Progression in Maximum Satisfiability

A. Ignatiev¹, A. Morgado¹, V. Manquinho¹, I. Lynce¹ and J. Marques-Silva¹,²

¹IST/INESC-ID, Lisbon, Portugal
²UCD CASL, Dublin, Ireland

ECAI 2014
Prague, Czech Rep, Aug 2014
Recap MaxSAT

- Given **unsatisfiable** formula, find **largest** subset of clauses that is satisfiable
Recap MaxSAT

- Given unsatisfiable formula, find largest subset of clauses that is satisfiable
- A Minimal Correction Subset (MCS) is an irreducible relaxation of the formula
Recap MaxSAT

- Given unsatisfiable formula, find largest subset of clauses that is satisfiable
- A Minimal Correction Subset (MCS) is an irreducible relaxation of the formula
- The MaxSAT solution is one of the smallest MCSes
MaxSAT problem(s)

<table>
<thead>
<tr>
<th>Weights?</th>
<th>Hard Clauses?</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### MaxSAT problem(s)

<table>
<thead>
<tr>
<th>Weights?</th>
<th>Hard Clauses?</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MaxSAT problem(s)

<table>
<thead>
<tr>
<th>Hard Clauses?</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weights?</td>
<td>Plain</td>
<td>Partial</td>
</tr>
<tr>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Weighted</td>
<td>Weighted Partial</td>
</tr>
</tbody>
</table>

- **Must** satisfy *hard* clauses, if any
- Compute set of satisfied *soft* clauses with **maximum cost**
  - Without weights, cost of each falsified soft clause is 1
- **Or**, compute set of falsified *soft* clauses with **minimum cost** (s.t. *hard* & remaining *soft* clauses are satisfied)
Many MaxSAT approaches

- Branch & Bound
- Model Guided
- Iterative MHS
- Iterative
- Core Guided

For practical (industrial) instances: core-guided approaches the most effective [MaxSAT14]
Many MaxSAT approaches

- Branch & Bound
  - No unit prop; No cl. learning
- Model Guided
- Iterative MHS
- Core Guided
- Iterative
  - For practical (industrial) instances: core-guided approaches the most effective [MaxSAT14]
Many MaxSAT approaches

MaxSAT Algorithms

- Branch & Bound
- Model Guided
- Iterative MHS
- Core Guided
- Iterative

All cls relaxed

For practical (industrial) instances: core-guided approaches the most effective [MaxSAT14]
Many MaxSAT approaches

MaxSAT Algorithms

- Branch & Bound
- Core Guided
- Model Guided
- Iterative MHS
- Iterative

- Relax cls given unsat cores

For practical (industrial) instances: core-guided approaches the most effective [MaxSAT14]
Many MaxSAT approaches

MaxSAT Algorithms

- Branch & Bound
- Model Guided
- Iterative MHS
- Core Guided

- Iterative MHS & SAT

For practical (industrial) instances: core-guided approaches the most effective [MaxSAT14]
Many MaxSAT approaches

Relax cls given models

MaxSAT Algorithms

- Branch & Bound
- Model Guided
- Interactive
- Iterative MHS
- Core Guided

For practical (industrial) instances: core-guided approaches the most effective [MaxSAT14]
Many MaxSAT approaches

- For practical (industrial) instances: core-guided approaches the most effective

[MaxSAT14]
Query complexity of MaxSAT

- $W$: sum of weights of soft clauses
- $C$: the minimum cost of falsified clauses, i.e. the MaxSAT solution

- Worst case number of oracle calls: $O(\log W)$ \[P94\]
- In practical settings, $W$ often much larger than $C$
  - E.g. design debugging: Often, need to relax a few gates given universe of several million gates
Query complexity of MaxSAT

- $W$: sum of weights of soft clauses
- $C$: the minimum cost of falsified clauses, i.e. the MaxSAT solution

- Worst case number of oracle calls: $O(\log W)$ [P94]
- In practical settings, $W$ often much larger than $C$
  - E.g. design debugging: Often, need to relax a few gates given universe of several million gates

Our work:
1. Develop algorithm with $\log C$ worst-case number of oracle calls
2. Integrate new approach with core-guided paradigm
Outline

Progressions in MaxSAT

Core-Guided Progression Algorithms

Experimental Results
Outline

Progressions in MaxSAT

Core-Guided Progression Algorithms

Experimental Results
Motivation:

- Avoid unnecessary binary search iterations when $W \gg C$
MaxSAT using geometric progressions

Progression_Iterative(\(\mathcal{F}\))

Input: \(\mathcal{F} = \mathcal{F}_S \cup \mathcal{F}_H\)

\((R, \mathcal{F}_W) \leftarrow \text{Relax}(\emptyset, \mathcal{F}, \mathcal{F}_S)\) \quad \text{// Relax & harden soft clause } c_i \text{ with } r_i
\((\lambda, j) \leftarrow (0, 0)\) \quad \text{// LB & progression index}

while true do
    \(\tau \leftarrow 2^j - 1\) \quad \text{// Tentative UB w/ geom. prog.}
    if \(\tau > \sum_{r_i \in R} w_i\) then
        return BinSearch(\(\mathcal{F}_W, R, \lambda, \emptyset\)) \quad \text{// Bin search if UB } \geq W
    \((\text{st}, \mathcal{A}) \leftarrow \text{SAT}(\mathcal{F}_W \cup \text{CNF}(\sum_{r_i \in R} w_i r_i \leq \tau))\)
    if st = true then
        return BinSearch(\(\mathcal{F}_W, R, \lambda, \mathcal{A}\)) \quad \text{// Bin search given (actual) UB}
    else
        \(\lambda \leftarrow \tau\) \quad \text{// Update LB}
        \(j \leftarrow j + 1\) \quad \text{// Increase progression index}
MaxSAT using geometric progressions

Progression_Iterative($\mathcal{F}$)

Input: $\mathcal{F} = \mathcal{F}_S \cup \mathcal{F}_H$

$(R, \mathcal{F}_W) \leftarrow \text{Relax}(\emptyset, \mathcal{F}, \mathcal{F}_S)$ \hspace{1cm} // Relax & harden soft clause $c_i$ with $r_i$

$(\lambda, j) \leftarrow (0, 0)$ \hspace{1cm} // LB & progression index

while true do

$\tau \leftarrow 2^j - 1$ \hspace{1cm} // Tentative UB w/ geom. prog.

if $\tau > \sum_{r_i \in R} w_i$ then

return BinSearch($\mathcal{F}_W, R, \lambda, \emptyset$) \hspace{1cm} // Bin search if UB $\geq W$

$(\text{st}, \mathcal{A}) \leftarrow \text{SAT}(\mathcal{F}_W \cup \text{CNF}(\sum_{r_i \in R} w_i r_i \leq \tau))$

if st = true then

return BinSearch($\mathcal{F}_W, R, \lambda, \mathcal{A}$) \hspace{1cm} // Bin search given (actual) UB

else

$\lambda \leftarrow \tau$ \hspace{1cm} // Update LB

$j \leftarrow j + 1$ \hspace{1cm} // Increase progression index

• Worst-case number of oracle calls: $\mathcal{O}(\log C)$
Earlier work using geometric progressions

- Used for improving lower bounds
  - Optimization problems in planning
  - Job shop scheduling

- Used in algorithms for computing a minimal set subject to a monotone predicate (MSMP)
  - E.g. MUSes, MCSes, minimal models, etc.
  - Also used in ILOG
Outline

Progressions in MaxSAT

Core-Guided Progression Algorithms

Experimental Results
Progression & core-guided algorithms

- Use geometric progression (instead of binary) search

- Refine computed upper bound with core-guided algorithm:
  - Core-guided binary search (Bin Core, BC) [HMMS11]
  - Bin Core with disjoint cores (BCD/BCD2) [HMMS11, MHMS12]
  - Actually, any core-guided algorithm that refines \((LB, UB]\) can be used

- Worst case number of oracle calls in \(O(m + \log C)\)
  - \(O(m + \log C)\): geometric progression step
  - \(O(m + \log C)\): BC/BCD/BCD2 step
  - Where, \(O(m)\) captures the iterative relaxation of soft clauses
  - Compare with \(O(m + \log W)\) for BC, BCD/BCD2 [HMMS11]
Progression with core-guided binary search

\[
\text{Progression\_BinCore}(\mathcal{F})
\]

**Input:** \( \mathcal{F} = \mathcal{F}_S \cup \mathcal{F}_H \)

\((R, \mathcal{F}_W) \leftarrow (\emptyset, \mathcal{F}) \quad \text{// Initially no clauses relaxed}

\((\lambda, j) \leftarrow (0, 0) \quad \text{// LB & progression index} \)

**while** true

\( \tau \leftarrow 2^j - 1 \quad \text{// Tentative UB w/ geom. prog.} \)

\[ \begin{aligned}
&\text{if } \tau > \sum_{r_i \in R} w_i \text{ then} \\
&\quad \text{return } \text{BinCore}(\mathcal{F}_W, R, \lambda, \emptyset) \quad \text{// Bin core if UB} \geq W
\end{aligned} \]

\((\text{st}, \mathcal{U}, \mathcal{A}) \leftarrow \text{SAT}(\mathcal{F}_W \cup \text{CNF}(\sum_{r_i \in R} w_i r_i \leq \tau)) \)

\[ \begin{aligned}
&\text{if } \text{st} = \text{true} \text{ then} \\
&\quad \text{return } \text{BinCore}(\mathcal{F}_W, R, \lambda, \mathcal{A}) \quad \text{// Bin core given (actual) UB}
\end{aligned} \]

\[ \begin{aligned}
&\text{else} \\
&\quad \text{if } \mathcal{U} \cap \mathcal{F}_S = \emptyset \text{ then} \\
&\quad \quad \lambda \leftarrow \tau \quad \text{// Update LB}
\end{aligned} \]

\[ \begin{aligned}
&\quad \text{else} \\
&\quad \quad j \leftarrow j + 1 \quad \text{// Increase progression index}
\end{aligned} \]

\[ \begin{aligned}
&\quad \text{else} \\
&\quad \quad (R, \mathcal{F}_W) \leftarrow \text{Relax}(R, \mathcal{F}_W, \mathcal{U} \cap \mathcal{F}_S) \quad \text{// Relax & harden soft clauses}
\end{aligned} \]
Outline

Progressions in MaxSAT

Core-Guided Progression Algorithms

Experimental Results
Experimental setup

- All industrial instances from 2013 MaxSAT evaluation:
  - Plain: 55; Partial: 627; Weighted partial: 396
  - Total: 1078 instances

- HPC cluster
  - Each node: two processors E5-2620 @2GHz
  - Each processor: 6 cores and 128 GByte of RAM
  - Each process limited to 4 GByte of RAM and timeout of 1800s

- MaxSAT solvers: best from 2013 MaxSAT evaluation
  - QMxS: QMaxSAT 0.21 (only partial MaxSAT) [KZFH12]
  - MFMx: MiFuMax [J13]
  - MSUC: MSUnCore with BCD2 algorithm [HMMS11,MHMS12]
  - WPM1 [ABL13]
  - WPM2 [ABL13,ABGL13]
  - New algorithms: PRG_BC & PRG_BCD
Cactus plots on all instances

Better performance on MaxSAT14 benchmarks
– But also additional better performing algorithms in 2014
Better performance on MaxSAT14 benchmarks
- But also additional better performing algorithms in 2014
Better performance on MaxSAT14 benchmarks
  - But also additional better performing algorithms in 2014
Scatter plot – PRG_BCD vs. QMaxSAT (partial MaxSAT)
Scatter plot – PRG_BCD vs. WPM2
Scatter plot – PRG_BCD vs. MSUC
Conclusions & research directions

- Applied geometric progressions to iterative MaxSAT solving
  - PRG with binary search
- Extended approach to core-guided algorithms
  - PRG_BC & PRG_BCD
- New algorithms outperform state of the art MaxSAT solvers

- Remarkable improvements in MaxSAT solvers in recent years
  - Core-guide best performing, but many core-guided approaches
  - Why are core-guided algorithms so effective in practice?
  - Also, several recent solvers to compare against [NB14,MJML14,MDMS14]
- VBS results motivate portfolio of solvers
Thank You