SAT Preprocessing for MUS Extraction and MaxSAT

Anton Belov

Joint work with: Matti Järvisalo, António Morgado, Joao Marques-Silva

University College Dublin, Ireland
University of Helsinki, Finland
IST/INESC-ID, Lisbon, Portugal

EPCL Basic Training Camp
November 13-19, 2013
Dresden, Germany
Abstract

State-of-the-art algorithms for industrial instances of MUS extraction problem and MaxSAT rely on iterative calls to a SAT solver. Preprocessing is crucial for the acceleration of SAT solving, and the key preprocessing techniques rely on the application of resolution and subsumption elimination. Additionally, satisfiability-preserving clause elimination procedures are often used. Since the computation typically involves a large number of SAT calls, an interesting question is whether an input instance to a problem can be preprocessed up-front, i.e. prior to running an MUS extractor or a MaxSAT solver, rather than (or, in addition to) during each iterative SAT solver call. The key requirement in this setting is that the preprocessing has to be sound, i.e. so that the solution to the original problem can be reconstructed correctly and efficiently after the execution of an algorithm on the preprocessed instance. In this talk we will examine some of the obstacles to such up-front preprocessing, and will discuss a solution that involves re-casting the MUS and MaxSAT computation problems in a so-called labelled-CNF framework.
Outline

Introduction: MUSes, MaxSAT, SAT preprocessing

Direct reconstruction: techniques that work

Direct reconstruction: techniques that break

Sound preprocessing in Labelled CNF framework
Outline

Introduction: MUSes, MaxSAT, SAT preprocessing

Direct reconstruction: techniques that work

Direct reconstruction: techniques that break

Sound preprocessing in Labelled CNF framework
Introduction: MUSes, group-MUSes

\[ C_1 \quad C_2 \quad C_3 \]
\[
\begin{array}{c}
(p) \\
(q) \\
(\neg p \lor \neg q)
\end{array}
\]

\[ M = \{ C_1, C_2, C_3 \} \text{ is UNSAT} \]
Introduction: MUSes, group-MUSes

\[
\begin{array}{ccc}
C_1 & C_2 & C_3 \\
(p) & (q) & (\neg p \lor \neg q) \\
\end{array}
\]

\[M = \{C_1, C_2, C_3\}\] is UNSAT, and \(\forall C \in M, M \setminus \{C\}\) is SAT.
Introduction: MUSes, group-MUSes

\[\begin{align*}
C_1 & \quad (p) \\
C_2 & \quad (q) \\
C_3 & \quad (\neg p \lor \neg q)
\end{align*}\]

\[M = \{C_1, C_2, C_3\}\text{ is minimal unsatisfiable (MU).}\]
**Introduction: MUSes, group-MUSes**

\[
\begin{array}{cccccc}
C_1 & C_2 & C_3 & C_4 & C_5 & C_6 \\
(p) & (q) & (\neg p \lor \neg q) & (\neg p \lor r) & (p \lor q) & (\neg q \lor \neg r)
\end{array}
\]

\(M = \{C_1, C_2, C_3\}\) is *minimal unsatisfiable (MU)*.

\(F = \{C_1, \ldots, C_6\}\) is UNSAT, but not MU.
Introduction: MUSes, group-MUSes

<table>
<thead>
<tr>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_3$</th>
<th>$C_4$</th>
<th>$C_5$</th>
<th>$C_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(p)$</td>
<td>$(q)$</td>
<td>$(\neg p \lor \neg q)$</td>
<td>$(\neg p \lor r)$</td>
<td>$(p \lor q)$</td>
<td>$(\neg q \lor \neg r)$</td>
</tr>
</tbody>
</table>

$M = \{C_1, C_2, C_3\}$ is **minimal unsatisfiable (MU)**.

$F = \{C_1, \ldots, C_6\}$ is UNSAT, but not MU.

$M$ is a **minimal unsatisfiable subformula (MUS)** of $F$. 

Introduction: MUSes, group-MUSes

\[\begin{align*}
C_1 & : (p) \\
C_2 & : (q) \\
C_3 & : (\neg p \lor \neg q) \\
C_4 & : (\neg p \lor r) \\
C_5 & : (p \lor q) \\
C_6 & : (\neg q \lor \neg r)
\end{align*}\]

\(M = \{C_1, C_2, C_3\}\) is \textit{minimal unsatisfiable (MU)}. \\
\(F = \{C_1, \ldots, C_6\}\) is UNSAT, but not MU. \\
\(M\) is a \textit{minimal unsatisfiable subformula (MUS)} of \(F\). And there is another one.
Introduction: MUSes, group-MUSes

\[
\begin{align*}
C_1 & : (p) \\
C_2 & : (q) \\
C_3 & : (\neg p \lor \neg q) \\
C_4 & : (\neg p \lor r) \\
C_5 & : (p \lor q) \\
C_6 & : (\neg q \lor \neg r)
\end{align*}
\]

\(M = \{C_1, C_2, C_3\}\) is minimal unsatisfiable (MU).

\(F = \{C_1, \ldots, C_6\}\) is UNSAT, but not MU.

\(M\) is a minimal unsatisfiable subformula (MUS) of \(F\). And there is another one.

\[
\begin{align*}
G_0 & : (p) (q) \\
G_1 & : (\neg p \lor \neg q) (\neg p \lor r) \\
G_2 & : (p \lor q) (\neg q \lor \neg r)
\end{align*}
\]

A group-CNF formula — CNF partitioned into groups of clauses.
Introduction: MUSes, group-MUSes

\[
\begin{array}{cccccc}
C_1 & C_2 & C_3 & C_4 & C_5 & C_6 \\
(p) & (q) & (\neg p \lor \neg q) & (\neg p \lor r) & (p \lor q) & (\neg q \lor \neg r)
\end{array}
\]

\[M = \{C_1, C_2, C_3\}\] is \textit{minimal unsatisfiable (MU)}.\[F = \{C_1, \ldots, C_6\}\] is UNSAT, but not MU.\[M\] is a \textit{minimal unsatisfiable subformula (MUS)} of \(F\). And there is another one.

\[
\begin{array}{ccc}
G_0 & | & G_1 & | & G_2 \\
(p) & (q) & (\neg p \lor \neg q) & (\neg p \lor r) & (p \lor q) & (\neg q \lor \neg r)
\end{array}
\]

A \textit{group-CNF} formula — CNF partitioned into \textit{groups} of clauses.\[F = \{G_0, G_1, G_2\}\] is UNSAT, but not “group-MU” (can remove \(G_2\)).
Introduction: MUSes, group-MUSes

\[ C_1, C_2, C_3, C_4, C_5, C_6 \]

\[
\begin{align*}
(p) & \quad (q) & \quad (\neg p \lor \neg q) & \quad (\neg p \lor r) & \quad (p \lor q) & \quad (\neg q \lor \neg r)
\end{align*}
\]

\( M = \{C_1, C_2, C_3\} \) is minimal unsatisfiable (MU).

\( F = \{C_1, \ldots, C_6\} \) is UNSAT, but not MU.

\( M \) is a minimal unsatisfiable subformula (MUS) of \( F \). And there is another one.

\[
\begin{align*}
\quad G_0 & \quad | & \quad G_1 & \quad | & \quad G_2 \\
(p) & \quad (q) & \quad (\neg p \lor \neg q) & \quad (\neg p \lor r) & \quad (p \lor q) & \quad (\neg q \lor \neg r)
\end{align*}
\]

A group-CNF formula — CNF partitioned into groups of clauses.

\( F = \{G_0, G_1, G_2\} \) is UNSAT, but not “group-MU” (can remove \( G_2 \)).

\( M = \{G_1\} \) is (the only) group-MUS of \( F \).
Introduction: MUSes, group-MUSes

\[ C_1 \quad C_2 \quad C_3 \quad C_4 \quad C_5 \quad C_6 \]

\[
(p) \quad (q) \quad (\neg p \lor \neg q) \quad (\neg p \lor r) \quad (p \lor q) \quad (\neg q \lor \neg r)
\]

\[ M = \{ C_1, C_2, C_3 \} \] is minimal unsatisfiable (MU).

\[ F = \{ C_1, \ldots, C_6 \} \] is UNSAT, but not MU.

\[ M \] is a minimal unsatisfiable subformula (MUS) of \( F \). And there is another one.

\[ G_0 \quad G_1 \quad G_2 \]

\[
(p) \quad (q) \quad (\neg p \lor \neg q) \quad (\neg p \lor r) \quad (p \lor q) \quad (\neg q \lor \neg r)
\]

A group-CNF formula — CNF partitioned into groups of clauses.

\[ F = \{ G_0, G_1, G_2 \} \] is UNSAT, but not “group-MU” (can remove \( G_2 \)).

\[ M = \{ G_1 \} \] is (the only) group-MUS of \( F \).
A MaxSAT solution for $F$, is an assignment $\tau$ that maximizes $|\text{Sat}(F, \tau)|$. Alternatively (and more customary): $\tau$ minimizes $|\text{Unsat}(F, \tau)|$.

\[
\begin{array}{cccccc}
C_1 & C_2 & C_3 & C_4 & C_5 & C_6 \\
(p) & (q) & (\neg p \lor \neg q) & (\neg p \lor r) & (p \lor q) & (\neg q \lor \neg r)
\end{array}
\]
Introduction: MaxSAT

A MaxSAT solution for $F$, is an assignment $\tau$ that maximizes $|\text{Sat}(F, \tau)|$. Alternatively (and more customary): $\tau$ minimizes $|\text{Unsat}(F, \tau)|$.

\[
\begin{align*}
C_1 & \quad C_2 & \quad C_3 & \quad C_4 & \quad C_5 & \quad C_6 \\
\text{(p)} & \quad \text{(q)} & \quad \text{(-p v -q)} & \quad \text{(-p v r)} & \quad \text{(p v q)} & \quad \text{(-q v -r)}
\end{align*}
\]

$\tau_1 = \{\neg p, q, \neg r\}$
Introduction: MaxSAT

A **MaxSAT solution** for $F$, is an assignment $\tau$ that maximizes $|Sat(F, \tau)|$. Alternatively (and more customary): $\tau$ minimizes $|Unsat(F, \tau)|$.

$$\begin{align*}
C_1 & \quad C_2 & \quad C_3 & \quad C_4 & \quad C_5 & \quad C_6 \\
(p) & \quad (q) & \quad (\neg p \lor \neg q) & \quad (\neg p \lor r) & \quad (p \lor q) & \quad (\neg q \lor \neg r)
\end{align*}$$

$\tau_1 = \{-p, q, \neg r\}$, $\tau_2 = \{p, \neg q, r\}$
Introduction: MaxSAT

A MaxSAT solution for $F$, is an assignment $\tau$ that maximizes $|\text{Sat}(F, \tau)|$. Alternatively (and more customary): $\tau$ minimizes $|\text{Unsat}(F, \tau)|$.

$$C_1 \quad C_2 \quad C_3 \quad C_4 \quad C_5 \quad C_6$$

$$\text{(p)} \quad \text{(q)} \quad (\neg p \lor \neg q) \quad (\neg p \lor r) \quad (p \lor q) \quad (\neg q \lor \neg r)$$

$\tau_1 = \{\neg p, q, \neg r\}$, $\tau_2 = \{p, \neg q, r\}$

Note: a MaxSAT solution is a model of an MSS of maximum cardinality $\Rightarrow$ MaxSAT is about finding an MCS of a minimum cardinality.
Introduction: MaxSAT

A **MaxSAT solution** for $F$, is an assignment $\tau$ that maximizes $|Sat(F, \tau)|$. Alternatively (and more customary): $\tau$ minimizes $|Unsat(F, \tau)|$.

\[
\begin{align*}
C_1 & : (p) \\
C_2 & : (q) \\
C_3 & : (\neg p \lor \neg q) \\
C_4 & : (\neg p \lor r) \\
C_5 & : (p \lor q) \\
C_6 & : (\neg q \lor \neg r)
\end{align*}
\]

$\tau_1 = \{\neg p, q, \neg r\}$, $\tau_2 = \{p, \neg q, r\}$

**Note:** a MaxSAT solution is a model of an MSS of *maximum* cardinality $\Rightarrow$ MaxSAT is about finding an MCS of a *minimum* cardinality.

**Weighted Partial MaxSAT:**

- each clause $C$ has a non-negative weight ($\in \mathbb{N}^+$), $w(C)$;
- some clauses are **hard** — *must* be satisfied (akin to group-0);
- weighted CNF (WCNF) formula: $F = F^H \cup F^S$;
- $\tau$ is a weighted partial MaxSAT solution for WCNF $F$ if $\tau(F^H) = 1$, and $\tau$ minimizes $cost(\tau) = \sum_{C \in Unsat(F^S, \tau)} w(C)$. 
Introduction: applications

Applications

- Identification and repair of sources of inconsistency
  - circuit error diagnosis; error localization in product configuration.
- Identification of relevant features of systems:
  - automatic abstraction in model checking;
  - environmental assumptions in formal equivalence checking.
- MaxSAT: debugging, optimization, bioinformatics, etc.
Introduction: applications

Applications

▶ Identification and repair of sources of inconsistency
  - circuit error diagnosis; error localization in product configuration.
▶ Identification of relevant features of systems:
  - automatic abstraction in model checking;
  - environmental assumptions in formal equivalence checking.
▶ MaxSAT: debugging, optimization, bioinformatics, etc.

Computation of (group-)MUSes and MaxSAT solutions

▶ Algorithms aimed at industrial instances are based on iterative calls to a SAT solver.
▶ SAT solving is the main bottleneck.
▶ Number of SAT calls is a function of the size of the input formula.
▶ In the SAT world: preprocessing is essential for efficient SAT solving.
Clause elimination procedures

\[ E : \text{CNF} \mapsto \text{CNF}, \ E(F) \subseteq F, \text{ and } E(F) \text{ is equisatisfiable with } F. \]
Clause elimination procedures

\[ E : CNF \mapsto CNF, \ E(F) \subseteq F, \text{ and } E(F) \text{ is equisatisfiable with } F. \]

- **Subsumption elimination**
  
  \( C \) subsumes \( C' \) if \( C \subseteq C' \).

Important:

a model of original formula can be reconstructed efficiently.
Introduction: preprocessing for SAT

Clause elimination procedures

\[ E : \text{CNF} \mapsto \text{CNF}, \quad E(F) \subseteq F, \quad \text{and} \quad E(F) \text{ is equisatisfiable with } F. \]

- **Subsumption elimination**
  
  \( C \) subsumes \( C' \) if \( C \subset C' \).

- **Blocked clause elimination (BCE)**
  
  \( C \) is blocked if every resolvent of \( C \) on some \( l \in C \) is a tautology.

Important: a model of original formula can be reconstructed efficiently.
Introduction: preprocessing for SAT

Clause elimination procedures

\[ E : CNF \mapsto CNF, \; E(F) \subseteq F, \; \text{and} \; E(F) \text{ is equisatisfiable with } F. \]

- **Subsumption elimination**
  
  \( C \) subsumes \( C' \) if \( C \subset C' \).

- **Blocked clause elimination (BCE)**
  
  \( C \) is blocked if every resolvent of \( C \) on some \( l \in C \) is a tautology.

Resolution-based preprocessing

- **Boolean constraint propagation (BCP)**
Introduction: preprocessing for SAT

Clause elimination procedures

\( E : \text{CNF} \mapsto \text{CNF}, \ E(F) \subseteq F, \) and \( E(F) \) is equisatisfiable with \( F. \)

- **Subsumption elimination**
  
  \( C \) subsumes \( C' \) if \( C \subset C'. \)

- **Blocked clause elimination (BCE)**

  \( C \) is blocked if every resolvent of \( C \) on some \( l \in C \) is a tautology.

Resolution-based preprocessing

- **Boolean constraint propagation (BCP)**

- **Variable elimination (VE) (aka DP-reduction)**

  \[
  \text{VE}(F, x) = F \cup (F_x \otimes_x F_{\neg x}) \setminus (F_x \cup F_{\neg x})
  \]
Introduction: preprocessing for SAT

Clause elimination procedures

\[ E : \text{CNF} \mapsto \text{CNF}, \ E(F) \subseteq F, \ \text{and} \ E(F) \ \text{is equisatisfiable with} \ F. \]

- **Subsumption elimination**

  \[ C \ \text{subsumes} \ C' \ \text{if} \ C \subset C'. \]

- **Blocked clause elimination (BCE)**

  \[ C \ \text{is blocked if every resolvent of} \ C \ \text{on some} \ l \in C \ \text{is a tautology.} \]

Resolution-based preprocessing

- **Boolean constraint propagation (BCP)**

- **Variable elimination (VE) (aka DP-reduction)**

  \[ \text{VE}(F, x) = F \cup (F_x \otimes F_{\neg x}) \setminus (F_x \cup F_{\neg x}) \]

- **Self-subsuming resolution (SSR)**

  replace \((x \lor C) \ (\neg x \lor C \lor D)\) with \((x \lor C) \ (\ C \lor D)\).
Introduction: preprocessing for SAT

Clause elimination procedures

\( E : \text{CNF} \mapsto \text{CNF}, \ E(F) \subseteq F, \text{ and } E(F) \text{ is equisatisfiable with } F. \)

- **Subsumption elimination**
  
  \( C \) subsumes \( C' \) if \( C \subset C' \).

- **Blocked clause elimination (BCE)**
  
  \( C \) is blocked if every resolvent of \( C \) on some \( l \in C \) is a tautology.

Resolution-based preprocessing

- **Boolean constraint propagation (BCP)**

- **Variable elimination (VE) (aka DP-reduction)**
  
  \[ \text{VE}(F, x) = F \cup (F_x \otimes_x F_{\neg x}) \setminus (F_x \cup F_{\neg x}) \]

- **Self-subsuming resolution (SSR)**
  
  replace \( (x \lor C) (\neg x \lor C \lor D) \) with \( (x \lor C) (C \lor D) \).

**Important:** a model of original formula can be reconstructed efficiently.
Introduction: preprocessing for MUSes and MaxSAT

Input formula $F$

preprocessor

Preprocessed formula $F'$

MUS extractor

MaxSAT solver

MUS of $F'$

MaxSAT solution of $F'$

$P$-time procedure
Introduction: preprocessing for MUSes and MaxSAT

Input formula $F$

Preprocessor

Preprocessed formula $F'$

MUS extractor

MUS of $F'$

MaxSAT solver

MaxSAT solution of $F'$
Introduction: preprocessing for MUSes and MaxSAT

Input formula $F$

preprocessor

Preprocessed formula $F'$

MUS extractor

MUS of $F'$

$P$-time procedure

MUS of $F$

MaxSAT solver

MaxSAT solution of $F'$

$P$-time procedure

MaxSAT solution of $F$
Outline

Introduction: MUSes, MaxSAT, SAT preprocessing

Direct reconstruction: techniques that work

Direct reconstruction: techniques that break

Sound preprocessing in Labelled CNF framework
Plain MUS: clause elimination

Prop: Any MUS of $M = M'$ is an MUS of $F = E(F)$.

Reconstruction is trivial.
Plain MUS: clause elimination

\[ F \]

\[ M = M' \]

\[ E \]

\[ F' \]

\[ M' \]

**Prop:** Any MUS of \( F' = E(F) \) is an MUS of \( F \).

Reconstruction is trivial.
Plain MUS: BCP

\[ F' = BCP(F, l) \]

support\(_{BCP}(C_1, F)\) support\(_{BCP}(C_2, F)\) support\(_{BCP}(C_4, F)\) support\(_{BCP}(C_5, F)\)

\( F \)

\( (l) \)

\( (l \lor C_1) \)

\( (l \lor C_2) \)

\( (l \lor C_3) \)

\( C_4 \)

\( C_5 \)

\( C_1 \)

\( C_2 \)

\( C_4 \)

\( C_5 \)

\( \ldots \) \( \ldots \) \( \ldots \)

Prop: If \( M' \) is an MUS of \( F' = BCP(F, l) \), then \( M = \bigcup_{C \in M'} \text{support}_{BCP}(C, F) \) is an MUS of \( F' \).

Pf: Take a witness \( \tau \) for \( C_2 \) in \( F' \). Then, \( \tau \cup \{l\} \) is a witness for \( (l) \), while \( \tau \cup \{\neg l\} \) is a witness for \( (l \lor C_3) \). If \( \tau \) is a witness for \( C_4 \) in \( F' \), then \( \tau \cup \{l\} \) is a witness for \( C_4 \) in \( F' \).

Reconstruction can be done in P-time.

A. Belov

Preprocessing for MUSes and MaxSAT

EPCL Training Camp, 2013 # 12
Plain MUS: BCP

\[ F' = BCP(F, l) \]
Plain MUS: BCP

$F' = \text{BCP}(F, l)$

**Prop:** If $M'$ is an MUS of $F' = \text{BCP}(F, l)$, then

$$M = \bigcup_{C \in M'} \text{support}_{\text{BCP}}(C, F)$$

is an MUS of $F$. 

A. Belov

Preprocessing for MUSes and MaxSAT

EPCL Training Camp, 2013

# 12
Prop: If $M'$ is an MUS of $F' = \text{BCP}(F, l)$, then
$$M = \bigcup_{C \in M', \text{support}_{\text{BCP}}(C, F)}$$
is an MUS of $F$.

Pf: Take a witness $\tau$ for $C_2$ in $F'$. Then, $\tau \cup \{l\}$ is a witness for $(\neg l \lor C_2)$, while $\tau \cup \{-l\}$ is a witness for $(l)$. If $\tau$ is a witness for $C_4$ in $F'$, then $\tau \cup \{l\}$ is a witness for $C_4$ in $F$. 

Reconstruction can be done in P-time.
Prop: If $M'$ is an MUS of $F' = \text{BCP}(F, l)$, then

$$M = \bigcup_{C \in M'} \text{support}_{\text{BCP}}(C, F)$$

is an MUS of $F$.

Pf: Take a witness $\tau$ for $C_2$ in $F'$. Then, $\tau \cup \{l\}$ is a witness for $(\neg l \lor C_2)$, while $\tau \cup \{\neg l\}$ is a witness for $(l)$. If $\tau$ is a witness for $C_4$ in $F'$, then $\tau \cup \{l\}$ is a witness for $C_4$ in $F$.

Reconstruction can be done in P-time.
Group-MUS: *monotone* clause elimination

**Def:** A clause elimination procedure $E$ is *monotone* iff for any $F' \subseteq F$, $E(F') \subseteq E(F)$.

**Example:** BCE — if $C$ is blocked in $F$, its blocked in any $F' \subseteq F$.

**Non-example:** SUB — if $C_1 \subset C_2$ in $F$, but $C_1$ is not in $F'$.
Group-MUS: *monotone* clause elimination

**Def:** A clause elimination procedure $E$ is *monotone* iff for any $F' \subseteq F$, $E(F') \subseteq E(F)$.

**Example:** BCE — if $C$ is blocked in $F$, its blocked in any $F' \subseteq F$.

**Non-example:** SUB — if $C_1 \subset C_2$ in $F$, but $C_1$ is not in $F'$.

**Prop:** If $E$ is monotone, then $\text{MUS}(E(F)) = \text{MUS}(F)$, i.e. any monotone clause elimination procedure is *MUS-preserving*.

**Pf:** if $M \in \text{MUS}(E(F))$, then $M \subseteq E(F) \subseteq F$, i.e. $M \in \text{MUS}(F)$ (doesn’t matter that $E$ is monotone); if $M \in \text{MUS}(F)$, then $E(M) = M$ because $E$ is SAT-preserving, and since $M \subseteq F$, we have $E(M) \subseteq E(F)$ by monotonicity, i.e. $M \in \text{MUS}(E(F))$.
Given a group-MUS $M'$ of $F' = E(F)$, let $M = \{G_i \in F | G_i' \in M'\}$.

Prop: If $E$ is MUS-preserving, then $M$ is a group-MUS of $F$.

Note: in particular, this is true if $E$ is monotone (e.g. BCE).
Group-MUS: monotone clause elimination

Given a group-MUS $M$ of $F = E(F)$, let $M = \{G_i \in F | G'_i \in M'\}$. 

Prop: If $E$ is MUS-preserving, then $M$ is a group-MUS of $F$. 

Note: in particular, this is true if $E$ is monotone (e.g. BCE).
Given a group-MUS $M'$ of $F' = E(F)$, let $M = \{ G_i \in F \mid G'_i \in M' \}$.
Given a group-MUS $M'$ of $F' = E(F)$, let $M = \{ G_i \in F \mid G'_i \in M' \}$.

**Prop:** If $E$ is MUS-preserving, then $M$ is a group-MUS of $F$.

**Note:** in particular, this is true if $E$ is monotone (e.g. BCE).
MaxSAT: *monotone* clause elimination

If $E$ is MUS-preserving, and $\tau$ is a MaxSAT solution for $E(F)$, then $\alpha_E(\tau)$ is a MaxSAT solution for $F$.

**Pf:**

$E$ is MUS-preserving $\implies E$ is MCS-preserving (HS-duality), and so min-cost MCS is the same. Only MSS clauses will be eliminated. $\tau$ is a model of the MSS of $E(F)$, $\alpha_E(\tau)$ must be a model of corresponding MSS of $F$.

Note: in particular, this is true if $E$ is monotone (e.g. BCE).
MaxSAT: monotone clause elimination

If \( E \) is MUS-preserving, and \( \tau \) is a MaxSAT solution for \( E(F) \), then \( \alpha(E) \) is a MaxSAT solution for \( F \).

Proof:
- \( E \) is MUS-preserving \( \Rightarrow \) \( E \) is MCS-preserving (HS-duality), and so min-cost MCS is the same. Only MSS clauses will be eliminated.
- \( \tau \) is a model of the MSS of \( E(F) \), \( \alpha(E) \) must be a model of corresponding MSS of \( F \).

Note: in particular, this is true if \( E \) is monotone (e.g. BCE).
MaxSAT: *monotone* clause elimination

Prop: If $E$ is MUS-preserving, and $\tau$ is a MaxSAT solution for $E(F)$, then $\alpha_E(\tau)$ is a MaxSAT solution for $F$.

Pf: $E$ is MUS-preserving $\Rightarrow E$ is MCS-preserving (HS-duality), and so min-cost MCS is the same. Only MSS clauses will be eliminated. $\tau$ is a model of the MSS of $E(F)$, $\alpha_E(\tau)$ must be a model of corresponding MSS of $F$. 
MaxSAT: monotone clause elimination

**Prop:** If $E$ is MUS-preserving, and $\tau$ is a MaxSAT solution for $E(F)$, then $\alpha_E(\tau)$ is a MaxSAT solution for $F$.

**Pf:** $E$ is MUS-preserving $\Rightarrow$ $E$ is MCS-preserving (HS-duality), and so min-cost MCS is the same. Only MSS clauses will be eliminated. $\tau$ is a model of the MSS of $E(F)$, $\alpha_E(\tau)$ must be a model of corresponding MSS of $F$.

**Note:** in particular, this is true if $E$ is monotone (e.g. BCE).
Outline

Introduction: MUSes, MaxSAT, SAT preprocessing

Direct reconstruction: techniques that work

Direct reconstruction: techniques that break

Sound preprocessing in Labelled CNF framework
Plain MUS: SSR

\[ F = (\neg x \lor p) \quad C_1 = (x \lor p \lor q) \quad C_2 = (\neg p) \quad (x \lor \neg q) \quad (\neg x) \]
Plain MUS: SSR

\[ F = \neg x \lor p \quad C_1 = (x \lor p \lor q) \quad C_2 = \neg p \quad (x \lor \neg q) \quad \neg x \]

\[ F' = \text{SSR}(F, C_1, C_2, x) \]

\[ F' = \neg x \lor p \quad (p \lor q) \quad \neg p \quad (x \lor \neg q) \quad \neg x \]
Plain MUS: SSR

\[ F' = SSR(F, C_1, C_2, x) \]
Plain MUS: SSR

\[ F' = SSR(F, C_1, C_2, x) \]
Plain MUS: SSR

\[ F' = \text{SSR}(F, C_1, C_2, x) \]
Plain MUS: SSR

\[ F' = SSR(F, C_1, C_2, x) \]
Plain MUS: SSR

\[ F' = \text{SSR}(F, C_1, C_2, x) \]
Plain MUS: SSR

For an MUS $M'$ of $F' = SSR(F, C, D, l)$, $M = \bigcup_{C \in M'} \text{support}_{SSR}(C, F)$ might be not an MUS of $F$. 
Plain MUS: VE

Canonical CNF on \( p, q, r \) without \( (p \lor q \lor r) \) and \( (p \lor \neg q \lor r) \)

\[ F' = \text{VE}(F, x) \]
Plain MUS: VE

$F = \{(x \lor p \lor q \lor r), (x \lor \neg q \lor r), (\neg x \lor p \lor s), (\neg x \lor q), (\neg s)\}$

MUS

Canonical CNF on $p, q, r$ without $(p \lor q \lor r)$ and $(p \lor \neg q \lor r)$

$F' = \text{VE}(F, x)$
Not an MUS

Canonical CNF on p, q, r without
(p ∨ q ∨ r) and (p ∨ ¬q ∨ r)

F

F' = VE(F, x)

A. Belov
Preprocessing for MUSes and MaxSAT
EPCL Training Camp, 2013  # 18
For an MUS $M'$ of $F' = \text{VE}(F, x)$,  

$$M = \bigcup_{C \in M'} \text{support}_{\text{VE}}(C, F)$$

might be not an MUS of $F$.

Doesn’t work even if $M$ is greedily minimized to avoid duplicate resolvents.
Group-MUS: subsumption

\[ (\neg r) \quad (\neg p \lor q) \quad (\neg q \lor r) \quad (p \lor r) \quad \text{(p)} \]

\[ (\neg r) \quad (\neg p \lor q) \quad (\neg q \lor r) \quad \text{(p)} \]

\[ \text{SUBS}(F) \]
Not a group-MUS

For a group-MUS $M'$ of $F' = E(F)$, $M = \{ G_i \in F | G'_i \in F' \}$ might be not a group-MUS of $F$.

Note: can do certain things with subsumption: eliminate within a group, or from group-0 to other groups.
For a group-MUS $M'$ of $F' = E(F)$,

$$M = \{ G_i \in F \mid G'_i \in F' \}$$

might be not a group-MUS of $F$.

**Note:** can do *certain* things with subsumption: eliminate *within* a group, or from group-0 to other groups.
Group MUS: BCP

\[
F = G_0 \lor (y) \lor (z \lor p \lor q) \lor (\neg z) \lor (\neg y \lor \neg z) \lor (p \lor \neg q) \lor (\neg p \lor q) \lor (\neg p \lor \neg q)
\]

\[
G_0' \lor (y) \lor (p \lor q) \lor (p \lor \neg q) \lor (\neg p \lor q) \lor (\neg p \lor \neg q)
\]

\[
F' = BCP(F, \neg z)
\]
Group MUS: BCP

\[
\begin{align*}
F &= (y) (z \lor p \lor q) (\neg z) (\neg y \lor \neg z) (p \lor \neg q) (\neg p \lor q) (\neg p \lor \neg q) \\
G_0' &= (y) (p \lor q) (p \lor \neg q) (\neg p \lor q) (\neg p \lor \neg q) \\
G_1' &= (p \lor q) (p \lor \neg q) (\neg p \lor q) (\neg p \lor \neg q) \\
G_2' &= (\neg z) (\neg y \lor \neg z) (p \lor \neg q) (\neg p \lor q) (\neg p \lor \neg q) \\
G_3' &= (\neg z) (\neg y \lor \neg z) (p \lor \neg q) (\neg p \lor q) (\neg p \lor \neg q)
\end{align*}
\]

\[F' = \text{BCP}(F, \neg z)\]
BCP in the group-MUS setting is problematic.

Note: again, can do certain things with BCP: propagate within a group, or from group-0 to other groups.
Group MUS: BCP

BCP in the group-MUS setting is problematic.

**Note:** again, can do *certain* things with BCP: propagate *within* a group, or from group-0 to other groups.

**Note:** VE and SSR didn’t work for plain MUSes, and so won’t work for group-MUSes either.
MaxSAT: subsumption

$$F = (p) \quad (\neg p) \quad (p \lor q) \quad (p \lor \neg q) \quad (r) \quad (\neg r)$$
MaxSAT: subsumption

\[ F = (p) \land (\neg p) \land (p \lor q) \land (p \lor \neg q) \land (r) \land (\neg r) \]

\[
\begin{array}{cccc}
(p) & (\neg p) & (p \lor q) & (p \lor \neg q) \\
(\neg p) & (p \lor q) & (p \lor \neg q) & (r) \\
(p \lor \neg q) & (r) & (\neg r) & (\neg r) \\
\end{array}
\]

\[ \text{SUBS}(F) \]

\[ \tau = \{ \neg p, \neg r \} \text{ is a MaxSAT solution for SUBS}(F), \text{cost}(\tau) = 2. \]

\[ \tau' = \{ \neg p, \neg r, q \} \text{ is not a MaxSAT solution for } F, \text{cost}(\tau') = 3, \text{ but } \text{cost}\left(\{p, \neg r, q\}\right) = 2. \]

Reason: subsumption may remove some of the MUSes of \( F \).
MaxSAT: subsumption

\[ \tau = \{ \neg p, \neg r \} \text{ is a MaxSAT solution for } SUBS(F), \quad \text{cost}(\tau) = 2. \]
MaxSAT: subsumption

\[ F = \{ p, \neg p, p \lor q, p \lor \neg q, r, \neg r \} \]

\[ \text{SUBS}(F) \]

\[ \tau = \{ \neg p, \neg r \} \text{ is a MaxSAT solution for } \text{SUBS}(F), \; \text{cost}(\tau) = 2. \]

\[ \tau' = \{ \neg p, \neg r, q \} \text{ is not a MaxSAT solution for } F, \; \text{cost}(\tau') = 3, \text{ but } \text{cost}(\{ p, \neg r, q \}) = 2. \]
MaxSAT: subsumption

\[ F = (p) (\neg p) (p \lor q) (p \lor \neg q) (r) (\neg r) \]

\[ \tau = \{ \neg p, \neg r \} \] is a MaxSAT solution for \( SUBS(F) \), \( \text{cost}(\tau) = 2 \).

\[ \tau' = \{ \neg p, \neg r, q \} \] is not a MaxSAT solution for \( F \), \( \text{cost}(\tau') = 3 \), but \( \text{cost}(\{ p, \neg r, q \}) = 2 \).

**Reason:** subsumption may remove some of the MUSes of \( F \).
MaxSAT: VE

\[ F = (p) (\neg p) (p \lor q) (p \lor \neg q) (r) (\neg r) \]

\( \tau = \{ \neg p, \neg r \} \) is a MaxSAT solution for \( \text{VE}(F, q) \), cost \( (\tau) = 2 \).

\( \tau' = \{ \neg p, \neg r, q \} \) is not a MaxSAT solution for \( F \), cost \( (\tau') = 3 \), but cost \( \{ p, \neg r, q \} = 2 \).

Reason: VE changes the MUSes of \( F \). Additional problem: what to do with weights?

A. Belov

Preprocessing for MUSes and MaxSAT

EPCL Training Camp, 2013
$\text{MaxSAT: VE}$

$F$

\[
\begin{array}{cccccc}
(p) & (\neg p) & (p \lor q) & (p \lor \neg q) & (r) & (\neg r) \\
(p) & (\neg p) & (r) & (\neg r) \end{array}
\]

$VE(F, q)$

$\tau = \{\neg p, \neg r\}$ is a MaxSAT solution for $VE(F, q)$, $cost(\tau) = 2$.

$\tau' = \{\neg p, \neg r, q\}$ is not a MaxSAT solution for $F$, $cost(\tau') = 3$, but $cost(\{p, \neg r, q\}) = 2$.

Reason: $VE$ changes the MUSes of $F$. Additional problem: what to do with weights?

A. Belov

Preprocessing for MUSes and MaxSAT

EPCL Training Camp, 2013

# 22
MaxSAT: VE

\[ F = \{ p, \neg p, p \lor q, p \lor \neg q, r, \neg r \} \]

\[ VE(F, q) \]

\[ \tau = \{ \neg p, \neg r \} \] is a MaxSAT solution for \( VE(F, q) \), \( cost(\tau) = 2 \).
τ = \{-p, \neg r\} is a MaxSAT solution for \(VE(F, q)\), \(cost(\tau) = 2\).

\(\tau' = \{-p, \neg r, q\}\) is not a MaxSAT solution for \(F\), \(cost(\tau') = 3\), but 
\(cost(\{p, \neg r, q\}) = 2\).
\( \tau = \{\neg p, \neg r\} \) is a MaxSAT solution for \( VE(F, q) \), \( cost(\tau) = 2 \).

\( \tau' = \{\neg p, \neg r, q\} \) is not a MaxSAT solution for \( F \), \( cost(\tau') = 3 \), but 
\( cost(\{p, \neg r, q\}) = 2 \).

**Reason:** \( VE \) changes the MUSes of \( F \). Additional problem: what to do with weights?
MaxSAT: resolution-based preprocessing

None of the resolution-based techniques are *sound* for MaxSAT, because the resolution rule itself is not sound.

\[(x \lor C_1) \otimes_x (\neg x \lor C_2) = (C_1 \lor C_2)\]

Consider an assignment \(\tau\) s.t. \(\tau(\neg x \lor C_2) = 0\) and \(\tau(C_1) = 1\).
MaxSAT: resolution-based preprocessing

None of the resolution-based techniques are sound for MaxSAT, because the resolution rule itself is not sound.

\[(x \lor C_1) \otimes_x (\neg x \lor C_2) = (C_1 \lor C_2)\]

Consider an assignment \(\tau\) s.t. \(\tau(\neg x \lor C_2) = 0\) and \(\tau(C_1) = 1\).

**MaxSAT Resolution rule** [Bonet et al, 07]

\[
\begin{align*}
x \lor a_1 \lor \cdots \lor a_s \\
\bar{x} \lor b_1 \lor \cdots \lor b_t \\
\hline
 a_1 \lor \cdots \lor a_s \lor b_1 \lor \cdots \lor b_t \\
x \lor a_1 \lor \cdots \lor a_s \lor \bar{b}_1 \\
x \lor a_1 \lor \cdots \lor a_s \lor b_1 \lor \bar{b}_2 \\
\cdots \\
x \lor a_1 \lor \cdots \lor a_s \lor b_1 \lor \cdots \lor b_{t-1} \lor \bar{b}_t \\
\bar{x} \lor b_1 \lor \cdots \lor b_t \lor \bar{a}_1 \\
\bar{x} \lor b_1 \lor \cdots \lor b_t \lor a_1 \lor \bar{a}_2 \\
\cdots \\
\bar{x} \lor b_1 \lor \cdots \lor b_t \lor a_1 \lor \cdots \lor a_{s-1} \lor \bar{a}_s 
\end{align*}
\]
None of the resolution-based techniques are *sound* for MaxSAT, because the resolution rule itself is not sound.

\[(x \lor C_1) \otimes_x (\neg x \lor C_2) = (C_1 \lor C_2)\]

Consider an assignment \(\tau\) s.t. \(\tau(\neg x \lor C_2) = 0\) and \(\tau(C_1) = 1\).

**MaxSAT Resolution rule** [Bonet et al, 07]

\[\begin{align*}
x \lor a_1 \lor \cdots \lor a_s \\
\bar{x} \lor b_1 \lor \cdots \lor b_t
\end{align*}\]

\[\begin{array}{c}
a_1 \lor \cdots \lor a_s \lor b_1 \lor \cdots \lor b_t \\
x \lor a_1 \lor \cdots \lor a_s \lor \bar{b}_1 \\
x \lor a_1 \lor \cdots \lor a_s \lor b_1 \lor \bar{b}_2 \\
\vdots \\
x \lor a_1 \lor \cdots \lor a_s \lor b_1 \lor \cdots \lor b_{t-1} \lor \bar{b}_t \\
\bar{x} \lor b_1 \lor \cdots \lor b_t \lor \bar{a}_1 \\
\bar{x} \lor b_1 \lor \cdots \lor b_t \lor a_1 \lor \bar{a}_2 \\
\vdots \\
\bar{x} \lor b_1 \lor \cdots \lor b_t \lor a_1 \lor \cdots \lor a_{s-1} \lor \bar{a}_s
\end{array}\]

Although sound and complete for MaxSAT solving, is not suitable for preprocessing.
Summary of Direct Reconstruction

Techniques that work

- **plain-MUS**: any clause elimination, BCP.
- **group-MUS**: MUS-preserving (sp. monotone) clause elimination.
- **MaxSAT**: MUS-preserving (sp. monotone) clause elimination.

Techniques that break

- **plain-MUS**: VE, SSR
- **group-MUS**: SUB, BCP, VE, SSR
- **MaxSAT**: SUB, BCP, VE, SSR
Summary of Direct Reconstruction

Techniques that work

- plain-MUS: any clause elimination, BCP.
- group-MUS: MUS-preserving (sp. monotone) clause elimination.
- MaxSAT: MUS-preserving (sp. monotone) clause elimination.

Techniques that break

- plain-MUS: VE, SSR
- group-MUS: SUB, BCP, VE, SSR
- MaxSAT: SUB, BCP, VE, SSR

Important: not known that sound direct methods are impossible.
Summary of Direct Reconstruction

Techniques that work

▶ plain-MUS: any clause elimination, BCP.
▶ group-MUS: MUS-preserving (sp. monotone) clause elimination.
▶ MaxSAT: MUS-preserving (sp. monotone) clause elimination.

Techniques that break

▶ plain-MUS: VE, SSR
▶ group-MUS: SUB, BCP, VE, SSR
▶ MaxSAT: SUB, BCP, VE, SSR

Important: not known that sound direct methods are impossible.

Note: Some special cases do work:

▶ group-MUS: within groups, or from group-0 to other groups.
▶ MaxSAT: within $F^H$, or from $F^H$ to soft clauses.

Need a generic framework for guaranteed correctness-preserving application of preprocessing techniques.
Outline

Introduction: MUSes, MaxSAT, SAT preprocessing

Direct reconstruction: techniques that work

Direct reconstruction: techniques that break

Sound preprocessing in Labelled CNF framework
Labelled CNF (LCNF) formulas

**Original motivation:** generalize group-MUS to *intersecting* groups

**Note:** those arise naturally in circuit-MUS and variable-MUS settings.

**Note:** generalizes other related problems, such as MCS computation and MaxSAT.
Labelled CNF (LCNF) formulas

Original motivation: generalize group-MUS to intersecting groups

Note: those arise naturally in circuit-MUS and variable-MUS settings.

Note: generalizes other related problems, such as MCS computation and MaxSAT.

The components of the framework

- \( Lbls \): a countable set of \textit{labels}.

- \textit{Labelled clause} \( C^L \) is a tuple \( \langle C, L \rangle \), where \( C \) is a clause, and \( L \subset Lbls \) is a finite set of labels.

- \textit{LCNF formula} \( \Phi \) is a set of labelled clauses.
  - \( Cls(\Phi) \) - the “normal” clauses of \( \Phi \), i.e. \( \bigcup_{C^L \in \Phi} C \).
  - \( Lbls(\Phi) \) - the labels of \( \Phi \), i.e. \( \bigcup_{C^L \in \Phi} L \).
Labelled CNF (LCNF) formulas

Original motivation: generalize group-MUS to intersecting groups

Note: those arise naturally in circuit-MUS and variable-MUS settings.

Note: generalizes other related problems, such as MCS computation and MaxSAT.

The components of the framework

- **Lbls**: a countable set of *labels*.
- **Labelled clause** $C^L$ is a tuple $\langle C, L \rangle$, where $C$ is a clause, and $L \subset Lbls$ is a finite set of labels.
- **LCNF formula** $\Phi$ is a set of labelled clauses.
  - $Cls(\Phi)$ - the “normal” clauses of $\Phi$, i.e. $\bigcup_{C^L \in \Phi} C$.
  - $Lbls(\Phi)$ - the labels of $\Phi$, i.e. $\bigcup_{C^L \in \Phi} L$.

Example ($Lbls = \mathbb{N}$):

$\Phi = \{(x)^0, (y)^0, (z \lor p \lor q)^1, (p \lor \neg q)^2, (\neg p \lor q)^{2,3}, (\neg p \lor \neg q)^3\}$
Def: For an LCNF $\Phi$, let $M \subseteq Lbls(\Phi)$. Then, $\Phi|_M = \{ C^L \in \Phi \mid L \subseteq M \}$ is a subformula of $\Phi$ \textit{induced} by $M$.

Alternatively: any clause that has a label \textit{outside} of $M$ is removed from $\Phi$. 

Note: Clauses with the label-set $\emptyset$ cannot be removed ⇒ a convenient way to represent $G_0$ clauses in group-MUS, or $F_H$ in MaxSAT.
LCNF formulas: induced subformulas

**Def:** For an LCNF $\Phi$, let $M \subseteq Lbls(\Phi)$. Then, $\Phi|_M = \{C^L \in \Phi \mid L \subseteq M\}$ is a subformula of $\Phi$ induced by $M$.

Alternatively: any clause that has a label outside of $M$ is removed from $\Phi$.

\[
\Phi = \begin{array}{cccccc}
(x)^0 & (y)^0 & (z \lor p \lor q)^{\{1\}} & (p \lor \neg q)^{\{2\}} & (\neg p \lor q)^{\{2,3\}} & (\neg p \lor \neg q)^{\{3\}} \\
\end{array}
\]

\[
\Phi|_{\{1,2\}} = \begin{array}{cccc}
(x)^0 & (y)^0 & (z \lor p \lor q)^{\{1\}} & (p \lor \neg q)^{\{2\}} \\
\end{array}
\]

\[
\Phi|_{\{2\}} = \begin{array}{cccc}
(x)^0 & (y)^0 & (p \lor \neg q)^{\{2\}} \\
\end{array}
\]

Note: Clauses with the label-set $\emptyset$ cannot be removed. A convenient way to represent $G_0$ clauses in group-MUS, or $F_H$ in MaxSAT.
**Def:** For an LCNF $\Phi$, let $M \subseteq Lbls(\Phi)$. Then, $\Phi\vert_M = \{C^L \in \Phi \mid L \subseteq M\}$ is a subformula of $\Phi$ _induced_ by $M$.

Alternatively: any clause that has a label _outside_ of $M$ is removed from $\Phi$.

\[
\Phi = \begin{cases} 
(x)^\emptyset & (y)^\emptyset & (z \lor p \lor q)^{1} & (p \lor \neg q)^{2} & (-p \lor q)^{2,3} & (-p \lor \neg q)^{3} 
\end{cases}
\]

\[
\Phi\vert_{\{1,2\}} = \begin{cases} 
(x)^\emptyset & (y)^\emptyset & (z \lor p \lor q)^{1} & (p \lor \neg q)^{2} 
\end{cases}
\]

\[
\Phi\vert_{\{2\}} = \begin{cases} 
(x)^\emptyset & (y)^\emptyset & (p \lor \neg q)^{2} 
\end{cases}
\]

**Note:** Clauses with the label-set $\emptyset$ cannot be removed $\Rightarrow$ a convenient way to represents $G_0$ clauses in group-MUS, or $F^H$ in MaxSAT.
Satisfiability: LCNF $\Phi$ is SAT iff $\text{Cls}(\Phi)$ is SAT.

**Def:** $M \subseteq Lbls(\Phi)$ is an *minimal unsatisfiable subformula (MUS)* of $\Phi$ if $\Phi|_M \in \text{UNSAT}$, and $\forall l \in M$, $\Phi|_{M\setminus\{l\}} \in \text{SAT}$

i.e. the removal of *any* label from $\Phi|_M$ makes it SAT.
Satisfiability: LCNF $\Phi$ is SAT iff $\text{Cls}(\Phi)$ is SAT.

**Def:** $M \subseteq \text{Lbls}(\Phi)$ is an **minimal unsatisfiable subformula (MUS)** of $\Phi$ if $\Phi|_M \in \text{UNSAT}$, and $\forall l \in M$, $\Phi|_{M\setminus\{l\}} \in \text{SAT}$

I.e. the removal of any label from $\Phi|_M$ makes it SAT.

Natural mapping between CNF/group-CNФ and LCNФ:

$$
\begin{array}{cccccc}
C_1 & C_2 & C_3 & C_4 & C_5 & C_6 \\
(p) & (q) & (\neg p \lor \neg q) & (\neg p \lor r) & (p \lor q) & (\neg q \lor \neg r)
\end{array}
$$

MUS is \{ $C_1$, $C_2$, $C_4$, $C_6$ \}

$$
\begin{array}{cccccc}
\Phi & (p)^1 & (q)^2 & (\neg p \lor \neg q)^3 & (\neg p \lor r)^4 & (p \lor q)^5 & (\neg q \lor \neg r)^6
\end{array}
$$

LMUS is \{ 1, 2, 4, 6 \}
Satisfiability: LCNF $\Phi$ is SAT iff $\mathrm{Cls}(\Phi)$ is SAT.

**Def:** $M \subseteq \mathrm{Lbls}(\Phi)$ is an *minimal unsatisfiable subformula (MUS)* of $\Phi$ if $\Phi|_M \in \mathrm{UNSAT}$, and $\forall l \in M$, $\Phi|_{M\setminus\{l\}} \in \mathrm{SAT}$

I.e. the removal of any label from $\Phi|_M$ makes it SAT.

Natural mapping between CNF/group-CNF and LCNF:

\[
\begin{align*}
G_0 & : \quad (p) \quad (q) \\
G_1 & : \quad (\neg p \lor \neg q) \quad (\neg p \lor r) \\
G_2 & : \quad (p \lor q) \quad (\neg q \lor \neg r)
\end{align*}
\]

group-MUS is \{ $G_1$ \}

$\Phi$\[\begin{array}{cccccc}
(p^0) & (q^0) & (\neg p \lor \neg q)^{(1)} & (\neg p \lor r)^{(1)} & (p \lor q)^{(2)} & (\neg q \lor \neg r)^{(2)}
\end{array}\]

LMUS is \{ 1 \}
LCNF formulas: MCSes and MaxSAT

**Def:** $S \subseteq Lbls(\Phi)$ is a **maximal satisfiable subformula (MSS)** of $\Phi$ if $\Phi|_S \in \text{SAT}$, and $\forall l \in Lbls(\Phi) \setminus S$, $\Phi|_{S\cup\{l\}} \in \text{UNSAT}$.

**Def:** $R \subseteq Lbls(\Phi)$ is a **minimal correction subset (MCS)** of $\Phi$ if $\Phi|_{Lbls(\Phi)\setminus R} \in \text{SAT}$, and $\forall l \in R$, $\Phi|_{(Lbls(\Phi)\setminus R)\cup\{l\}} \in \text{UNSAT}$.

**Note:** hitting sets duality between MUSes and MCSes holds for LCNFs.
LCNF formulas: MCSes and MaxSAT

**Def:** $S \subseteq \text{Lbls}(\Phi)$ is a **maximal satisfiable subformula (MSS)** of $\Phi$ if $\Phi|_S \in \text{SAT}$, and $\forall l \in \text{Lbls}(\Phi) \setminus S$, $\Phi|_{S \cup \{l\}} \in \text{UNSAT}$.

**Def:** $R \subseteq \text{Lbls}(\Phi)$ is a **minimal correction subset (MCS)** of $\Phi$ if $\Phi|_{\text{Lbls}(\Phi) \setminus R} \in \text{SAT}$, and $\forall l \in R$, $\Phi|_{(\text{Lbls}(\Phi) \setminus R) \cup \{l\}} \in \text{UNSAT}$.

**Note:** hitting sets duality between MUSes and MCSes holds for LCNFs.

MaxSAT for LCNFs:

- each label $l$ has a **weight** $w(l) \in \mathbb{N}^+$;
- the **cost** of a set of labels $L$ is $\sum_{l \in L} w(l)$.
- weighted LCNF: just an LCNF + the weights for labels.
- a **MaxSAT solution** for $\Phi$ is a model of an MSS of the maximum cost. Alternatively: falsifies only the clauses of an MCS of the minimum cost.
LCNF formulas: MCSes and MaxSAT

**Def:** \( S \subseteq Lbls(\Phi) \) is a *maximal satisfiable subformula (MSS)* of \( \Phi \) if \( \Phi|_S \in \text{SAT} \), and \( \forall l \in Lbls(\Phi) \setminus S \), \( \Phi|_{S \cup \{l\}} \in \text{UNSAT} \).

**Def:** \( R \subseteq Lbls(\Phi) \) is a *minimal correction subset (MCS)* of \( \Phi \) if \( \Phi|_{Lbls(\Phi) \setminus R} \in \text{SAT} \), and \( \forall l \in R \), \( \Phi|_{(Lbls(\Phi) \setminus R) \cup \{l\}} \in \text{UNSAT} \).

**Note:** hitting sets duality between MUSes and MCSes holds for LCNFs.

MaxSAT for LCNFs:
- each *label* \( l \) has a *weight* \( w(l) \in \mathbb{N}^+ \);
- the *cost* of a set of labels \( L \) is \( \sum_{l \in L} w(l) \).
- weighted LCNF: just an LCNF + the weights for labels.
- a *MaxSAT solution* for \( \Phi \) is a model of an MSS of the *maximum cost*. Alternatively: falsifies only the clauses of an MCS of the *minimum cost*.

**Note:** a natural mapping between MaxSAT instances and weighted LCNFs.
Main point: For LCNF formulas MUSes and MCSes are preserved under a (suitably defined) preprocessing techniques.
Main point: For LCNF formulas MUSes and MCSes are preserved under a (suitably defined) preprocessing techniques.

Subsumption elimination

Theorem: Let $C_{1}^{L_{1}}$ and $C_{2}^{L_{2}}$ in $\Phi$ be such that $C_{1} \subseteq C_{2}$ and $L_{1} \subseteq L_{2}$. Then, any MUS of $\Phi \setminus \{C_{2}^{L_{2}}\}$ is an MUS of $\Phi$, and vice versa.

Note: by hitting sets duality, $\text{MCS}(\Phi \setminus \{C_{2}^{L_{2}}\}) = \text{MCS}(\Phi)$. Any MaxSAT solution $\tau$ for $\Phi \setminus \{C_{2}^{L_{2}}\}$ is a MaxSAT solution for $\Phi$. 

▶ I.e. subsumption is OK, as long as the labels also have a subset relationship.

▶ Effectively, blocks subsumption across groups in group-MUS, and across soft clauses in MaxSAT (good).

▶ Blocks subsumption in plain MUS instances: a weaker condition that allows for it seems to be possible.
LCNF formulas: preprocessing

**Main point:** For LCNF formulas MUSes and MCSes are *preserved* under a (suitably defined) preprocessing techniques.

**Subsumption elimination**

**Theorem:** Let $C_1^{L_1}$ and $C_2^{L_2}$ in $\Phi$ be such that $C_1 \subseteq C_2$ and $L_1 \subseteq L_2$. Then, any MUS of $\Phi \setminus \{C_2^{L_2}\}$ is an MUS of $\Phi$, and vice versa.

**Note:** by hitting sets duality, $\text{MCS}(\Phi \setminus \{C_2^{L_2}\}) = \text{MCS}(\Phi)$. Any MaxSAT solution $\tau$ for $\Phi \setminus \{C_2^{L_2}\}$ is a MaxSAT solution for $\Phi$.

- I.e. subsumption is OK, as long as the labels also have a subset relationship.
- Effectively, blocks subsumption across groups in group-MUS, and across soft clauses in MaxSAT (good).
- Blocks subsumption in plain MUS instances: a weaker condition that allows for it seems to be possible.
**Def:** Resolution: $(x \lor C_1)^{L_1} \otimes_x (\neg x \lor C_2)^{L_2} = (C_1 \lor C_2)^{L_1 \cup L_2}$.

**Def:** Variable elimination: $\text{VE}(\Phi, x) = \Phi \cup (\Phi_x \otimes_x \Phi_{\neg x}) \setminus (\Phi_x \cup \Phi_{\neg x})$. 

Theorem: Any LMUS of $\text{LVE}(\Phi, x)$ is an LMUS of $\Phi$, and vice versa.

Note: by hitting sets duality, $\text{MCS}(\text{VE}(\Phi, x)) = \text{MCS}(\Phi)$.

For any MaxSAT solution $\tau$ for $\text{VE}(\Phi, x)$ there is a MaxSAT solution for $\Phi$ with exactly the same cost (exactly the same MCS).

To get a solution for $\Phi$ (the model of the MSS) do the standard reconstruction after VE.
Def: Resolution: \((x \lor C_1)^{L_1} \otimes_x (\neg x \lor C_2)^{L_2} = (C_1 \lor C_2)^{L_1 \cup L_2}\).

Def: Variable elimination: \(VE(\Phi, x) = \Phi \cup (\Phi_x \otimes_x \Phi_{\neg x}) \setminus (\Phi_x \cup \Phi_{\neg x})\).

Theorem: Any LMUS of \(LVE(\Phi, x)\) is an LMUS of \(\Phi\), and vice versa.
**Def:** Resolution: \((x \lor C_1)^{L_1} \otimes_x (^\neg x \lor C_2)^{L_2} = (C_1 \lor C_2)^{L_1 \cup L_2}\).

**Def:** Variable elimination: \(\text{VE}(\Phi, x) = \Phi \cup (\Phi_x \otimes_x \Phi_{^\neg x}) \setminus (\Phi_x \cup \Phi_{^\neg x})\).

**Theorem:** Any LMUS of \(\text{LVE}(\Phi, x)\) is an LMUS of \(\Phi\), and vice versa.

**Note:** by hitting sets duality, \(\text{MCS(VE(\Phi, x))} = \text{MCS(\Phi)}\).

For any MaxSAT solution \(\tau\) for \(\text{VE}(\Phi, x)\) there is a MaxSAT solution for \(\Phi\) with exactly the same cost (exactly the same MCS).

To get a solution for \(\Phi\) (the model of the MSS) do the standard reconstruction after VE.
Remember the bad example for VE?

For an MUS $M'$ of $F' = \text{VE}(F, x)$,

$$M = \bigcup_{C \in M'} \text{support}_{\text{VE}}(C, F)$$

might be not an MUS of $F$. 

Canonical CNF on $p, q, r$ without $(p \lor q \lor r)$ and $(p \lor \neg q \lor r)$

Not an MUS

Redundant

$F' = \text{VE}(F, x)$
Def: Resolution: \((x \lor C_1)^{L_1} \otimes_x (\neg x \lor C_2)^{L_2} = (C_1 \lor C_2)^{L_1 \cup L_2}\).

Def: Variable elimination: \(\text{VE}(\Phi, x) = \Phi \cup (\Phi_x \otimes_x \Phi_{\neg x}) \setminus (\Phi_x \cup \Phi_{\neg x})\).

Theorem: Any LMUS of \(\text{LVE}(\Phi, x)\) is an LMUS of \(\Phi\), and vice versa.
Def: Resolution: \((x \lor C_1)^{L_1} \otimes_x (\neg x \lor C_2)^{L_2} = (C_1 \lor C_2)^{L_1 \cup L_2}\).

Def: Variable elimination: \(\text{VE}(\Phi, x) = \Phi \cup (\Phi_x \otimes_x \Phi_{\neg x}) \setminus (\Phi_x \cup \Phi_{\neg x})\).

Theorem: Any LMUS of \(\text{LVE}(\Phi, x)\) is an LMUS of \(\Phi\), and vice versa.

\[
\begin{align*}
(x \lor p \lor q \lor r)^{\{1\}} & \quad (x \lor \neg q \lor r)^{\{2\}} & \quad (\neg x \lor p \lor s)^{\{3\}} & \quad (\neg x \lor q)^{\{4\}} & \quad (\neg s)^{\{5\}} \\
(p \lor q \lor r \lor s)^{\{1,3\}} & \quad (p \lor q \lor r)^{\{1,4\}} & \quad (p \lor \neg q \lor r \lor s)^{\{2,3\}} & \quad (\neg s)^{\{5\}} & \quad R
\end{align*}
\]

\(\Phi' = \text{LVE}(\Phi, x)\)

Label 4 is redundant! The LMUS of \(\Phi'\) is \(\{1, 2, 3, 5, \ldots\}\) – same as for \(\Phi\).
LCNF formulas: preprocessing

BCP and SSR can also be defined for LCNFs so that the correctness is preserved.

**Bottom line:** labels enable correctness-preserving preprocessing for MUS and group-MUS extraction, and for computing MaxSAT solutions.
LCNF formulas: preprocessing

BCP and SSR can also be defined for LCNFs so that the correctness is preserved.

**Bottom line:** labels enable correctness-preserving preprocessing for MUS and group-MUS extraction, and for computing MaxSAT solutions.

**Preprocessing flow**

Given $F$, a (group-)MUS instance or an instance of MaxSAT:

1. Run any MUS-preserving clause elimination procedure (like BCE).
2. For plain-MUSes may also run SUB and BCP (while keeping trace).
3. Convert the resulting formula $F_1$ to LCNF $\Phi_1$.
4. Run VE, SSR on $\Phi_1$ to get $\Phi_2$.
5. Compute an MUS $M$ of $\Phi_2$ or a MaxSAT solution $\tau$ for $\Phi_2$.
6. The labels in $M$ represent a (group-)MUS of $F_1$. The cost of a MaxSAT solution of $F_1$ is the same as the cost of $\tau$; to get a solution do the standard reconstruction after VE.
LCNF formulas: preprocessing

BCP and SSR can also be defined for LCNFs so that the correctness is preserved.

**Bottom line:** labels enable correctness-preserving preprocessing for MUS and group-MUS extraction, and for computing MaxSAT solutions.

**Preprocessing flow**
Given $F$, a (group-)MUS instance or an instance of MaxSAT:

1. Run any MUS-preserving clause elimination procedure (like BCE).
2. For plain-MUSes may also run SUB and BCP (while keeping trace).
3. Convert the resulting formula $F_1$ to LCNF $\Phi_1$.
4. Run VE, SSR on $\Phi_1$ to get $\Phi_2$.
5. **Compute an MUS $M$ of $\Phi_2$ or a MaxSAT solution $\tau$ for $\Phi_2$.**
6. The labels in $M$ represent a (group-)MUS of $F_1$. The cost of a MaxSAT solution of $F_1$ is the same as the cost of $\tau$; to get a solution do the standard reconstruction after VE.
Given an LCNF $\Phi$, for each label $l_i \in Lbls(\Phi)$, $1 \leq i \leq n$, introduce a fresh variable $p_i$.

Then, convert each labelled clause $C\{l_1,\ldots,l_k\}$ in $\Phi$ into a clause $(p_1 \lor \cdots \lor p_k \lor C)$. Let $F_\Phi$ denote the resulting CNF.
LCNF formulas: computing MUSes

Given an LCNF $\Phi$, for each label $l_i \in Lbls(\Phi)$, $1 \leq i \leq n$, introduce a fresh variable $p_i$.

Then, convert each labelled clause $C\{l_1, \ldots, l_k\}$ in $\Phi$ into a clause $(p_1 \lor \cdots \lor p_k \lor C)$. Let $F_\Phi$ denote the resulting CNF.

Reduction to group-MUS problem
Let $G_0 = F_\Phi$, and for each $l_i \in Lbls(\Phi)$, let $G_i = \{(\neg p_i)\}$.

Compute a group-MUS of $\{G_0, G_1, \ldots, G_n\}$: if $G_i$ is in, then $l_i$ is in.
Given an LCNF $\Phi$, for each label $l_i \in Lbls(\Phi), 1 \leq i \leq n$, introduce a fresh variable $p_i$.

Then, convert each labelled clause $C\{l_1, \ldots, l_k\}$ in $\Phi$ into a clause $(p_1 \lor \cdots \lor p_k \lor C)$. Let $F_\Phi$ denote the resulting CNF.

**Reduction to group-MUS problem**
Let $G_0 = F_\Phi$, and for each $l_i \in Lbls(\Phi)$, let $G_i = \{\neg p_i\}$.
Compute a group-MUS of $\{G_0, G_1, \ldots, G_n\}$: if $G_i$ is in, then $l_i$ is in.

**Direct computation**
Load $F_\Phi$ into an incremental SAT solver, and use $p_i$'s as assumptions.

**Note:** this is how most of the MUS extractors compute MUSes and group-MUSes anyway.
LCNF formulas: computing MaxSAT solutions

Given a weighted LCNF $\Phi$, for each label $l_i \in Lbls(\Phi)$, $1 \leq i \leq n$, introduce a fresh variable $p_i$.

Then, convert each labelled clause $C\{l_1,\ldots,l_k\}$ in $\Phi$ into a clause $(p_1 \lor \cdots \lor p_k \lor C)$. Let $F_{\Phi}$ denote the resulting CNF.
LCNF formulas: computing MaxSAT solutions

Given a weighted LCNF $\Phi$, for each label $l_i \in Lbls(\Phi)$, $1 \leq i \leq n$, introduce a fresh variable $p_i$.

Then, convert each labelled clause $C\{l_1, \ldots, l_k\}$ in $\Phi$ into a clause $(p_1 \lor \cdots \lor p_k \lor C)$. Let $F_\Phi$ denote the resulting CNF.

Reduction to weighted partial MaxSAT problem

Let $F^H = F_\Phi$, and for each $l_i \in Lbls(\Phi)$, create a soft clause $C_i = (\neg p_i)$ with weight $w(l_i)$.

Compute a MaxSAT solution $\tau$ of $F^H \cup C_1 \cup \ldots C_n$. $\tau$ is a MaxSAT solution of $\Phi$. 

Note: this also gives a nice incremental-SAT based MaxSAT algorithm.
LCNF formulas: computing MaxSAT solutions

Given a weighted LCNF $\Phi$, for each label $l_i \in Lbls(\Phi)$, $1 \leq i \leq n$, introduce a fresh variable $p_i$.

Then, convert each labelled clause $C\{l_1,\ldots,l_k\}$ in $\Phi$ into a clause $(p_1 \lor \cdots \lor p_k \lor C)$. Let $F_\Phi$ denote the resulting CNF.

Reduction to weighted partial MaxSAT problem

Let $F^H = F_\Phi$, and for each $l_i \in Lbls(\Phi)$, create a soft clause $C_i = (\neg p_i)$ with weight $w(l_i)$.

Compute a MaxSAT solution $\tau$ of $F^H \cup C_1 \cup \ldots C_n$. $\tau$ is a MaxSAT solution of $\Phi$.

Direct computation

Load $F_\Phi$ into an incremental SAT solver, and use $p_i$’s as assumptions, and run a modified MaxSAT algorithm (need to “relax” $p_i$’s, not clauses).

Note: this also gives a nice incremental-SAT based MaxSAT algorithm.
Experimental evaluation: group-MUS

- 99 group-CNF instances from Intel (PBA), 148 instances derived from HWMCC.
- Time limit 1800 sec, memory limit 4 GB.
Experimental evaluation: MaxSAT

- 1000 instances from MaxSAT Competition 2013.
- Time limit 1800 sec, memory limit 4 GB.

<table>
<thead>
<tr>
<th></th>
<th>MaxSAT</th>
<th>Partial MaxSAT</th>
<th>Weighted Partial MaxSAT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Sol.</td>
<td>A.CPU</td>
<td>#Sol.</td>
</tr>
<tr>
<td>Instances</td>
<td>55</td>
<td>627</td>
<td>397</td>
</tr>
<tr>
<td>P_NI</td>
<td>37</td>
<td>172.76</td>
<td>254</td>
</tr>
<tr>
<td>P_NI+BCE</td>
<td>41</td>
<td>237.58</td>
<td>241</td>
</tr>
<tr>
<td>P_NI+BCE+RS</td>
<td>35</td>
<td>177.37</td>
<td>240</td>
</tr>
<tr>
<td>P_NI+RS</td>
<td>37</td>
<td>246.68</td>
<td>265</td>
</tr>
<tr>
<td>P</td>
<td>37</td>
<td>236.26</td>
<td>237</td>
</tr>
<tr>
<td>P+BCE</td>
<td>38</td>
<td>180.22</td>
<td>227</td>
</tr>
<tr>
<td>P+BCE+RS</td>
<td>37</td>
<td>209.48</td>
<td>221</td>
</tr>
<tr>
<td>P+RS</td>
<td>34</td>
<td>151.77</td>
<td>238</td>
</tr>
<tr>
<td>L</td>
<td>36</td>
<td>101.92</td>
<td>270</td>
</tr>
<tr>
<td>L+BCE</td>
<td>37</td>
<td>67.88</td>
<td>271</td>
</tr>
<tr>
<td>L+BCE+RS</td>
<td>38</td>
<td>96.02</td>
<td>271</td>
</tr>
<tr>
<td>L+RS</td>
<td>38</td>
<td>161.71</td>
<td>276</td>
</tr>
<tr>
<td>WMSU1</td>
<td>39</td>
<td>165.64</td>
<td>241</td>
</tr>
</tbody>
</table>
To sum up ...

Direct preprocessing

- plain-MUS: any clause elimination and BCP.
- group-MUS: only MUS-preserving (sp. monotone) clause elimination.
- MaxSAT: only MUS-preserving (sp. monotone) clause elimination.
To sum up ...

Direct preprocessing

- plain-MUS: any clause elimination and BCP.
- group-MUS: only MUS-preserving (sp. monotone) clause elimination.
- MaxSAT: only MUS-preserving (sp. monotone) clause elimination.

Labelled CNF (LCNF) framework

Allows for sound reconstruction of (group-)MUSes and MaxSAT solutions after preprocessing.
To sum up ...

Direct preprocessing

▶ plain-MUS: any clause elimination and BCP.
▶ group-MUS: only MUS-preserving (sp. monotone) clause elimination.
▶ MaxSAT: only MUS-preserving (sp. monotone) clause elimination.

Labelled CNF (LCNF) framework

Allows for sound reconstruction of (group-)MUSes and MaxSAT solutions after preprocessing.

Experimental evaluation

Good results on industrially-relevant group-MUS computation problems

Improvements on weighted partial MaxSAT problems
Current and Future Work

- Analyze additional preprocessing techniques.
- Relax the conditions for labelled subsumption elimination.
- Proper implementation and additional MaxSAT algorithms.
Current and Future Work

- Analyze additional preprocessing techniques.

- Relax the conditions for labelled subsumption elimination.

- Proper implementation and additional MaxSAT algorithms.

Thank you for your attention!
Related Papers

