Improvements to Core-Guided Binary Search for MaxSAT

A.Morgado  F.Heras  J.Marques-Silva

CASL/CSI, University College Dublin
Dublin, Ireland

SAT, June 2012
**MaxSAT**

**Maximum Satisfiability** (MaxSAT) Given a propositional formula in Conjunctive Normal Form (CNF), compute an assignment to the variables that maximizes the number of satisfied clauses

\[(x_1) \land (\neg x_1 \lor \neg x_2) \land (x_2)\]

The instance is unsatisfiable, and the MaxSAT solution is 2
Maximum Satisfiability (MaxSAT) Given a propositional formula in Conjunctive Normal Form (CNF), compute an assignment to the variables that maximizes the number of satisfied clauses

\[(x_1) \land (\neg x_1 \lor \neg x_2) \land (x_2)\]

The instance is unsatisfiable, and the MaxSAT solution is 2

MaxSAT as MinUNSAT - MaxSAT solvers compute the minimum number of falsified clauses (1 in the example)
MaxSAT

Maximum Satisfiability (MaxSAT) Given a propositional formula in Conjunctive Normal Form (CNF), compute an assignment to the variables that maximizes the number of satisfied clauses

\[(x_1) \land (\neg x_1 \lor \neg x_2) \land (x_2)\]

The instance is unsatisfiable, and the MaxSAT solution is 2

MaxSAT as MinUNSAT - MaxSAT solvers compute the minimum number of falsified clauses (1 in the example)

There are 3 variants of the MaxSAT problem
MaxSAT Variants

- **Partial MaxSAT**
  - Set of clauses divided in two:
    - Hard clauses - have to be satisfied
    - Soft clauses - satisfied or not
  - Objective: Minimize the number of falsified soft clauses while satisfying all hard clauses

- **Weighted MaxSAT**
  - Clauses have weights associated that represent the cost of falsifying the clause
  - Objective: Minimize the sum of the weights of the falsified clauses

- **Weighted Partial MaxSAT**
  - Combines the previous two
  - Objective: Minimize the sum of the weights of the falsified soft clauses while satisfying all the hard clauses
MaxSAT Variants

- **Partial MaxSAT**
  - Set of clauses divided in two:
    - Hard clauses - have to be satisfied
    - Soft clauses - satisfied or not
  - Objective: Minimize the number of falsified soft clauses while satisfying all hard clauses

- **Weighted MaxSAT**
  - Clauses have weights associated that represent the cost of falsifying the clause
  - Objective: Minimize the sum of the weights of the falsified clauses

- **Weighted Partial MaxSAT**
  - Combines the previous two
  - Objective: Minimize the sum of the weights of the falsified soft clauses while satisfying all the hard clauses
MaxSAT Variants

- **Partial MaxSAT**
  - Set of clauses divided in two:
    - Hard clauses - have to be satisfied
    - Soft clauses - satisfied or not
  - Objective: Minimize the number of falsified soft clauses while satisfying all hard clauses

- **Weighted MaxSAT**
  - Clauses have weights associated that represent the cost of falsifying the clause
  - Objective: Minimize the sum of the weights of the falsified clauses

- **Weighted Partial MaxSAT**
  - Combines the previous two
  - Objective: Minimize the sum of the weights of the falsified soft clauses while satisfying all the hard clauses
MaxSAT Variants

- **Partial MaxSAT**
  - Set of clauses divided in two:
    - Hard clauses - have to be satisfied
    - Soft clauses - satisfied or not
  - Objective: Minimize the number of falsified soft clauses while satisfying all hard clauses

- **Weighted MaxSAT**
  - Clauses have weights associated that represent the cost of falsifying the clause
  - Objective: Minimize the sum of the weights of the falsified clauses

- **Weighted Partial MaxSAT**
  - Combines the previous two
  - Objective: Minimize the sum of the weights of the falsified soft clauses while satisfying all the hard clauses
MaxSAT Problem

- **Complexity:**
  - Decision version of MaxSAT is NP-Complete
  - MaxSAT is $\Delta_2^p$–complete

- Many optimization problems can be *modeled* as MaxSAT: Bayesian Networks, Physics (Spinglass), BioInformatics (haplotyping, protein alignment), Circuit Debugging, Fault Localization, Graph problems (Max-Cut, Max-Clique), etc.
Motivation

- Study MaxSAT algorithms based on iteratively calling a SAT solver
  - most of them require an exponential number of calls to the SAT solver
Motivation

- Study MaxSAT algorithms based on iteratively calling a SAT solver
  - most of them require an exponential number of calls to the SAT solver
  - Binary Search requires a linear number of calls to the SAT solver

[fm-SAT06]
Motivation

• Study MaxSAT algorithms based on iteratively calling a SAT solver
  – most of them require an exponential number of calls to the SAT solver
  – **Binary Search** requires a **linear** number of calls to the SAT solver
    [fm-SAT06]

• Extend binary search for MaxSAT:
  – **BC** - **Core-guided binary search**: Relax soft clauses on demand as dictated by the unsatisfiable cores found so far
    [hmms-AAAI11]
  – **BCD** - **Core-guided binary search with disjoint cores**: Maintaining disjoint cores allows adding smaller constraints
    [hmms-AAAI11]

  – **BCD2** - Improvement of BCD: New bounds, better maintenance of the bounds
    [mhms-SAT12]
Computing a MaxSAT Solution - Notation

- \( \varphi \) - **Unsatisfiable** Weighted CNF Formula: 
  \((c_i, w_i)\) weighted clause:
  - \( c_i \) is a clause
  - \( w_i \) is a non-negative integer or \( \top \);
    cost of falsifying \( c_i \)

- \( \mathcal{A} \) - Complete Assignment: \( \mathcal{A} = \{x_1 = 1, x_2 = 0, \ldots\} \)
  Cost of assignment: Sum of weights of falsified clauses

- OPT - Optimum cost: **Minimum** cost

- \( \mu \) - Upper Bound: Cost not lower than OPT

- \( \lambda \) - Lower Bound: Value lower than OPT

\[ \lambda \quad \mu \]

OPT
Motivation

Previous Algorithms - Overview

BCD2

Optimizations

Results

Conclusions
Binary Search (1/2)

Proposed by Fu&Malik SAT2006

\[(c_i, w_i)\quad ((x_1), 2)\quad ((x_2), 5)\]
\[((\neg x_1 \lor \neg x_2), \top)\]
Binary Search (1/2)

Proposed by Fu&Malik SAT2006

\[(c_i, w_i)\] \[\Downarrow\] \[(c_i \lor r_i, w_i)\] \[\Downarrow\] \[((x_1 \lor r_1), 2) \ (x_2 \lor r_2), 5)\] \[\Downarrow\] \[((\neg x_1 \lor \neg x_2), \top)\]
Binary Search (1/2)

Proposed by Fu & Malik SAT2006

\[ (c_i, w_i) \]
\[ \downarrow \]
\[ (c_i \lor r_i, w_i) \]

\[ ((x_1), 2) \]
\[ ((x_2), 5) \]
\[ (((\neg x_1 \lor \neg x_2), \top) \]

\[ \downarrow \]
\[ ((x_1 \lor r_1), 2) \]
\[ ((x_2 \lor r_2), 5) \]
\[ (((\neg x_1 \lor \neg x_2), \top) \]

If \( A \not\models c_i \) then \( A \models r_i \).
Binary Search (1/2)

Proposed by Fu&Malik SAT2006

\[(c_i, w_i) \quad \overset{\text{\downarrow}}{\sim} \quad ((x_1), 2) \quad ((x_2), 5) \quad (((\neg x_1 \lor \neg x_2), \top))\]

\[(c_i \lor r_i, w_i) \quad \overset{\text{\downarrow}}{\sim} \quad ((x_1 \lor r_1), 2) \quad ((x_2 \lor r_2), 5) \quad (((\neg x_1 \lor \neg x_2), \top))\]

If \(A \not\models c_i\) then \(A \models r_i\)

Minimize sum of weights of true relaxation variables

\[\Leftrightarrow\]

Minimize sum of weights of falsified soft clauses
In each iteration:

- Consider $\tau$ - maximum allowed sum of weights of true r.v.
In each iteration:

- Consider $\tau - \text{maximum allowed sum of weights of true r.v.}$

$$\tau = \left\lfloor \frac{\mu + \lambda}{2} \right\rfloor$$
In each iteration:

- Consider $\tau$ - maximum allowed sum of weights of true r.v.

$$\tau = \left\lfloor \frac{\mu + \lambda}{2} \right\rfloor$$

- Add PB-constraint

$CNF(\sum w_i \ r_i \leq \tau)$
In each iteration:
- Consider $\tau$ - maximum allowed sum of weights of true r.v.
  \[ \tau = \left\lfloor \frac{\mu + \lambda}{2} \right\rfloor \]
- Add PB-constraint
  \[ CNF(\sum w_i \ r_i \leq \tau) \]

$$((x_1 \lor r_1), 2) \quad ((x_2 \lor r_2), 5)$$
$$((\neg x_1 \lor \neg x_2), \top)$$

$$\mu_0 = 8 \quad \lambda_0 = -1$$

OPT = 2

$\lambda = -1$  $\mu = 8$
Binary Search (2/2)

In each iteration:
- Consider $\tau$ - maximum allowed sum of weights of true r.v.
  $$\tau = \lfloor \frac{\mu + \lambda}{2} \rfloor$$
- Add PB-constraint
  $$CNF(\sum w_i \ r_i \leq \tau)$$

$$((x_1 \lor r_1), \ 2) \ \ ((x_2 \lor r_2), \ 5)$$
$$((\neg x_1 \lor \neg x_2), \ 1)$$

$$\mu_0 = 8 \ \ \lambda_0 = -1$$

$OPT = 2$

$\lambda = -1 \ \ \tau = \lfloor \frac{\lambda + \mu}{2} \rfloor = 3 \ \ \ \mu = 8$
In each iteration:

- Consider \( \tau \) - maximum allowed sum of weights of true r.v.
  \[
  \tau = \left\lfloor \frac{\mu + \lambda}{2} \right\rfloor
  \]

- Add PB-constraint
  \[
  CNF(\sum w_i \ r_i \leq \tau)
  \]

\[
((x_1 \lor r_1), \ 2) \ (\ (x_2 \lor r_2), \ 5) \\
((\neg x_1 \lor \neg x_2), \ \top)
\]

\[
\mu_1 = 3 \quad \lambda_1 = -1
\]

\[\text{OPT} = 2\]

\[\lambda \quad \tau \quad \mu = 3\]
In each iteration:
- Consider $\tau$ - maximum allowed sum of weights of true r.v.
  \[
  \tau = \lfloor \frac{\mu + \lambda}{2} \rfloor
  \]
- Add PB-constraint
  \[
  \text{CNF} \left( \sum w_i \ r_i \leq \tau \right)
  \]

\[
((x_1 \lor r_1), 2) \quad ((x_2 \lor r_2), 5) \\
((\neg x_1 \lor \neg x_2), \top)
\]

$\mu_2 = 3$  $\lambda_2 = 1$

$\text{OPT} = 2$
In each iteration:

- Consider $\tau$ - maximum allowed sum of weights of true r.v.

\[ \tau = \lfloor \frac{\mu + \lambda}{2} \rfloor \]

- Add PB-constraint CNF($\sum w_i \cdot r_i \leq \tau$)

\[ ((x_1 \lor r_1), 2) \quad ((x_2 \lor r_2), 5) \]
\[ ((\neg x_1 \lor \neg x_2), \top) \]

\[ \mu_3 = 2 \quad \lambda_3 = 1 \]

OPT = 2
BC - Core-Guided Binary Search

- Binary Search relaxes all soft clauses
- State of the art SAT solvers report Unsatisfiable Cores on unsatisfiable formulas
  - $\varphi_C$ - Unsatisfiable Core: subset of unsatisfiable clauses
BC - Core-Guided Binary Search

- Binary Search relaxes all soft clauses
- State of the art SAT solvers report Unsatisfiable Cores on unsatisfiable formulas
  
  - $\varphi_C$ - Unsatisfiable Core: subset of unsatisfiable clauses

- BC only relaxes a clause when it belongs to an unsatisfiable core
  
  - Initially no clause is relaxed
  - On every unsatisfiable iteration obtain the core $\varphi_C$
  - If $\varphi_C$ contains unrelaxed soft clauses clauses
    
    then relax unrelaxed soft clauses
  
  else proceed as Binary Search
BC - Core-Guided Binary Search

- Binary Search relaxes all soft clauses
- State of the art SAT solvers report Unsatisfiable Cores on unsatisfiable formulas
  - $\varphi_C$ - Unsatisfiable Core: subset of unsatisfiable clauses
- BC only relaxes a clause when it belongs to an unsatisfiable core
  - Initially no clause is relaxed
  - On every unsatisfiable iteration obtain the core $\varphi_C$
  - If $\varphi_C$ contains unrelaxed soft clauses clauses
    then relax unrelaxed soft clauses
  else proceed as Binary Search

\[((x_1, 2), (x_2), 5), ((x_6), 1), ((\neg x_1 \lor \neg x_2), \top)\]
Outline

Motivation

Previous Algorithms - Overview

BCD2

Optimizations

Results

Conclusions
BCD/BCD2 - Core-Guided Binary Search with Disjoint Cores

- BC maintains one “large” PB-constraint
- Some cores do not intersect other cores throughout the iterations
BCD/BCD2 - Core-Guided Binary Search with Disjoint Cores

- BC maintains one "large" PB-constraint
- Some cores do not intersect other cores throughout the iterations

- BCD/BCD2 maintain disjoint cores separate:
  - Each disjoint core $C_i$ is a core with its own:
    - $\mu_i$ - upper bound
    - $\lambda_i$ - lower bound
    - $R_i$ - set of relaxation variables
    - $\tau_i$ - middle value
  - Each disjoint core $C_i$ adds a PB-constraint to the formula

$$CNF(\sum_{r_j \in R_i} w_j r_j \leq \tau_i)$$
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(x₁, 1)</td>
<td></td>
<td>(x₆, 1)</td>
</tr>
<tr>
<td>(x₅, 1)</td>
<td></td>
<td>(x₂, 1)</td>
<td></td>
</tr>
<tr>
<td>(x₄, 1)</td>
<td></td>
<td>(x₃, 1)</td>
<td></td>
</tr>
</tbody>
</table>

\[(¬x₁ \lor ¬x₂, T)\]
\[(¬x₂ \lor ¬x₃, T)\]
\[(¬x₃ \lor ¬x₄, T)\]
\[(¬x₄ \lor ¬x₅, T)\]
\[(¬x₅ \lor ¬x₁, T)\]
BCD/BCD2 on Example

\[(x_1, 1)\]

\[(x_5, 1)\]

\[(x_4, 1)\]

\[(x_2, 1)\]

\[(x_3, 1)\]

\[(x_6, 1)\]

\[\neg x_6, 1\]

\[\neg x_1 \lor \neg x_2, T\]

\[\neg x_2 \lor \neg x_3, T\]

\[\neg x_3 \lor \neg x_4, T\]

\[\neg x_4 \lor \neg x_5, T\]

\[\neg x_5 \lor \neg x_1, T\]
BCD/BCD2 on Example

\[ (x_1, 1) \]

\[ (x_5, 1) \]

\[ (x_4, 1) \]

\[ (x_2, 1) \]

\[ (x_3, 1) \]

\[ (x_6 \lor r_7, 1) \]

\[ \neg x_6 \lor r_7, 1 \]

\[ \mu = 3 \]
\[ \lambda = 0 \]
\[ \tau = 1 \]
\[ CNF(r_6 + r_7 \leq 1) \]

\[ (\neg x_1 \lor \neg x_2, T) \]
\[ (\neg x_2 \lor \neg x_3, T) \]
\[ (\neg x_3 \lor \neg x_4, T) \]
\[ (\neg x_4 \lor \neg x_5, T) \]
\[ (\neg x_5 \lor \neg x_1, T) \]
BCD/BCD2 on Example

\( \mu = 3 \)
\( \lambda = 0 \)
\( \tau = 1 \)

\( \text{CNF}(r_6 + r_7 \leq 1) \)
$BCD/BCD2$ on Example

$CNF(r_1 + r_2 \leq 1) \quad (x_1 \lor r_1, 1)$

$CNF(r_6 + r_7 \leq 1) \quad (x_6 \lor r_6, 1)$

$(x_5, 1)$

$(x_4, 1)$

$(x_2 \lor r_2, 1)$

$(x_3, 1)$

$(\neg x_1 \lor \neg x_2, 1)$

$(\neg x_2 \lor \neg x_3, 1)$

$(\neg x_3 \lor \neg x_4, 1)$

$(\neg x_4 \lor \neg x_5, 1)$

$(\neg x_5 \lor \neg x_1, 1)$

$\mu = 3$

$\lambda = 0$

$\tau = 1$
BCD/BCD2 on Example

\[
\begin{align*}
(x_1 \lor r_1, 1) \\
(x_2 \lor r_2, 1) \\
(x_3, 1) \\
(x_4, 1) \\
(x_5, 1)
\end{align*}
\]

\[
\begin{align*}
\mu &= 3 \\
\lambda &= 0 \\
\tau &= 1
\end{align*}
\]

\[
\text{CNF}(r_1 + r_2 \leq 1)
\]

\[
\begin{align*}
(x_6 \lor r_6, 1) \\
(x_6, 1)
\end{align*}
\]

\[
\begin{align*}
\mu &= 3 \\
\lambda &= 0 \\
\tau &= 1
\end{align*}
\]

\[
\text{CNF}(r_6 + r_7 \leq 1)
\]

\[
\begin{align*}
(\neg x_1 \lor \neg x_2, \top) \\
(\neg x_2 \lor \neg x_3, \top) \\
(\neg x_3 \lor \neg x_4, \top) \\
(\neg x_4 \lor \neg x_5, \top) \\
(\neg x_5 \lor \neg x_1, \top)
\end{align*}
\]
BCD/BCD2 on Example

\[(x_1 \lor r_1, 1)\]
\[(x_5, 1)\]
\[\text{CNF}(r_1 + r_2 \leq 1)\]
\[\mu = 3\]
\[\lambda = 0\]
\[\tau = 1\]

\[(x_2 \lor r_2, 1)\]
\[\mu = 3\]
\[\lambda = 0\]
\[\tau = 1\]

\[(x_4 \lor r_4, 1)\]
\[\mu = 3\]
\[\lambda = 0\]
\[\tau = 1\]
\[\text{CNF}(r_3 + r_4 \leq 1)\]

\[(x_6 \lor r_6, 1)\]
\[\mu = 3\]
\[\lambda = 0\]
\[\tau = 1\]
\[\text{CNF}(r_6 + r_7 \leq 1)\]

\[\neg x_1 \lor \neg x_2, \top\]
\[\neg x_2 \lor \neg x_3, \top\]
\[\neg x_3 \lor \neg x_4, \top\]
\[\neg x_4 \lor \neg x_5, \top\]
\[\neg x_5 \lor \neg x_1, \top\]
BCD/BCD2 on Example

\[ \mu = 3 \]
\[ \lambda = 0 \]
\[ \tau = 1 \]

\[ CNF(r_1 + r_2 \leq 1) \]
\[ CNF(r_3 + r_4 \leq 1) \]

\[ \neg x_1 \lor \neg x_2, \top \]
\[ \neg x_2 \lor \neg x_3, \top \]
\[ \neg x_3 \lor \neg x_4, \top \]
\[ \neg x_4 \lor \neg x_5, \top \]
\[ \neg x_5 \lor \neg x_1, \top \]

\[ (\neg x_6 \lor r_7, 1) \]
\[ (\neg x_6 \lor r_7, 1) \]

\[ CNF(r_6 + r_7 \leq 1) \]
BCD/BCD2 on Example

\[
\begin{align*}
(x_1 \lor r_1, 1) \\
(x_2 \lor r_2, 1) \\
(x_3 \lor r_3, 1) \\
(x_4 \lor r_4, 1) \\
(x_5 \lor r_5, 1) \\
(x_6 \lor r_6, 1) \\
(\neg x_1 \lor \neg x_2, \top) \\
(\neg x_2 \lor \neg x_3, \top) \\
(\neg x_3 \lor \neg x_4, \top) \\
(\neg x_4 \lor \neg x_5, \top) \\
(\neg x_5 \lor \neg x_1, \top)
\end{align*}
\]

\[CNF(r_1 + r_2 + r_3 + r_4 + r_5 \leq 3)\]

\[CNF(r_6 + r_7 \leq 1)\]

\[\mu = 3\]

\[\lambda = 0\]

\[\tau = 1\]
BCD/BCD2 on Example

$$\begin{align*}
\text{CNF}(r_1 + r_2 + r_3 + r_4 + r_5 &\leq 3) \\
\mu = 6 &\quad \lambda = 0 &\quad \tau = 3 \\
\text{CNF}(r_6 + r_7 &\leq 1) \\
\mu = 3 &\quad \lambda = 0 &\quad \tau = 1
\end{align*}$$

$$\begin{align*}
\neg x_1 \lor \neg x_2, T \\
\neg x_2 \lor \neg x_3, T \\
\neg x_3 \lor \neg x_4, T \\
\neg x_4 \lor \neg x_5, T \\
\neg x_5 \lor \neg x_1, T
\end{align*}$$
BCD/BCD2 on Example

\[
\begin{align*}
\mu &= 1 \\
\lambda &= 0 \\
\tau &= 1 \\
CNF(\mu + \lambda + \tau &\leq 1) \\
CNF(r_1 + r_2 + r_3 + r_4 + r_5 &\leq 1)
\end{align*}
\]
BCD/BCD2 on Example

\[CNF(r_1 + r_2 + r_3 + r_4 + r_5 \leq 1)\]

\[CNF(r_6 + r_7 \leq 1)\]
BCD/BCD2 on Example

\[

cnf\left(r_6 + r_7 \leq 1\right)
\]

\[

cnf\left(r_1 + r_2 + r_3 + r_4 + r_5 \leq 2\right)
\]
BCD/BCD2 on Example

\( \mu = 3 \)
\( \lambda = 1 \)
\( \tau = 2 \)

\( \text{CNF}(r_{1} + r_{2} + r_{3} + r_{4} + r_{5} \leq 2) \)

\( \neg x_{1} \lor \neg x_{2}, \top \)
\( \neg x_{2} \lor \neg x_{3}, \top \)
\( \neg x_{3} \lor \neg x_{4}, \top \)
\( \neg x_{4} \lor \neg x_{5}, \top \)
\( \neg x_{5} \lor \neg x_{1}, \top \)

\( \mu = 1 \)
\( \lambda = 0 \)
\( \tau = 1 \)

\( \text{CNF}(r_{6} + r_{7} \leq 1) \)
BCD/BCD2 on Example

\[(x_1 \lor r_1, 1)\]
\[(x_5 \lor r_5, 1)\]
\[(x_4 \lor r_4, 1)\]
\[(x_2 \lor r_2, 1)\]
\[(x_3 \lor r_3, 1)\]

\[\neg x_1 \lor \neg x_2, \top\]
\[\neg x_2 \lor \neg x_3, \top\]
\[\neg x_3 \lor \neg x_4, \top\]
\[\neg x_4 \lor \neg x_5, \top\]
\[\neg x_5 \lor \neg x_1, \top\]

\[\mu = 3\]
\[\lambda = 2\]

\[\mu = 1\]
\[\lambda = 0\]
BCD2 - Improvements

- Improvements over BCD
  - On the maintenance of upper bound
    - Cost of Clause
    - Global Upper Bound
  - On the maintenance of lower bound
    - New Lower Bound
BCD2 - Improvements on the upper bound

- **Cost of Clause**
  - In BCD the cost of a clause consider the relaxation variable
    \[ w_i \cdot A(r_i) \]
  - In BCD2 the cost of a clause **disregards** the relaxation variable
    \[ w_i \cdot (1 - A(c_i \setminus \{r_i\})) \]
BCD2 - Improvements on the upper bound

- **Cost of Clause**
  - In BCD the cost of a clause consider the relaxation variable
    ▶ $w_i \cdot A(r_i)$
  - In BCD2 the cost of a clause **disregards** the relaxation variable
    ▶ $w_i \cdot (1 - A(c_i \setminus \{r_i\}))$

- **Global Upper Bound**
  - BCD only maintains local upper bounds $\mu_i$
  - $\sum_i \mu_i$ can **increase** in some iterations
    ▶ conservative update of $\mu_i$ over unrelaxed soft clauses
    ▶ the weight of the clause
  - BCD2 maintains $\mu$ as **global upper bound** (and its assignment $A_\mu$)
    ▶ $\mu_i$ updated on unrelaxed soft clauses with the cost of the clause on $A_\mu$
• New Lower Bound when merging disjoint cores with unrelaxed clauses
  – Merge $C_1, \ldots, C_k$
  – Add $(c_1, w_1), \ldots, (c_m, w_m)$
• BCD: $\lambda_{\text{new}} = \lambda_1 + \ldots + \lambda_k$
BCD2 - Improvements on the lower bound 1/2

- New Lower Bound when merging disjoint cores with unrelaxed clauses
  - Merge $C_1, \ldots, C_k$
  - Add $(c_1, w_1), \ldots, (c_m, w_m)$

- BCD: $\lambda_{new} = \lambda_1 + \ldots + \lambda_k$

- BCD2: $\lambda_{new} = \lambda_1 + \ldots + \lambda_k + \Delta$
  - $\Delta = \ldots$ (next slide)
  - Why the current unsatisfiable core? Because:
    - Unrelaxed Soft Clauses
    - Disjoint Core needs update of bounds
    - A mix of both
• Why the current unsatisfiable core?
  Because:
  – Unrelaxed Soft Clauses
    ▶ One of the unrelaxed clauses unsatisfied: \((c_x, w_x)\)
    ▶ Increment by \(w_x\)
Why the current unsatisfiable core? Because:

- Unrelaxed Soft Clauses
  - One of the unrelaxed clauses unsatisfied: \((c_x, w_x)\)
  - Increment by \(w_x\)
  - Safe minimum: \(\delta_1 = \min\{w_1, \ldots, w_m\}\)
Why the current unsatisfiable core?
Because:

- Unrelaxed Soft Clauses
  - One of the unrelaxed clauses unsatisfied: \((c_x, w_x)\)
  - Increment by \(w_x\)
  - Safe minimum: \(\delta_1 = \min\{w_1, \ldots, w_m\}\)

- Disjoint Core needs update of bounds
  - Update lower bound of disjoint core: \(\lambda_x = \tau_x\)
  - Increment by \(\tau_x - \lambda_x\)
Why the current unsatisfiable core? Because:

- Unrelaxed Soft Clauses
  - One of the unrelaxed clauses unsatisfied: \((c_x, w_x)\)
  - Increment by \(w_x\)
  - Safe minimum: \(\delta_1 = \min\{w_1, \ldots, w_m\}\)

- Disjoint Core needs update of bounds
  - Update lower bound of disjoint core: \(\lambda_x = \tau_x\)
  - Increment by \(\tau_x - \lambda_x\)
  - Safe minimum: \(\delta_2 = \min\{\tau_1 - \lambda_1, \ldots, \tau_k - \lambda_k\}\)
Why the current unsatisfiable core?

Because:

- Unrelaxed Soft Clauses
  - One of the unrelaxed clauses unsatisfied: \((c_x, w_x)\)
  - Increment by \(w_x\)
  - Safe minimum: \(\delta_1 = \min\{w_1, \ldots, w_m\}\)

- Disjoint Core needs update of bounds
  - Update lower bound of disjoint core: \(\lambda_x = \tau_x\)
  - Increment by \(\tau_x - \lambda_x\)
  - Safe minimum: \(\delta_2 = \min\{\tau_1 - \lambda_1, \ldots, \tau_k - \lambda_k\}\)

- A mix of both
  - Safe minimum: \(\delta_1 + \delta_2\)
Why the current unsatisfiable core?
Because:
- Unrelaxed Soft Clauses
  - One of the unrelaxed clauses unsatisfied: \((c_x, w_x)\)
  - Increment by \(w_x\)
  - Safe minimum: \(\delta_1 = \min\{w_1, \ldots, w_m\}\)
- Disjoint Core needs update of bounds
  - Update lower bound of disjoint core: \(\lambda_x = \tau_x\)
  - Increment by \(\tau_x - \lambda_x\)
  - Safe minimum: \(\delta_2 = \min\{\tau_1 - \lambda_1, \ldots, \tau_k - \lambda_k\}\)
- A mix of both
  - Safe minimum: \(\delta_1 + \delta_2\)

In BCD2 \(\Delta = \min\{\delta_1, \delta_2, \delta_1 + \delta_2\} = \min\{\delta_1, \delta_2\}\)
Outline

Motivation

Previous Algorithms - Overview

BCD2

Optimizations

Results

Conclusions
Additional Techniques

• Hardening
  – Adapted the Hardening Rule from B&B MaxSAT algorithms
  – Use global $\mu$ and $\sum \lambda_i$ as bounds
  – Whenever bounds are updated then test hardening rule on remaining soft clauses
  – May declare soft clauses as hard
  – Details in the paper
Additional Techniques

• **Hardening**
  - Adapted the Hardening Rule from B&B MaxSAT algorithms
  - Use global $\mu$ and $\sum \lambda_i$ as bounds
  - Whenever bounds are updated then test hardening rule on remaining soft clauses
  - May declare soft clauses as hard
  - Details in the paper

• **Biased Search**
  - Decide $\tau$ depending on the outcomes of previous iterations
  - Not necessarily the middle point
  - Details in the paper
Outline

Motivation

Previous Algorithms - Overview

BCD2

Optimizations

Results

Conclusions
Experimental Results

- Linux cluster with 50 nodes, each node 3Ghz and 32 GB
- Time limit 1800 seconds and memory limit 4GB

- Instances from 2009-2011 MaxSAT Evaluation instances
  - All Non-Random instances - crafted and industrial
  - All categories: (Plain) MaxSAT, Partial MaxSAT, Weighted MaxSAT, Weighted Partial MaxSAT
  - > 2600 instances
BC, BCD and BCD2 (and extensions)

BC 1730; BCD 1801; BCD2 1813; BCD2-B-H 1832
BCD vs BCD2
1305 instances with $\geq 20\%$ runtime difference:

- 918 BCD2 wins
- 387 BCD wins
1305 instances with $\geq$ 20% runtime difference:
- 918 BCD2 wins
- 387 BCD wins

1793 instances solved by both:
- BCD: 124,907 SAT calls
- BCD2: 68,690 SAT calls
1305 instances with $\geq 20\%$ runtime difference:
- 918 BCD2 wins
- 387 BCD wins

1793 instances solved by both:
- BCD: 124 907 SAT calls
- BCD2: 68 690 SAT calls
- close 50% less calls
Motivation

Previous Algorithms - Overview

BCD2

Optimizations

Results

Conclusions
Contributions

• Proposed improvements to BCD
  – Improvement on the organization of the BCD algorithm:
    ▶ Maintain a global upper bound $\mu$ and associated assignment $A_\mu$
    ▶ Maintain the contribution of each soft clause to the cost
    ▶ New more aggressive lower bound
    ▶ Reduced the number of iterations (fewer SAT calls)
  – Proposed two techniques - Hardening & Biased Search
    ▶ Can be implemented on any core-guided MaxSAT algorithm (that uses upper and lower bounds)

• BCD2 (with extensions) shown to improved BCD on extensive set of benchmarks
Obrigado. Thank you. Grazie.