RAND-ORIG</th>
<th>BVA-RAND-3HOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>mp1-blockpuzzle_5x10_s2_free5</td>
<td>1326.3</td>
<td>159.4</td>
<td>352.9</td>
<td>129.2</td>
</tr>
<tr>
<td>mp1-blockpuzzle_5x10_s3_free4</td>
<td>1802.9</td>
<td>579.1</td>
<td>940.1</td>
<td>360.0</td>
</tr>
<tr>
<td>mp1-blockpuzzle_5x10_s5_free3</td>
<td>273.4</td>
<td>163.0</td>
<td>174.8</td>
<td>85.5</td>
</tr>
<tr>
<td>mp1-blockpuzzle_5x10_s7_free4</td>
<td>703.5</td>
<td>249.5</td>
<td>356.0</td>
<td>142.2</td>
</tr>
<tr>
<td>mp1-blockpuzzle_5x10_s8_free3</td>
<td>657.2</td>
<td>444.9</td>
<td>391.1</td>
<td>158.0</td>
</tr>
<tr>
<td>mp1-blockpuzzle_5x12_s6_free3</td>
<td>734.1</td>
<td>662.1</td>
<td>896.3</td>
<td>275.5</td>
</tr>
<tr>
<td>mp1-blockpuzzle_7x10_s10_free4</td>
<td>627.4</td>
<td>687.8</td>
<td>1017.4</td>
<td>232.1</td>
</tr>
<tr>
<td>mp1-blockpuzzle_7x10_s9_free6</td>
<td>527.2</td>
<td>89.0</td>
<td>156.0</td>
<td>53.6</td>
</tr>
<tr>
<td>mp1-blockpuzzle_8x8_s1_free4</td>
<td>649.9</td>
<td>295.2</td>
<td>427.8</td>
<td>191.9</td>
</tr>
<tr>
<td>mp1-blockpuzzle_9x9_s4_free3</td>
<td>2283.8</td>
<td>939.9</td>
<td>1276.8</td>
<td>680.3</td>
</tr>
<tr>
<td>mp1-blockpuzzle_9x9_s5_free3</td>
<td>529.5</td>
<td>851.5</td>
<td>760.1</td>
<td>199.0</td>
</tr>
<tr>
<td>mp1-blockpuzzle_9x9_s8_free3</td>
<td>994.5</td>
<td>1214.0</td>
<td>881.9</td>
<td>413.9</td>
</tr>
</tbody>
</table>
Reducing size only loosely correlates with speedup
Open-source Tool: SBVA

- MIT License
- C++
- Generates DRAT proofs

[GitHub Repository](https://github.com/hgarrereyn/SBVA)

**as a preprocessor**

```
./sbva -i formula.cnf -o reduced.cnf
```

**with a solver**

```
./sbva_wrapped <solver> <formula> <proof>
```
Summary

- Bounded Variable Addition is sensitive to randomization
- Structured BVA augments BVA with a heuristic to construct more effective variable additions
- Structured BVA is effective on a variety of formulas, and is beneficial to run as a preprocessor