Impact of SAT-Based Preprocessing on Core-Guided MaxSAT Solving[1]

Jeremias Berg and Matti Järvisalo

HIIT, Dept. Computer Science
University of Helsinki
Finland

Originally presented at
CP 2016
Toulouse
September 7, 2016

[1] Work funded by Academy of Finland, grants 251170 COIN, 276412, and 284591; and Doctoral School in Computer Science DoCS and Research Funds of the University of Helsinki.

Our Contributions

- Core guided MaxSAT solvers use a SAT-solver as a subroutine.
  - Understanding what factors affect number of calls important for developing efficient solvers.
- *Preprocessing:* essential in SAT-solving.
  - Motivates development of preprocessing for MaxSAT.
- We analyze the effect of preprocessing on the number of SAT calls required by core guided MaxSAT solvers.

Our Contributions

- Core guided MaxSAT solvers use a SAT-solver as a subroutine.
  - Understanding what factors affect number of calls important for developing efficient solvers.
- *Preprocessing:* essential in SAT-solving.
  - Motivates development of preprocessing for MaxSAT.
- We analyze the effect of preprocessing on the number of SAT calls required by core guided MaxSAT solvers.

Our Contributions

- Core guided MaxSAT solvers use a SAT-solver as a subroutine.
  - Understanding what factors affect number of calls important for developing efficient solvers.
- *Preprocessing:* essential in SAT-solving.
  - Motivates development of preprocessing for MaxSAT.
- We analyze the effect of preprocessing on the number of SAT calls required by core guided MaxSAT solvers.

Our Contributions

- Core guided MaxSAT solvers use a SAT-solver as a subroutine.
  - Understanding what factors affect number of calls important for developing efficient solvers.
- *Preprocessing:* essential in SAT-solving.
  - Motivates development of preprocessing for MaxSAT.
- We analyze the effect of preprocessing on the number of SAT calls required by core guided MaxSAT solvers.

Our Results:

- Preprocessing has *no effect* on the best case number of iterations.
- Preprocessing can improve the worst case number of iterations.

Outline

- Background / Motivation
- Satisfiability (SAT) and Maximum Satisfiability (MaxSAT)
- Preprocessing SAT
- Preprocessing MaxSAT
- Research question
- Abstractions of MaxSAT solvers
- Results
- Proof intuition
- Summary

Background / Motivation

- Maximum Satisfiability
  - The optimization counterpart of the Satisfiability problem
- Exact MaxSAT solving is an active area of research.
- Solvers have improved significantly over the last years.
- Applications in: inconsistency analysis, diagnosis, design debugging, and fault localization, AI, combinatorics, data analysis, bioinformatics, ...

Park [2002]
Chen et al. [2009]
Chen et al. [2010]
Argelich et al. [2010]
Lynce and Marques-Silva [2011]
Zhu et al. [2011]
Jose and Majumdar [2011]
Zhang and Bacchus [2012]
Ansótegui et al. [2013b]
Ignatiev et al. [2014]
Berg et al. [2014]
Fang et al. [2014]
Berg and Järvisalo [2014]
Marques-Silva et al. [2015]
Berg and Järvisalo [2015]
Wallner et al. [2016]

Background / Motivation

- Maximum Satisfiability
  - The optimization counterpart of the Satisfiability problem
- Exact MaxSAT solving is an active area of research.
- Solvers have improved significantly over the last years.
- Applications in: inconsistency analysis, diagnosis, design debugging, and fault localization, AI, combinatorics, data analysis, bioinformatics, ...

Park [2002]
Chen et al. [2009]
Chen et al. [2010]
Argelich et al. [2010]
Lynce and Marques-Silva [2011]
Zhu et al. [2011]
Jose and Majumdar [2011]
Zhang and Bacchus [2012]
Ansótegui et al. [2013b]
Ignatiev et al. [2014]
Berg et al. [2014]
Fang et al. [2014]
Berg and Järvisalo [2014]
Marques-Silva et al. [2015]
Berg and Järvisalo [2015]
Wallner et al. [2016]

MaxSAT for real world problems

- Most MaxSAT solvers for industrial problems use of a SAT solver as a subroutine.
  - Unsat Core extraction
- Potential speedups: faster or fewer SAT solver calls.
- *Preprocessing:* an essential part of SAT-solving
- Currently not as well understood for MaxSAT.

Ansótegui et al. [2013a]
Morgado et al. [2013]
Ansótegui et al. [2010]
Bacchus and Narodytska [2014]
Belov et al. [2013]
Koshimura et al. [2012]
Davies and Bacchus [2013]
Bjørner and Narodytska [2015]
Morgado et al. [2014]
Martins et al. [2014b]
Eén and Biere [2005]
Järvisalo et al. [2012]

MaxSAT for real world problems

- Most MaxSAT solvers for industrial problems use of a SAT solver as a subroutine.
  - Unsat Core extraction
- Potential speedups: faster or fewer SAT solver calls.
- *Preprocessing:* an essential part of SAT-solving
- Currently not as well understood for MaxSAT.

Ansótegui et al. [2013a]
Morgado et al. [2013]
Ansótegui et al. [2010]
Bacchus and Narodytska [2014]
Belov et al. [2013]
Koshimura et al. [2012]
Davies and Bacchus [2013]
Bjørner and Narodytska [2015]
Morgado et al. [2014]
Martins et al. [2014b]
Eén and Biere [2005]
Järvisalo et al. [2012]

MaxSAT for real world problems

- Most MaxSAT solvers for industrial problems use of a SAT solver as a subroutine.
  - Unsat Core extraction
- Potential speedups: faster or fewer SAT solver calls.
- *Preprocessing:* an essential part of SAT-solving
- Currently not as well understood for MaxSAT.

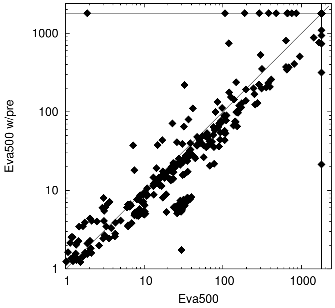
Ansótegui et al. [2013a]
Morgado et al. [2013]
Ansótegui et al. [2010]
Bacchus and Narodytska [2014]
Belov et al. [2013]
Koshimura et al. [2012]
Davies and Bacchus [2013]
Bjørner and Narodytska [2015]
Morgado et al. [2014]
Martins et al. [2014b]
Eén and Biere [2005]
Järvisalo et al. [2012]

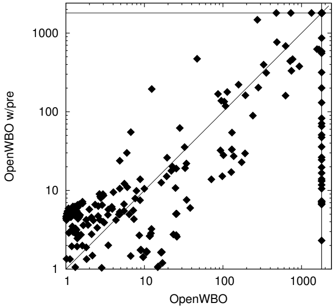
MaxSAT for real world problems

- Most MaxSAT solvers for industrial problems use of a SAT solver as a subroutine.
  - Unsat Core extraction
- Potential speedups: faster or fewer SAT solver calls.
- *Preprocessing:* an essential part of SAT-solving
- Currently not as well understood for MaxSAT.

Ansótegui et al. [2013a]
Morgado et al. [2013]
Ansótegui et al. [2010]
Bacchus and Narodytska [2014]
Belov et al. [2013]
Koshimura et al. [2012]
Davies and Bacchus [2013]
Bjørner and Narodytska [2015]
Morgado et al. [2014]
Martins et al. [2014b]
Eén and Biere [2005]
Järvisalo et al. [2012]

Effect of Preprocessing





- Solvers: Eva500 and Open-Wbo

  [Narodytska and Bacchus, 2014; Martins, Joshi, Manquinho, and Lynce, 2014a]

- Preprocessing + Solving time on Y-axis.

Our Contributions

- Formal Analysis of the effect of preprocessing on MaxSAT solving.

Results

- Neither preprocessing (nor core-minimization) has *any effect* on the best case performance.
- Preprocessing (and core minimization) can improve the worst case performance of both solvers,

Our Contributions

- Formal Analysis of the effect of preprocessing on MaxSAT solving.
  - The effect of preprocessing on the number of SAT-solver calls required by two abstractions of MaxSAT solvers
  - As a byproduct, similar analysis on core minimization.

Results

- Neither preprocessing (nor core-minimization) has *any effect* on the best case performance.
- Preprocessing (and core minimization) can improve the worst case performance of both solvers,

Our Contributions

- Formal Analysis of the effect of preprocessing on MaxSAT solving.
  - The effect of preprocessing on the number of SAT-solver calls required by two abstractions of MaxSAT solvers
  - As a byproduct, similar analysis on core minimization.

Results

- Neither preprocessing (nor core-minimization) has *any effect* on the best case performance.
- Preprocessing (and core minimization) can improve the worst case performance of both solvers,

- Satisfiability (SAT): Given a CNF-formula F, decide if its satisfiable

$$F = \quad (y \vee \neg z) \wedge (x \vee z) \wedge (\neg z) \wedge (y \vee z) \wedge (\neg x) \wedge (\neg y \vee \neg z) \wedge (z \vee y)$$

- Satisfiability (SAT): Given a CNF-formula F, decide if its satisfiable

$$F = (y \vee \neg z) \wedge (x \vee z) \wedge (\neg z) \wedge (y \vee z) \wedge (\neg x) \wedge (\neg y \vee \neg z) \wedge (z \vee y)$$

Satisfiability

- Satisfiability (SAT): Given a CNF-formula F, decide if its satisfiable

$$F = \{\{y, \neg z\}, \{x, z\}, \{\neg z\}, \{y, z\}, \{\neg x\}, \{\neg y, \neg z\}, \{z, y\}\}$$

Maximum Satisfiability

- (Partial) Maximum Satisfiability (MaxSAT):
  - Two CNF-Formulas: F_h, (hard clauses) and F_s (soft clauses)

$$F_h = \{(x \vee z), (\neg z), (y \vee z)\}$$

$$F_s = \{(\neg x), (\neg y \vee \neg z), (z \vee y), (\neg z \vee y)\}$$

Maximum Satisfiability

- (Partial) Maximum Satisfiability (MaxSAT):
  - Two CNF-Formulas: F_h, (hard clauses) and F_s (soft clauses)
  - Find a truth assignment satisfying F_h and a maximum number of clauses in F_s.

$$F_h = \{(x \vee z), (\neg z), (y \vee z)\}$$

$$F_s = \{(\neg x), (\neg y \vee \neg z), (z \vee y), (\neg z \vee y)\}$$

Maximum Satisfiability

- (Weighted Partial) Maximum Satisfiability (MaxSAT):
  - Two CNF-Formulas: F_h, (hard clauses) and F_s (soft clauses)
  - Find a truth assignment satisfying F_h and a maximum sum of weights of clauses in F_s.

$$F_h = \{(x \lor z), (\neg z), (y \lor z)\}$$

$$F_s = \{((\neg x), 1), ((\neg y \lor \neg z), 3), ((z \lor y), 7), ((\neg z \lor y), 3)\}$$

Maximum Satisfiability

- (Weighted Partial) Maximum Satisfiability (MaxSAT):
  - Two CNF-Formulas: F_h, (hard clauses) and F_s (soft clauses)
  - Find a truth assignment satisfying F_h and a maximum sum of weights of clauses in F_s.
- A (minimal) unsatisfiable core (MUS) C: a (subset minimal) $C \subseteq F_s$ s.t. $C \wedge F_h$ is unsatisfiable.
  - $mus(F)$: the set of all MUSes of F.

$$F_h = \{(\mathbf{x} \vee \mathbf{z}), (\neg \mathbf{z}), (y \vee z)\}$$

$$F_s = \{((\neg \mathbf{x}), 1), ((\neg y \vee \neg z), 3), ((z \vee y), 7), ((\neg z \vee y), 3)\}$$

Preprocessing SAT

Preprocessing pipeline:

1. Apply (fast) satisfiability preserving simplifications to F to obtain $pre(F)$.
2. Solve $pre(F)$.
3. Reconstruct solution to F (if needed).

- Subsumption Elimination (SE)
  - If there exists clauses $C, D \in F$ s.t. $C \subseteq D$, remove D.
- Bounded Variable Elimination (BVE) [Eén and Biere, 2005]
- Self Subsuming Resolution (SSR)
- Blocked Clause Elimination (BCE) [Järvisalo, Heule, and Biere, 2012]

Preprocessing SAT

Preprocessing pipeline:

1. Apply (fast) satisfiability preserving simplifications to F to obtain $pre(F)$.
2. Solve $pre(F)$.
3. Reconstruct solution to F (if needed).

Goal: Preprocessing + solving + reconstructing faster than solving F.

- Subsumption Elimination (SE)
  - If there exists clauses $C, D \in F$ s.t. $C \subseteq D$, remove D.
- Bounded Variable Elimination (BVE) [Eén and Biere, 2005]
- Self Subsuming Resolution (SSR)
- Blocked Clause Elimination (BCE) [Järvisalo, Heule, and Biere, 2012]

Preprocessing SAT

Preprocessing pipeline:

1. Apply (fast) satisfiability preserving simplifications to F to obtain $pre(F)$.
2. Solve $pre(F)$.
3. Reconstruct solution to F (if needed).

Goal: Preprocessing + solving + reconstructing faster than solving F.

Techniques

- Subsumption Elimination (SE)
  - If there exists clauses $C, D \in F$ s.t. $C \subseteq D$, remove D.
- Bounded Variable Elimination (BVE) [Eén and Biere, 2005]
- Self Subsuming Resolution (SSR)
- Blocked Clause Elimination (BCE) [Järvisalo, Heule, and Biere, 2012]

Preprocessing SAT

Preprocessing pipeline:

1. Apply (fast) satisfiability preserving simplifications to F to obtain $pre(F)$.
2. Solve $pre(F)$.
3. Reconstruct solution to F (if needed).

Goal: Preprocessing + solving + reconstructing faster than solving F.

Techniques

- Subsumption Elimination (SE)
  - If there exists clauses $C, D \in F$ s.t. $C \subseteq D$, remove D.
- Bounded Variable Elimination (BVE) [Eén and Biere, 2005]
- Self Subsuming Resolution (SSR)
- Blocked Clause Elimination (BCE) [Järvisalo, Heule, and Biere, 2012]

Preprocessing SAT

Preprocessing pipeline:

1. Apply (fast) satisfiability preserving simplifications to F to obtain $pre(F)$.
2. Solve $pre(F)$.
3. Reconstruct solution to F (if needed).

Goal: Preprocessing + solving + reconstructing faster than solving F.

SAT-based preprocessing

- Subsumption Elimination **(SE)**
  - If there exists clauses $C, D \in F$ s.t. $C \subseteq D$, remove D.
- Bounded Variable Elimination **(BVE)** [Eén and Biere, 2005]
- Self Subsuming Resolution **(SSR)**
- Blocked Clause Elimination **(BCE)** [Järvisalo, Heule, and Biere, 2012]

For the rest of this talk: **SAT-based preprocessing techniques**

Preprocessing MaxSAT

- SAT-based preprocessing techniques have been lifted to MaxSAT.
  [Belov, Morgado, and Marques-Silva, 2013]
  - Requires fresh "label[2]" variables on soft clauses.

[2] Assumption, Reification

Preprocessing MaxSAT

- SAT-based preprocessing techniques have been lifted to MaxSAT.

  [Belov, Morgado, and Marques-Silva, 2013]

  - Requires fresh "label[2]" variables on soft clauses.

[2] Assumption, Reification

Preprocessing MaxSAT

- SAT-based preprocessing techniques have been lifted to MaxSAT.
  [Belov, Morgado, and Marques-Silva, 2013]
  - Requires fresh "label[2]" variables on soft clauses.

Preprocessing a MaxSAT instance (F_h, F_s)

1. Run SAT-preprocessor on $F_h \cup F_s^a$ where $F_s^a = \{C \lor l_C \mid C \in F_s\}$
   - Do not resolve on l_C variables.
   - Output $pre(F)_h$
2. $pre(F_s) = \{(\neg l_C) \mid C \in F_s\}$
3. Solve $(pre(F)_h, pre(F)_s)$

[2] Assumption, Reification

Effect of Preprocessing

For a fixed MaxSAT algorithm $\mathcal{A}$, consider:

1. $\mathcal{A}$
2. $\mathcal{A}_{\text{pre}}$: $\mathcal{A}$ + SAT based preprocessing.
3. $\mathcal{A}^{mus}$: $\mathcal{A}$ using a SAT solver that is guaranteed to return a MUS when invoked on an unsatisfiable formula
4. $\mathcal{A}^{mus}_{\text{pre}}$: $\mathcal{A}^{mus}$ + SAT based preprocessing.

Effect of Preprocessing

For a fixed MaxSAT algorithm $\mathcal{A}$, consider:

1. $\mathcal{A}$
2. $\mathcal{A}_{\text{pre}}$: $\mathcal{A}$ + SAT based preprocessing.
3. $\mathcal{A}^{mus}$: $\mathcal{A}$ using a SAT solver that is guaranteed to return a MUS when invoked on an unsatisfiable formula
4. $\mathcal{A}^{mus}_{\text{pre}}$: $\mathcal{A}^{mus}$ + SAT based preprocessing.

Our Research Question

How does SAT based preprocessing affect the number of SAT-solver calls required by these variants for $\mathcal{A} \in \{\text{CG}, \text{HS}\}$?

The abstract MaxSAT solvers we analyze in this work.

Bacchus and Narodytska [2014]

CG:
$F_w^1 \leftarrow F_h \cup F_s$
for *i=1...* **do**
  $(\textit{result}, \kappa, \tau) \leftarrow \text{SATSOLVE}(F_w^i)$
  if *result="satisfiable"* **then**
    return τ // optimal solution
  else
    // κ = unsat core
    $F_w^i = (F_w^i \setminus \kappa)$
    $F_w^{i+1} \leftarrow \text{RELAX}(F_w^i, \kappa)$
  end
end

- Core guided solver.
- Iteratively extracts cores from the instances and relaxes them.
- Refines a lower bound on the optimal cost
- Instantiated as: Fu-Malik, WPM1, MSU3, ...

The abstract MaxSAT solvers we analyze in this work.

Davies and Bacchus [2013]; Saikko et al. [2016]

- Implicit hitting set approach to MaxSAT.
- Iteratively extracts cores from the instances and computes hitting sets over the set of found cores.
- Instantiated as: MaxHS, LMHS

HS:

$\mathcal{K} \leftarrow \emptyset$ // set of found unsat cores of F
$F_w \leftarrow (F_h \cup F_s)$
while *true* **do**
  $H \leftarrow$ MINCOSTHITTINGSET($\mathcal{K}$)
  $F_w \leftarrow F_h \cup (F_s \setminus H)$
  (*result*, κ, τ) $\leftarrow$ SATSOLVE(F_w)
  if *result="satisfiable"* **then**
    return τ // optimal solution
  else
    // $\kappa =$ unsat core
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\kappa\}$
  end
end

The abstract MaxSAT solvers we analyze in this work.

CG:

$F_w^1 \leftarrow F_h \cup F_s$
for *i=1...* **do**
  $(result, \kappa, \tau) \leftarrow \text{SATSOLVE}(F_w^i)$
  if *result="satisfiable"* **then**
    return τ // optimal solution
  else
    // $\kappa =$ unsat core
    $F_w^i = (F_w^i \setminus \kappa)$
    $F_w^{i+1} \leftarrow \text{RELAX}(F_w^i, \kappa)$
  end
end

HS:

$\mathcal{K} \leftarrow \emptyset$ // set of found unsat cores of F
$F_w \leftarrow (F_h \cup F_s)$
while *true* **do**
  $H \leftarrow \text{MINCOSTHITTINGSET}(\mathcal{K})$
  $F_w \leftarrow F_h \cup (F_s \setminus H)$
  $(result, \kappa, \tau) \leftarrow \text{SATSOLVE}(F_w)$
  if *result="satisfiable"* **then**
    return τ // optimal solution
  else
    // $\kappa =$ unsat core
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\kappa\}$
  end
end

Example CG: WPM1

Manquinho et al. [2009]; Ansótegui et al. [2009]; Fu and Malik [2006]

$$F_h = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (x_5 \vee x_6 \vee x_7)\}$$

$$F_s = \{(\neg x_1), (\neg x_2), (\neg x_3), (\neg x_4), (\neg x_5), (\neg x_6), (\neg x_7)\}$$

Input

Example CG: WPM1

Manquinho et al. [2009]; Ansótegui et al. [2009]; Fu and Malik [2006]

$$F_h = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (x_5 \vee x_6 \vee x_7)\}$$

$$F_s = \{(\neg x_1), (\neg x_2), (\neg \mathbf{x_3}), (\neg \mathbf{x_4}), (\neg \mathbf{x_5}), (\neg x_6), (\neg x_7)\}$$

First Core $\{(\neg x_3), (\neg x_4), (\neg x_5)\}$

Example CG: WPM1

Manquinho et al. [2009]; Ansótegui et al. [2009]; Fu and Malik [2006]

$$F_h = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (x_5 \vee x_6 \vee x_7), \text{CNF}(r_1 + r_2 + r_3 = 1)\}$$

$$F_s = \{(\neg x_1), (\neg x_2), (\neg x_3 \vee r_1), (\neg x_4 \vee r_2), (\neg x_5 \vee r_3), (\neg x_6), (\neg x_7)\}$$

First Core $\{(\neg x_3), (\neg x_4), (\neg x_5)\}$

Example CG: WPM1

Manquinho et al. [2009]; Ansótegui et al. [2009]; Fu and Malik [2006]

$$F_h = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (x_5 \vee x_6 \vee x_7), \text{CNF}(r_1 + r_2 + r_3 = 1)\}$$

$$F_s = \{(\neg \mathbf{x_1}), (\neg \mathbf{x_2}), (\neg \mathbf{x_3} \vee \mathbf{r_1}), (\neg x_4 \vee r_2), (\neg \mathbf{x_5} \vee \mathbf{r_3}), (\neg \mathbf{x_6}), (\neg \mathbf{x_7})\}$$

Second Core $\{(\neg x_1), (\neg x_2), (\neg x_3 \vee r_1), (\neg x_5 \vee r_3), (\neg x_6), (\neg x_7)\}$

Example CG: WPM1

Manquinho et al. [2009]; Ansótegui et al. [2009]; Fu and Malik [2006]

$$F_h = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (x_5 \vee x_6 \vee x_7), \text{CNF}(r_1 + r_2 + r_3 = 1), \text{CNF}(r_4 + r_5 + r_6 + r_7 + r_8 + r_9 = 1)\}$$

$$F_s = \{(\neg x_1 \vee r_4), (\neg x_2 \vee r_5), (\neg x_3 \vee r_1 \vee r_6), (\neg x_4 \vee r_2), (\neg x_5 \vee r_3 \vee r_7), (\neg x_6 \vee r_8), (\neg x_7 \vee r_9)\}$$

Second Core $\{(\neg x_1), (\neg x_2), (\neg x_3 \vee r_1), (\neg x_5 \vee r_3), (\neg x_6), (\neg x_7)\}$

Example CG: WPM1

Manquinho et al. [2009]; Ansótegui et al. [2009]; Fu and Malik [2006]

$$F_h = \{(x_1 \lor x_2 \lor x_3), (x_3 \lor x_4 \lor x_5), (x_5 \lor x_6 \lor x_7), \\ \text{CNF}(r_1 + r_2 + r_3 = 1), \\ \text{CNF}(r_4 + r_5 + r_6 + r_7 + r_8 + r_9 = 1)\}$$

$$F_s = \{(\neg x_1 \lor r_4), (\neg x_2 \lor r_5), (\neg x_3 \lor r_1 \lor r_6), (\neg x_4 \lor r_2), \\ (\neg x_5 \lor r_3 \lor r_7), (\neg x_6 \lor r_8), (\neg x_7 \lor r_9)\}$$

SATISFIABLE
2 original soft clauses unsatisfied

$$F_h = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (x_5 \vee x_6 \vee x_7)\}$$

$$F_s = \{(\neg x_1), (\neg x_2), (\neg x_3), (\neg x_4), (\neg x_5), (\neg x_6), (\neg x_7)\}$$

Input

Example HS: Davies and Bacchus [2013]; Saikko et al. [2016]

$$F_h = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (x_5 \vee x_6 \vee x_7)\}$$

$$F_s = \{(\neg x_1), (\neg x_2), (\neg \mathbf{x_3}), (\neg \mathbf{x_4}), (\neg \mathbf{x_5}), (\neg x_6), (\neg x_7)\}$$

First Core $\{(\neg x_3), (\neg x_4), (\neg x_5)\}$

Set of Cores:

$\{(\neg x_3), (\neg x_4), (\neg x_5)\}$

$$F_h = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (x_5 \vee x_6 \vee x_7)\}$$

$$F_s = \{(\neg x_1), (\neg x_2), (\neg \mathbf{x_3}), (\neg \mathbf{x_4}), (\neg \mathbf{x_5}), (\neg x_6), (\neg x_7)\}$$

First Core $\{(\neg x_3), (\neg x_4), (\neg x_5)\}$

First Hitting Set $\{(\neg x_3)\}$

Set of Cores:

$\{(\neg x_3), (\neg x_4), (\neg x_5)\}$

$$F_h = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (x_5 \vee x_6 \vee x_7)\}$$

$$F_s = \{(\neg x_1), (\neg x_2), \cancel{(\neg x_3)}, (\neg x_4), (\neg x_5), (\neg x_6), (\neg x_7)\}$$

First Core $\{(\neg x_3), (\neg x_4), (\neg x_5)\}$

First Hitting Set $\{(\neg x_3)\}$

Set of Cores:

$\{(\neg x_3), (\neg x_4), (\neg x_5)\}$

$$F_h = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (x_5 \vee x_6 \vee x_7)\}$$

$$F_s = \{(\neg x_1), (\neg x_2), \cancel{(\neg x_3)}, (\neg x_4), (\neg \mathbf{x_5}), (\neg \mathbf{x_6}), (\neg \mathbf{x_7})\}$$

Second Core $\{(\neg x_5), (\neg x_6), (\neg x_7)\}$

Set of Cores:

$\{(\neg x_3), (\neg x_4), (\neg x_5)\}$

$\{(\neg x_5), (\neg x_6), (\neg x_7)\}$

Example HS: Davies and Bacchus [2013]; Saikko et al. [2016]

$$F_h = \{(x_1 \lor x_2 \lor x_3), (x_3 \lor x_4 \lor x_5), (x_5 \lor x_6 \lor x_7)\}$$

$$F_s = \{(\neg x_1), (\neg x_2), \cancel{(\neg x_3)}, (\neg x_4), (\neg \mathbf{x_5}), (\neg \mathbf{x_6}), (\neg \mathbf{x_7})\}$$

Second Core $\{(\neg x_5), (\neg x_6), (\neg x_7)\}$

Second Hitting Set $\{(\neg x_5)\}$

Set of Cores:

$\{(\neg x_3), (\neg x_4), (\neg x_5)\}$

$\{(\neg x_5), (\neg x_6), (\neg x_7)\}$

Example HS: Davies and Bacchus [2013]; Saikko et al. [2016]

$$F_h = \{(x_1 \lor x_2 \lor x_3), (x_3 \lor x_4 \lor x_5), (x_5 \lor x_6 \lor x_7)\}$$

$$F_s = \{(\neg x_1), (\neg x_2), (\neg x_3), (\neg x_4), \cancel{(\neg x_5)}, (\neg x_6), (\neg x_7)\}$$

Second Core $\{(\neg x_5), (\neg x_6), (\neg x_7)\}$

Second Hitting Set $\{(\neg x_5)\}$

Set of Cores:

$\{(\neg x_3), (\neg x_4), (\neg x_5)\}$

$\{(\neg x_5), (\neg x_6), (\neg x_7)\}$

Example HS: Davies and Bacchus [2013]; Saikko et al. [2016]

$$F_h = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (x_5 \vee x_6 \vee x_7)\}$$

$$F_s = \{(\neg \mathbf{x_1}), (\neg \mathbf{x_2}), (\neg \mathbf{x_3}), (\neg x_4), \cancel{(\neg x_5)}, (\neg x_6), (\neg x_7)\}$$

Third Core $\{(\neg x_1), (\neg x_2), (\neg x_3)\}$

Set of Cores:

$\{(\neg x_3), (\neg x_4), (\neg x_5)\}$

$\{(\neg x_5), (\neg x_6), (\neg x_7)\}$

$\{(\neg x_1), (\neg x_2), (\neg x_3)\}$

Example HS: Davies and Bacchus [2013]; Saikko et al. [2016]

$$F_h = \{(x_1 \lor x_2 \lor x_3), (x_3 \lor x_4 \lor x_5), (x_5 \lor x_6 \lor x_7)\}$$

$$F_s = \{(\neg \mathbf{x_1}), (\neg \mathbf{x_2}), (\neg \mathbf{x_3}), (\neg x_4), \cancel{(\neg x_5)}, (\neg x_6), (\neg x_7)\}$$

Third Core $\{(\neg x_1), (\neg x_2), (\neg x_3)\}$

Third Hitting Set $\{(\neg x_3), (\neg x_5)\}$

Set of Cores:

$\{(\neg x_3), (\neg x_4), (\neg x_5)\}$

$\{(\neg x_5), (\neg x_6), (\neg x_7)\}$

$\{(\neg x_1), (\neg x_2), (\neg x_3)\}$

Example HS: Davies and Bacchus [2013]; Saikko et al. [2016]

$$F_h = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (x_5 \vee x_6 \vee x_7)\}$$

$$F_s = \{(\neg x_1), (\neg x_2), \cancel{(\neg x_3)}, (\neg x_4), \cancel{(\neg x_5)}, (\neg x_6), (\neg x_7)\}$$

Third Core $\{(\neg x_1), (\neg x_2), (\neg x_3)\}$

Third Hitting Set $\{(\neg x_3), (\neg x_5)\}$

Set of Cores:

$\{(\neg x_3), (\neg x_4), (\neg x_5)\}$

$\{(\neg x_5), (\neg x_6), (\neg x_7)\}$

$\{(\neg x_1), (\neg x_2), (\neg x_3)\}$

Example HS: Davies and Bacchus [2013]; Saikko et al. [2016]

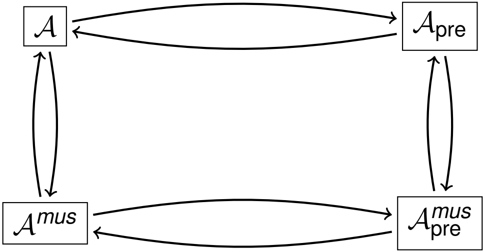
$$F_h = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (x_5 \vee x_6 \vee x_7)\}$$

$$F_s = \{(\neg x_1), (\neg x_2), \cancel{(\neg x_3)}, (\neg x_4), \cancel{(\neg x_5)}, (\neg x_6), (\neg x_7)\}$$

SATISFIABLE
2 original soft clauses unsatisfied

Our Results, Best-Case Performance

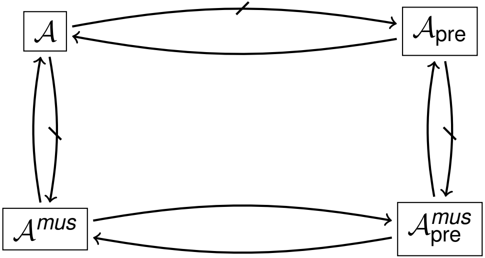
For $\mathcal{A} \in \{\text{CG}, \text{HS}\}$:



$X \rightarrow Y$ iff MINITERATIONS$(X) \leq$ MINITERATIONS(Y)

Our Results, Worst-Case Performance

For $\mathcal{A} \in \{\text{CG}, \text{HS}\}$:



$X \rightarrow Y$ iff $\text{MaxIterations}(X) \leq \text{MaxIterations}(Y)$

$X \nrightarrow Y$ indicates that $X \rightarrow Y$ does not hold.

Proof intuition

Belov et al. [2013]: Preprocess a MaxSAT instance F using SAT-based preprocessing to obtain $pre(F)$. Then

$$mus(F) = mus(pre(F)). \tag{1}$$

We show:

- The shortest executions of all solver variants require the SAT solver only extracting MUSes. Best Case results
- There are instances on which extracting non minimal cores forces both algorithms to iterate unnecessary many times. Worst case results

Proof intuition

Belov et al. [2013]: Preprocess a MaxSAT instance F using SAT-based preprocessing to obtain $pre(F)$. Then

$$mus(F) = mus(pre(F)). \quad (1)$$

We show:

- The shortest executions of all solver variants require the SAT solver only extracting MUSes. Best Case results
- There are instances on which extracting non minimal cores forces both algorithms to iterate unnecessary many times. Worst case results

Proof intuition

Belov et al. [2013]: Preprocess a MaxSAT instance F using SAT-based preprocessing to obtain $pre(F)$. Then

$$mus(F) = mus(pre(F)). \quad (1)$$

We show:

- The shortest executions of all solver variants require the SAT solver only extracting MUSes. Best Case results
- There are instances on which extracting non minimal cores forces both algorithms to iterate unnecessary many times. Worst case results

Proof intuition

Belov et al. [2013]: Preprocess a MaxSAT instance F using SAT-based preprocessing to obtain $pre(F)$. Then

$$mus(F) = mus(pre(F)). \tag{1}$$

We show:

- The shortest executions of all solver variants require the SAT solver only extracting MUSes. Best Case results
- There are instances on which extracting non minimal cores forces both algorithms to iterate unnecessary many times. Worst case results

Note: Results hold for any preprocessing techniques satisfying Eq. 1.

Summary

- Understanding what factors affect the number of SAT solver calls is important for developing more efficient MaxSAT solvers.
- In this work: Effect of SAT-based preprocessing on the number of iterations
  - No effect on the best case.
  - Can improve the worst case.
- Further work:
  - Similar analysis for other MaxSAT algorithms (MaxRES, OLL, . . .).
  - Effect of preprocessing on individual SAT-solver calls.
  - Development of other MaxSAT preprocessing techniques.

Summary

- Understanding what factors affect the number of SAT solver calls is important for developing more efficient MaxSAT solvers.
- In this work: Effect of SAT-based preprocessing on the number of iterations
  - No effect on the best case.
  - Can improve the worst case.
- Further work:
  - Similar analysis for other MaxSAT algorithms (MaxRES, OLL, . . .).
  - Effect of preprocessing on individual SAT-solver calls.
  - Development of other MaxSAT preprocessing techniques.

Summary

- Understanding what factors affect the number of SAT solver calls is important for developing more efficient MaxSAT solvers.
- In this work: Effect of SAT-based preprocessing on the number of iterations
  - No effect on the best case.
  - Can improve the worst case.
- Further work:
  - Similar analysis for other MaxSAT algorithms (MaxRES, OLL, . . .).
  - Effect of preprocessing on individual SAT-solver calls.
  - Development of other MaxSAT preprocessing techniques.

Summary

- Understanding what factors affect the number of SAT solver calls is important for developing more efficient MaxSAT solvers.
- In this work: Effect of SAT-based preprocessing on the number of iterations
  - No effect on the best case.
  - Can improve the worst case.
- Further work:
  - Similar analysis for other MaxSAT algorithms (MaxRES, OLL, ...).
  - Effect of preprocessing on individual SAT-solver calls.
  - Development of other MaxSAT preprocessing techniques.

C. Ansótegui, M.L. Bonet, and J. Levy. SAT-based MaxSAT algorithms. *Artificial Intelligence*, 196:77–105, 2013a.

Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. Solving (weighted) partial maxsat through satisfiability testing. In *Proc. SAT*, volume 5584 of *Lecture Notes in Computer Science*, pages 427–440. Springer, 2009.

Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. A new algorithm for weighted partial maxsat. In *Proc AAAI*. AAAI Press, 2010.

Carlos Ansótegui, Idelfonso Izquierdo, Felip Manyà, and José Torres-Jiménez. A Max-SAT-based approach to constructing optimal covering arrays. In *Proc. CCIA*, volume 256 of *Frontiers in Artificial Intelligence and Applications*, pages 51–59. IOS Press, 2013b.

Josep Argelich, Daniel Le Berre, Inês Lynce, João P. Marques-Silva, and Pascal Rapicault. Solving linux upgradeability problems using boolean optimization. In *Proc. LoCoCo*, volume 29 of *EPTCS*, pages 11–22, 2010.

Fahiem Bacchus and Nina Narodytska. Cores in core based MaxSat algorithms: An analysis. In *Proc. SAT*, volume 8561 of *Lecture Notes in Computer Science*, pages 7–15. Springer, 2014.

A. Belov, A. Morgado, and J. Marques-Silva. SAT-based preprocessing for MaxSAT. In *Proc. LPAR-19*, volume 8312 of *Lecture Notes in Computer Science*, pages 96–111. Springer, 2013.

J. Berg, M. Järvisalo, and B. Malone. Learning optimal bounded treewidth Bayesian networks via maximum satisfiability. In *Proc. AISTATS*, volume 33 of *JMLR Workshop and Conference Proceedings*, pages 86–95. JMLR.org, 2014.

Jeremias Berg and Matti Järvisalo. SAT-based approaches to treewidth computation: An evaluation. In *Proc. ICTAI*, pages 328–335. IEEE Computer Society, 2014.

Jeremias Berg and Matti Järvisalo. Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability. *Artificial Intelligence*, 2015. in press.

Nikolaj Bjørner and Nina Narodytska. Maximum satisfiability using cores and correction sets. In *Proc. IJCAI*, pages 246–252. AAAI Press, 2015.

Yibin Chen, Sean Safarpour, Andreas G. Veneris, and João P. Marques-Silva. Spatial and temporal design debug using partial MaxSAT. In *Proc. 19th ACM Great Lakes Symposium on VLSI*, pages 345–350. ACM, 2009.

Yibin Chen, Sean Safarpour, João Marques-Silva, and Andreas G. Veneris. Automated design debugging with maximum satisfiability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(11):1804–1817, 2010.

Bibliography II

J. Davies and F. Bacchus. Exploiting the power of MIP solvers in MaxSAT. In *Proc. SAT*, volume 7962 of *Lecture Notes in Computer Science*, pages 166–181. Springer, 2013.

N. Eén and A. Biere. Effective preprocessing in SAT through variable and clause elimination. In *Proc. SAT*, volume 3569 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2005.

Zhiwen Fang, Chu-Min Li, Kan Qiao, Xu Feng, and Ke Xu. Solving maximum weight clique using maximum satisfiability reasoning. In *Proc. ECAI*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 303–308. IOS Press, 2014.

Zhaohui Fu and Sharad Malik. On solving the partial MaxSAT problem. In *Proc. SAT*, volume 4121 of *Lecture Notes in Computer Science*, pages 252–265. Springer, 2006.

Alexey Ignatiev, Mikolás Janota, and João Marques-Silva. Towards efficient optimization in package management systems. In *Proc. ICSE*, pages 745–755. ACM, 2014.

Matti Järvisalo, Marijn Heule, and Armin Biere. Inprocessing rules. In *Proc. IJCAR*, volume 7364 of *Lecture Notes in Computer Science*, pages 355–370. Springer, 2012.

M. Jose and R. Majumdar. Cause clue clauses: error localization using maximum satisfiability. In *Proc. PLDI*, pages 437–446. ACM, 2011.

M. Koshimura, T. Zhang, H. Fujita, and R. Hasegawa. QMaxSAT: A partial Max-SAT solver. *Journal of Satisfiability, Boolean Modeling and Computation*, 8(1/2):95–100, 2012.

Inês Lynce and João Marques-Silva. Restoring CSP satisfiability with MaxSAT. *Fundam. Inform.*, 107(2-3):249–266, 2011.

Vasco M. Manquinho, João P. Marques-Silva, and Jordi Planes. Algorithms for weighted boolean optimization. In *Proc. SAT*, volume 5584 of *Lecture Notes in Computer Science*, pages 495–508. Springer, 2009.

J. Marques-Silva, M. Janota, A. Ignatiev, and A. Morgado. Efficient model based diagnosis with maximum satisfiability. In *Proc. IJCAI*, pages 1966–1972. AAAI Press, 2015.

R. Martins, S. Joshi, V.M. Manquinho, and I. Lynce. Incremental cardinality constraints for MaxSAT. In *Proc. CP*, volume 8656 of *Lecture Notes in Computer Science*, pages 531–548. Springer, 2014a.

Ruben Martins, Vasco M. Manquinho, and Inês Lynce. Open-WBO: A modular MaxSAT solver. In *Proc. SAT*, volume 8561 of *Lecture Notes in Computer Science*, pages 438–445. Springer, 2014b.

Bibliography III

A. Morgado, F. Heras, M.H. Liffiton, J. Planes, and J. Marques-Silva. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints*, 18(4):478–534, 2013.

A. Morgado, C. Dodaro, and J. Marques-Silva. Core-guided maxsat with soft cardinality constraints. In *Proc. CP*, volume 8656 of *Lecture Notes in Computer Science*, pages 564–573. Springer, 2014.

N. Narodytska and F. Bacchus. Maximum satisfiability using core-guided MaxSAT resolution. In *Proc. AAAI*, pages 2717–2723. AAAI Press, 2014.

James D. Park. Using weighted MAX-SAT engines to solve MPE. In *Proc. AAAI*, pages 682–687. AAAI Press / The MIT Press, 2002.

Paul Saikko, Jeremias Berg, and Matti Järvisalo. LMHS: A SAT-IP hybrid MaxSAT solver. In Nadia Creignou and Daniel Le Berre, editors, *Proc. SAT*, volume 9710 of *Lecture Notes in Computer Science*, pages 539–546. Springer, 2016.

Johannes Peter Wallner, Andreas Niskanen, and Matti Järvisalo. Complexity results and algorithms for extension enforcement in abstract argumentation. In *Proc. AAAI*. AAAI Press, 2016.

Lei Zhang and Fahiem Bacchus. MAXSAT heuristics for cost optimal planning. In *Proc. AAAI*. AAAI Press, 2012.

C.S. Zhu, G. Weissenbacher, and S. Malik. Post-silicon fault localisation using maximum satisfiability and backbones. In *Proc. FMCAD*, pages 63–66. FMCAD Inc., 2011.