PBS: A Pseudo-Boolean Solver and Optimizer

Fadi A. Aloul, Arathi Ramani, Igor L. Markov, Karem A. Sakallah
University of Michigan
Motivation ...

**SAT Solvers**
- Apps: Verification, Routing, ATPG, Timing Analysis
- Problem Type: CSP
- Problem Format: CNF
- Example: Chaff, GRASP, SATO

**Generic ILP Solvers**
- Apps: Routing, Planning, Scheduling
- Problem Type: CSP/Optimization
- Problem Format: ILP
- Example: CPLEX, LP_Solve

**Specialized 0/1 ILP Solvers**
- Apps: Verification, Routing, Binate Covering
- Problem Type: CSP/Optimization
- Problem Format: CNF/PB (0/1 ILP)
- Example: Satire, BSOL0, OPBDP, WSAT

© 2002 Fadi A. Aloul, University of Michigan
Motivation ...

- Many applications require "Counting Constraints" that impose upper/lower bounds on number of objects
- Introduce a new specialized 0-1 ILP SAT solver
- Describe Pseudo-Boolean (PB) search algorithms
- Adapt SAT applications expressed in pure CNF to CNF/PB format
- Empirically demonstrate effectiveness in EDA applications
Outline

- Boolean Satisfiability advances
- Processing Pseudo-Boolean constraints
- Applications
  - CSP
  - Optimization
- Experimental evaluation
- Conclusions
Backtrack Search (DPLL)

Init

Decision Engine
- Succeed
  - SAT
  - Deduction Engine
- Fail
  - Conflict
    - Exist
      - No
      - Yes
        - Diagnosis Engine
          - Fail
          - UNS
    - Succeed
Significantly improves the search performance

Classified as:
- Static
- Dynamic

Chaff introduced dynamic VSIDS:
- Shown to be effective on most benchmarks
- Selects most common literal and emphasizes variables in recent conflicts
**Improved BCP**

- Keeps track of any two unresolved literals in each clause instead of keeping track of all literals
- Leads to significant improvements over conventional BCP

[Moskewicz et al., Zhang et al.]
Conflict Diagnosis and Clause Deletion

- Add conflict-induced clauses to avoid regenerating similar conflicts in future parts of the search process
- Very effective in expediting the search process
- Allows non-chronological backtracking
- 1UIP learning scheme shown to perform best among other learning schemes [Zhang et al.]
Random Restarts and Backtracking

- Solver often gets stuck in local non-useful search space
- Random restarts periodically unassigns all decisions and randomly selects a new decision sequence
- Restarts ensures that different sub-trees are searched at every restart
- Randomization can be combined with backtracking
Outline

- Boolean Satisfiability advances
- Processing Pseudo-Boolean constraints
- Applications
  - CSP
  - Optimization
- Experimental evaluation
- Conclusions

© 2002 Fadi A. Aloul, University of Michigan
Pseudo-Boolean Constraints

\[ c_1 x_1 + \cdots + c_n x_n \sim g \]

\[ c_i, g \in Z \]

\[ \sim \in \{=, \leq, \geq\} \]

\[ x_i \in \text{Literals} \]

- Clauses can be generalized as a PB constraint: \((x + y) \Rightarrow (x + y \geq 1)\)

- None of the presented algorithms rely on the integrality of \(c_i\) and can be implemented for floating-point \(c_i\)
Motivating Example

- **Objective:**
  - limit the true assignments to \( k \) vars out of the \( n \) vars

- **Solution:**
  - **CNF:**
    \[
    \begin{pmatrix} n \choose k+1 \end{pmatrix} \text{ clauses}
    \]
    Each of size \((k+1)\)
  - **PB:** single PB constraint

- “at most 2 out of \( v_1, v_2, v_3, v_4, v_5 \), can be true”
  - **Pure CNF:**
    \[
    (\overline{v}_1 + \overline{v}_2 + v_3) \cdot (v_1 + \overline{v}_2 + v_4) \cdot \\
    (v_1 + \overline{v}_2 + \overline{v}_5) \cdot (\overline{v}_1 + v_3 + v_4) \cdot \\
    (v_1 + v_3 + \overline{v}_5) \cdot (\overline{v}_1 + v_4 + \overline{v}_5) \cdot \\
    (v_2 + v_3 + \overline{v}_4) \cdot (v_2 + \overline{v}_3 + v_5) \cdot \\
    (\overline{v}_2 + \overline{v}_4 + v_5) \cdot (v_3 + v_4 + v_5)
    \]
  - **PB form:**
    \[
    (1v_1 + 1v_2 + 1v_3 + 1v_4 + 1v_5 \leq 2)
    \]
PB Constraint Data Structure

Struct PBConstraint {
  Goal n; constraint type ~; list of c_i and x_i’s;
  initLHS; // sum of all c_i’s
  LHS;     // value of LHS based on current variable assignment
  maxLHS;  // maximal possible value of LHS given the current variable assignment
}

For efficiency:
- Sort the list of c_i x_i in order of increasing c_i
- Convert all negative c_i to positive:
  i.e. \( c_1 x_1 - c_2 x_2 \leq n \)
  \( c_1 x_1 - c_2 (1 - \bar{x}_2) \leq n \)
  \( c_1 x_1 + c_2 \bar{x}_2 \leq n + c_2 \)

© 2002 Fadi A. Aloul, University of Michigan
Assigning \( v_i \) to 1:
For each literal \( x_i \) of \( v_i \)
- If positive \( x_i \), \( \text{LHS} += c_i \)
- If negative \( x_i \), \( \text{maxLHS} -= c_i \)

Unassigning \( v_i \) from 1:
For each literal \( x_i \) of \( v_i \)
- If positive \( x_i \), \( \text{LHS} -= c_i \)
- If negative \( x_i \), \( \text{maxLHS} += c_i \)

PB constraint state:
- \( \geq \) type
  - \( \text{SAT} \): \( \text{LHS} \geq \text{goal} \)
  - \( \text{UNS} \): \( \text{maxLHS} < \text{goal} \)
- \( \leq \) type
  - \( \text{SAT} \): \( \text{maxLHS} \leq \text{goal} \)
  - \( \text{UNS} \): \( \text{LHS} > \text{goal} \)

5\(x_1+6x_2+3x_3 \leq 12 \)

<table>
<thead>
<tr>
<th>LHS</th>
<th>maxLHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>

SATISFIABLE
Algorithms for PB Search

- Identifying implications
  - $\leq$ type
    - if $c_i > \text{goal} - \text{LHS}$, $x_i = 0$
    - Implied by literals in PB assigned to 1
  - $\geq$ type
    - if $c_i > \text{maxLHS} - \text{goal}$, $x_i = 1$
    - Implied by literals in PB assigned to 0

\[
5x_1 + 6x_2 + 3x_3 \leq 12
\]

$\text{LHS} = 0$
$\text{maxLHS} = 14$
$\text{goal} - \text{LHS} = 12$

\[
5x_1 + 6x_2 + 3x_3 \leq 12
\]

$\text{LHS} = 8$
$\text{maxLHS} = 14$
$\text{goal} - \text{LHS} = 4$
Imply $x_2 = 0$
Algorithms for PB Search

- Identifying implications
  - $\leq$ type
    - if $c_i > \text{goal} - \text{LHS}$, $x_i = 0$
    - Implied by literals in PB assigned to 1
  - $\geq$ type
    - if $c_i > \text{maxLHS} - \text{goal}$, $x_i = 1$
    - Implied by literals in PB assigned to 0

$$5x_1+6x_2+3x_3 \geq 10$$

LHS = 0
maxLHS = 14
maxLHS - goal = 4
Imply $x_1=x_2=1$

© 2002 Fadi A. Aloul, University of Michigan
Outline

- Boolean Satisfiability advances
- Processing Pseudo-Boolean constraints
- Applications
  - CSP
  - Optimization
- Experimental evaluation
- Conclusions

© 2002 Fadi A. Aloul, University of Michigan
Applications - CSP

- Global Routing

- 2-D grid of cells arranged in rows/columns
- Cell boundaries are edges
- Capacity $C$ is associated with each edge (no more than $C$ routes can pass)
- Goal: route number of 2-pin connections in the grid with edge capacities
- Generate satisfiable instances using randomized flooding
Global Routing Formulation

- **Connectivity constraints** (for each net)
  - Exactly one edge selected at start/end point
  - If cell is a mid-point, either two or no edges are selected

- **Capacity constraints**
  - A net can use a single track across an edge
  - No two nets can use the same track across an edge

- Create a variable for each edge/net:
  \[ (vN + vW)(vN + vE)(vW + vE) \]
  \[ (vW + vN + vE) \]

\[ 2 \times 12 = 24 \text{ variables} \]
Global Routing Formulation

- Connectivity constraints (for each net)
  - Exactly one edge selected at start/end point
  - If cell is a mid-point, either two or no edges are selected

- Capacity constraints
  - A net can use a single track across an edge
  - No two nets can use the same track across an edge

- Create a variable for each edge/net
  \[ 2 \times 12 = 24 \text{ variables} \]

\[
(vN + vE + vW)(vN + \overline{vE} + vW) \\
(vN + vE + \overline{vW})(\overline{vN} + \overline{vE} + \overline{vW})
\]
Global Routing Formulation

- Connectivity constraints (for each net)
  - Exactly one edge selected at start/end point
  - If cell is a mid-point, either two or no edges are selected

- Capacity constraints
  - A net can use a single track across an edge
  - No two nets can use the same track across an edge

Create a variable for each edge/net
2 x 12 = 24 variables

30 Nets
10 Cap
#Cl = 55M
vs.
1 PB

© 2002 Fadi A. Aloul, University of Michigan
Global Routing Formulation

- Connectivity constraints (for each net)
  - Exactly one edge selected at start/end point
  - If cell is a mid-point, either two or no edges are selected

- Capacity constraints
  - A net can use a single track across an edge
  - No two nets can use the same track across an edge

- Additional Variables & Clauses
  - Create $Cap$ variables per edge/net
    - $2 \times 2 \times 12 = 48$ variables

$$
\#Cl = \left(\#Nets\right)\left(\frac{Cap}{2}\right) + \left(Cap\right)\left(\frac{\#Nets}{2}\right)
$$

© 2002 Fadi A. Aloul, University of Michigan
Applications - optimization

- Max-ONEs
  - Seeks an assignment that
    - Satisfies all constraints
    - Maximizes the number of variables assigned to true
  - Useful to represent “Max-Clique” problems
  - “Vertex Cover” can be reduced to Min-ONEs
  - Use a single PB constraint of type \( \geq \) that includes each variable with coefficient “1”
  - Iteratively increase the lower bound until the problem becomes unsatisfiable
  - Extendable to “Weighted Max-ONEs”
Applications - optimization

- Max-SAT
  - Finds an assignment that
    - Satisfies maximum possible number of clauses
  - Generalization of SAT
    - Provides more info for unsatisfiable instances
  - Used to represent “Max-CUT” problems
  - Expressed using a single PB constraint
  - Solved using PBS
  - Addressed indirectly using WalkSAT
Outline

- Boolean Satisfiability advances
- Processing Pseudo-Boolean constraints
- Applications
  - CSP
  - Optimization
- Experimental evaluation
- Conclusions
Experimental Setup

- Platform: Pentium-II 300 MHz with 512MB RAM running Linux
- Runtime limit: 5000 sec
- PBS Implemented in C++
- PBS settings:
  - VSIDS decision heuristic
  - Optimized BCP
  - Random Restarts
  - 1st UIP conflict analysis learning scheme
  - Clause deletion/random backtracking disabled
### Global Routing Experiment

<table>
<thead>
<tr>
<th>Instance</th>
<th>CNF + pseudo-Boolean</th>
<th>pure CNF</th>
<th>PBS Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>V</td>
<td>C</td>
<td>#PB</td>
</tr>
<tr>
<td>grout3.3-1</td>
<td>216</td>
<td>572</td>
<td>12</td>
</tr>
<tr>
<td>grout3.3-2</td>
<td>264</td>
<td>700</td>
<td>12</td>
</tr>
<tr>
<td>grout3.3-3</td>
<td>240</td>
<td>636</td>
<td>12</td>
</tr>
<tr>
<td>grout3.3-4</td>
<td>228</td>
<td>604</td>
<td>12</td>
</tr>
<tr>
<td>grout3.3-5</td>
<td>240</td>
<td>634</td>
<td>12</td>
</tr>
<tr>
<td>grout4.3-1</td>
<td>672</td>
<td>2004</td>
<td>24</td>
</tr>
<tr>
<td>grout4.3-2</td>
<td>648</td>
<td>1928</td>
<td>24</td>
</tr>
<tr>
<td>grout4.3-3</td>
<td>648</td>
<td>1930</td>
<td>24</td>
</tr>
<tr>
<td>grout4.3-4</td>
<td>696</td>
<td>2072</td>
<td>24</td>
</tr>
<tr>
<td>grout4.3-5</td>
<td>720</td>
<td>2144</td>
<td>24</td>
</tr>
<tr>
<td>grout4.3-6</td>
<td>624</td>
<td>1860</td>
<td>24</td>
</tr>
<tr>
<td>grout4.3-7</td>
<td>672</td>
<td>2006</td>
<td>24</td>
</tr>
<tr>
<td>grout4.3-8</td>
<td>432</td>
<td>1280</td>
<td>24</td>
</tr>
<tr>
<td>grout4.3-9</td>
<td>840</td>
<td>2502</td>
<td>24</td>
</tr>
<tr>
<td>grout4.3-10</td>
<td>840</td>
<td>2504</td>
<td>24</td>
</tr>
</tbody>
</table>

© 2002 Fadi A. Aloul, University of Michigan
# MaxONE Experiment

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Satisfiable Instance</th>
<th>V</th>
<th>C</th>
<th>Max-ONES</th>
<th>PBS</th>
<th>SATIRE</th>
<th>OPBDP</th>
<th>PBS Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DIMACS</strong></td>
<td>aim-50-1_6-yes1-1</td>
<td>50</td>
<td>80</td>
<td>29</td>
<td>0.01</td>
<td>0.01</td>
<td>0.02</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>aim-100-1_6-yes1-1</td>
<td>100</td>
<td>160</td>
<td>43</td>
<td>0.01</td>
<td>0.02</td>
<td>7.19</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>aim-200-2_0-yes1_1</td>
<td>200</td>
<td>400</td>
<td>96</td>
<td>0.01</td>
<td>0.06</td>
<td>5000</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>ii8b1</td>
<td>336</td>
<td>2068</td>
<td>275</td>
<td>4.69</td>
<td>3180</td>
<td>56.2</td>
<td>678</td>
</tr>
<tr>
<td></td>
<td>jnh1</td>
<td>100</td>
<td>850</td>
<td>55</td>
<td>0.32</td>
<td>2.2</td>
<td>0.12</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>jnh204</td>
<td>100</td>
<td>800</td>
<td>58</td>
<td>0.28</td>
<td>1.63</td>
<td>0.14</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>par8-1</td>
<td>350</td>
<td>1149</td>
<td>79</td>
<td>0.01</td>
<td>0.06</td>
<td>0.05</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>par8-2-c</td>
<td>68</td>
<td>270</td>
<td>20</td>
<td>0.01</td>
<td>0.02</td>
<td>0.01</td>
<td>2</td>
</tr>
<tr>
<td><strong>Beijing</strong></td>
<td>3blocks</td>
<td>283</td>
<td>9690</td>
<td>63</td>
<td>4.83</td>
<td>49.53</td>
<td>4494</td>
<td>10</td>
</tr>
<tr>
<td><strong>QG</strong></td>
<td>qg7-09</td>
<td>729</td>
<td>22060</td>
<td>81</td>
<td>0.1</td>
<td>5.41</td>
<td>9.8</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>qg6-09</td>
<td>729</td>
<td>21844</td>
<td>81</td>
<td>0.21</td>
<td>5.56</td>
<td>45</td>
<td>26</td>
</tr>
<tr>
<td><strong>Planning</strong></td>
<td>bw_a</td>
<td>459</td>
<td>4675</td>
<td>73</td>
<td>0.03</td>
<td>0.43</td>
<td>0.21</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>bw_b</td>
<td>1087</td>
<td>13772</td>
<td>136</td>
<td>0.58</td>
<td>6.39</td>
<td>17.86</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>bw_c</td>
<td>3016</td>
<td>50457</td>
<td>272</td>
<td>24.37</td>
<td>315.5</td>
<td>5000</td>
<td>13</td>
</tr>
</tbody>
</table>

© 2002 Fadi A. Aloul, University of Michigan
Conclusions

- Adapting SAT apps to use CNF/PB constraints leads to memory savings and runtime reductions
- Proposed new specialized 0-1 ILP solver, PBS
- Confirmed effectiveness on real world examples:
  - Global routing *consistency* instances
  - Max-ONEs *optimization* problems (extendable to Max-SAT, Min-ONEs)
Future Works

- Compare state-of-the-art Generic ILP solvers, such as CPLEX, to specialized 0-1 ILP solvers
- Apply PBS to Max-SAT and Min-ONEs problems
- Study applications to Max-Clique, Max Independent Set, and Min Vertex Cover