Quantified Maximum Satisfiability: A Core-Guided Approach

Alexey Ignatiev[1], Mikoláš Janota[1], and Joao Marques-Silva[1,2]

[1] INESC-ID/IST, Lisbon, Portugal
[2] CASL/CSI, University College Dublin, Ireland

Sixteenth International Conference on
Theory and Applications of Satisfiability Testing

Helsinki, Finland
July 11, 2013

Motivation

Many practical *decision* problems can be represented as QBF.

Motivation

Many practical *decision* problems can be represented as QBF.

Smallest MUS problem

Find a **smallest** unsatisfiable subformula of a CNF formula.

Decision version (for φ and k) is Σ_2^P-complete.

Motivation

Many practical *decision* problems can be represented as QBF.

Smallest MUS problem

Find a **smallest** unsatisfiable subformula of a CNF formula.

Decision version (for φ and k) is Σ_2^P-complete.

Quantified MaxSAT (QMaxSAT)

Find a solution of a QBF that has a **minimal** cost.

Motivation

Many practical *decision* problems can be represented as QBF.

Smallest MUS problem

Find a **smallest** unsatisfiable subformula of a CNF formula.

Decision version (for φ and k) is Σ_2^P-complete.

Quantified MaxSAT (QMaxSAT)

Find a solution of a QBF that has a **minimal** cost.

Applications — optimization problems with quantified constraints.

Quantified Boolean Formula

QBF — a quantified generalization of SAT:

Quantified Boolean Formula

QBF — a quantified generalization of SAT:

- $Q_1 X_1 \ldots Q_k X_k.\ \varphi$, where $Q_i \in \{\exists, \forall\}$

Quantified Boolean Formula

QBF — a quantified generalization of SAT:

- $Q_1 X_1 \ldots Q_k X_k.\ \varphi$, where $Q_i \in \{\exists, \forall\}$
- $\overrightarrow{Q}.\ \varphi$, where $\overrightarrow{Q} = (Q_1 X_1 \ldots Q_k X_k)$ `# short form`

Quantified Boolean Formula

QBF — a quantified generalization of SAT:

- $Q_1 X_1 \ldots Q_k X_k.\ \varphi$, where $Q_i \in \{\exists, \forall\}$
- $\overrightarrow{Q}.\ \varphi$, where $\overrightarrow{Q} = (Q_1 X_1 \ldots Q_k X_k)$ `# short form`

Example

$$\xi = \exists e_1, e_2\, \forall x_1, x_2.\ (\neg e_1 \wedge \neg e_2) \rightarrow (x_1 \vee x_2)$$

Quantified MaxSAT

$$\psi = \exists E\, \overrightarrow{Q}.\ \varphi$$

Definition

Assignment $\mathcal{A}_E$ is a ***solution*** of ψ iff $\overrightarrow{Q}.\ \varphi\big|_{\mathcal{A}_E}$ is true.

$\mathcal{M}(\psi)$ — set of all solutions of ψ.

Quantified MaxSAT

$$\psi = \exists E\, \overrightarrow{Q}.\ \varphi$$

Definition

Assignment $\mathcal{A}_E$ is a ***solution*** of ψ iff $\overrightarrow{Q}.\ \varphi\big|_{\mathcal{A}_E}$ is true.

$\mathcal{M}(\psi)$ — set of all solutions of ψ.

Example

$$\xi = \exists e_1, e_2\, \forall x_1, x_2.\ (\neg e_1 \wedge \neg e_2) \rightarrow (x_1 \vee x_2)$$
$$\mathcal{M}(\xi) = \{(0,1), (1,0), (1,1)\}$$

Quantified MaxSAT

$$\psi = \exists E\ \overrightarrow{Q}.\ \varphi$$

Definition

Assignment $\mathcal{A}_E$ is a ***solution*** of ψ iff $\overrightarrow{Q}.\ \varphi\big|_{\mathcal{A}_E}$ is true.

$\mathcal{M}(\psi)$ — set of all solutions of ψ.

Example

$$\xi = \exists e_1, e_2\ \forall x_1, x_2.\ (\neg e_1 \land \neg e_2) \rightarrow (x_1 \lor x_2)$$
$$\mathcal{M}(\xi) = \{(0,1), (1,0), (1,1)\}$$

What solution is the best?

Quantified MaxSAT

$$\psi = \exists E\, \overrightarrow{Q}.\ \varphi$$

Definition

Assignment $\mathcal{A}_E$ is a ***solution*** of ψ iff $\overrightarrow{Q}.\ \varphi\big|_{\mathcal{A}_E}$ is true.

$\mathcal{M}(\psi)$ — set of all solutions of ψ.

Example

$$\xi = \exists e_1, e_2\, \forall x_1, x_2.\ (\neg e_1 \wedge \neg e_2) \rightarrow (x_1 \vee x_2)$$
$$\mathcal{M}(\xi) = \{(0,1), (1,0), (1,1)\}$$

What solution is the best?

QMaxSAT

Consider a *cost function* $f(e_1, \ldots, e_l) = \sum_{i=1}^{l} a_i \cdot e_i$, $|E| = l$.
Find $\mathcal{A}_E \in \mathcal{M}(\psi)$ s. t. $\forall \mathcal{B}_E \in \mathcal{M}(\psi)$: $f(\mathcal{A}_E) \leqslant f(\mathcal{B}_E)$.

Quantified MaxSAT

$$\psi = \exists E \, \overrightarrow{Q}. \; \varphi$$

Definition

Assignment $\mathcal{A}_E$ is a ***solution*** of ψ iff $\overrightarrow{Q}. \; \varphi\big|_{\mathcal{A}_E}$ is true.

$\mathcal{M}(\psi)$ — set of all solutions of ψ.

Example

$$\xi = \exists e_1, e_2 \, \forall x_1, x_2. \; (\neg e_1 \wedge \neg e_2) \rightarrow (x_1 \vee x_2)$$
$$\mathcal{M}(\xi) = \{(0,1), (1,0), (1,1)\}$$

What solution is the best?

$$f(e_1, e_2) = 2 \cdot e_1 + 3 \cdot e_2$$

QMaxSAT

Consider a *cost function* $f(e_1, \ldots, e_l) = \sum_{i=1}^{l} a_i \cdot e_i$, $|E| = l$.
Find $\mathcal{A}_E \in \mathcal{M}(\psi)$ s. t. $\forall \mathcal{B}_E \in \mathcal{M}(\psi)$: $f(\mathcal{A}_E) \leqslant f(\mathcal{B}_E)$.

Quantified MaxSAT

$$\psi = \exists E\, \overrightarrow{Q}.\ \varphi$$

Definition

Assignment $\mathcal{A}_E$ is a ***solution*** of ψ iff $\overrightarrow{Q}.\ \varphi\big|_{\mathcal{A}_E}$ is true.

$\mathcal{M}(\psi)$ — set of all solutions of ψ.

Example

$$\xi = \exists e_1, e_2\, \forall x_1, x_2.\ (\neg e_1 \wedge \neg e_2) \to (x_1 \vee x_2)$$
$$\mathcal{M}(\xi) = \{(0, 1), (1, 0), (1, 1)\}$$

What solution is the best?

$f(e_1, e_2) = 2 \cdot e_1 + 3 \cdot e_2$ $\qquad f(1, 0) = \min_{\mathcal{M}(\xi)} f(e_1, e_2) = 2$

QMaxSAT

Consider a *cost function* $f(e_1, \ldots, e_l) = \sum_{i=1}^{l} a_i \cdot e_i, |E| = l$.
Find $\mathcal{A}_E \in \mathcal{M}(\psi)$ s. t. $\forall \mathcal{B}_E \in \mathcal{M}(\psi)$: $f(\mathcal{A}_E) \leqslant f(\mathcal{B}_E)$.

Approaches

$$\psi = \exists E\ \overrightarrow{Q}.\ \varphi$$

Approaches

$$\psi = \exists E\ \overrightarrow{Q}.\ \varphi$$

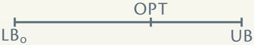
- **Linear search:** $\exists E\ \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$

Approaches

$$\psi = \exists E\ \overrightarrow{Q}.\ \varphi$$

- **Linear search:** $\exists E\ \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$

Refine LB

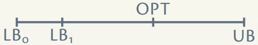


Approaches

$$\psi = \exists E\ \overrightarrow{Q}.\ \varphi$$

- **Linear search:** $\exists E\ \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$

Refine LB

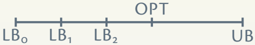


Approaches

$$\psi = \exists E\ \overrightarrow{Q}.\ \varphi$$

- **Linear search:** $\exists E\ \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$

Refine LB

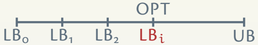


Approaches

$$\psi = \exists E\ \overrightarrow{Q}.\ \varphi$$

- **Linear search:** $\exists E\ \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$

Refine LB

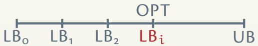


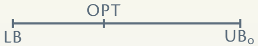
Approaches

$$\psi = \exists E\ \overrightarrow{Q}.\ \varphi$$

- **Linear search:** $\exists E\ \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$

Refine LB





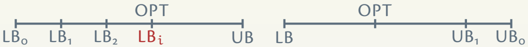
Refine UB

Approaches

$$\psi = \exists E\ \overrightarrow{Q}.\ \varphi$$

- **Linear search:** $\exists E\ \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$

Refine LB



Refine UB

Approaches

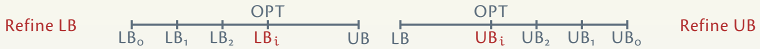
$$\psi = \exists E \, \overrightarrow{Q}. \; \varphi$$

- **Linear search:** $\exists E \, \overrightarrow{Q}. \; \varphi \land (f(e_1, \ldots, e_l) \leqslant k)$

Approaches

$$\psi = \exists E\ \overrightarrow{Q}.\ \varphi$$

- **Linear search:** $\exists E\ \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$

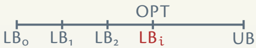


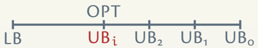
Approaches

$$\psi = \exists E\ \overrightarrow{Q}.\ \varphi$$

- Linear search: $\exists E\ \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$

Refine LB





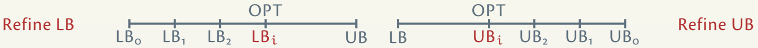
Refine UB

- Core-guided search:

Approaches

$$\psi = \exists E\, \overrightarrow{Q}.\ \varphi$$

- Linear search: $\exists E\, \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$

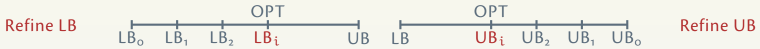


- Core-guided search:
  - $f(e_1, \ldots, e_l) = \sum_{i=1}^{l} e_l$ `# unweighted QMaxSAT`

Approaches

$$\psi = \exists E\ \overrightarrow{Q}.\ \varphi$$

- **Linear search:** $\exists E\ \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$



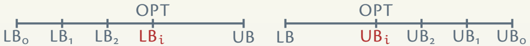
- **Core-guided search:**
  - $f(e_1, \ldots, e_l) = \sum_{i=1}^{l} e_l$ `# unweighted QMaxSAT`
  - $\varphi_S = \{\neg e_1, \ldots, \neg e_l\}$ `# soft clauses`

Approaches

$$\psi = \exists E\, \overrightarrow{Q}.\ \varphi$$

- Linear search: $\exists E\, \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$

Refine LB



Refine UB

- Core-guided search:
  - $f(e_1, \ldots, e_l) = \sum_{i=1}^{l} e_l$ `# unweighted QMaxSAT`
  - $\varphi_S = \{\neg e_1, \ldots, \neg e_l\}$ `# soft clauses`
  - $\#(\varphi_S, \mathcal{A}_E) = \sum_{c \in \varphi_S} c\big|_{\mathcal{A}_E}$ `# number of satisfied soft clauses`

Approaches

$$\psi = \exists E\, \overrightarrow{Q}.\ \varphi$$

- Linear search: $\exists E\, \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$

- Core-guided search:
  - $f(e_1, \ldots, e_l) = \sum_{i=1}^{l} e_l$ `# unweighted QMaxSAT`
  - $\varphi_S = \{\neg e_1, \ldots, \neg e_l\}$ `# soft clauses`
  - $\#(\varphi_S, \mathcal{A}_E) = \sum_{c \in \varphi_S} c\big|_{\mathcal{A}_E}$ `# number of satisfied soft clauses`
  - $\forall \mathcal{A}_E$:

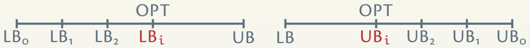
$$f(\mathcal{A}_E) = y \ \Leftrightarrow\ \#(\varphi_S, \mathcal{A}_E) = l - y$$

Approaches

$$\psi = \exists E\ \overrightarrow{Q}.\ \varphi$$

- Linear search: $\exists E\ \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$

Refine LB



Refine UB

- Core-guided search:
  - $f(e_1, \ldots, e_l) = \sum_{i=1}^{l} e_l$ `# unweighted QMaxSAT`
  - $\varphi_S = \{\neg e_1, \ldots, \neg e_l\}$ `# soft clauses`
  - $\#(\varphi_S, \mathcal{A}_E) = \sum_{c \in \varphi_S} c\big|_{\mathcal{A}_E}$ `# number of satisfied soft clauses`
  - $\forall \mathcal{A}_E$:

$$f(\mathcal{A}_E) = y \ \Leftrightarrow\ \#(\varphi_S, \mathcal{A}_E) = l - y$$

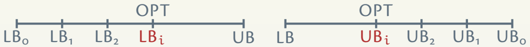
  - $\psi' = \exists E\ \overrightarrow{Q}.\ \varphi \wedge \varphi_S$ `# QBF to decide iteratively`

Approaches

$$\psi = \exists E\, \overrightarrow{Q}.\ \varphi$$

- Linear search: $\exists E\, \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$

Refine LB



Refine UB

- Core-guided search:
  - $f(e_1, \ldots, e_l) = \sum_{i=1}^{l} e_l$ `# unweighted QMaxSAT`
  - $\varphi_S = \{\neg e_1, \ldots, \neg e_l\}$ `# soft clauses`
  - $\#(\varphi_S, \mathcal{A}_E) = \sum_{c \in \varphi_S} c\big|_{\mathcal{A}_E}$ `# number of satisfied soft clauses`
  - $\forall \mathcal{A}_E$:

    $$f(\mathcal{A}_E) = y \ \Leftrightarrow\ \#(\varphi_S, \mathcal{A}_E) = l - y$$

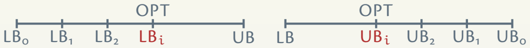
  - $\psi' = \exists E\, \overrightarrow{Q}.\ \varphi \wedge \varphi_S$ `# QBF to decide iteratively`
  - **find** $\mathcal{A}_E \in \mathcal{M}(\psi)$ that maximizes $\#(\varphi_S, \mathcal{A}_E)$

Approaches

$$\psi = \exists E\ \overrightarrow{Q}.\ \varphi$$

- **Linear search:** $\exists E\ \overrightarrow{Q}.\ \varphi \wedge (f(e_1, \ldots, e_l) \leqslant k)$

Refine LB



Refine UB

- Core-guided search:
  - $f(e_1, \ldots, e_l) = \sum_{i=1}^{l} e_l$ `# unweighted QMaxSAT`
  - $\varphi_S = \{\neg e_1, \ldots, \neg e_l\}$ `# soft clauses`
  - $\#(\varphi_S, \mathcal{A}_E) = \sum_{c \in \varphi_S} c\big|_{\mathcal{A}_E}$ `# number of satisfied soft clauses`
  - $\forall \mathcal{A}_E$:

$$f(\mathcal{A}_E) = y \quad \Leftrightarrow \quad \#(\varphi_S, \mathcal{A}_E) = l - y$$

  - $\psi' = \exists E\ \overrightarrow{Q}.\ \varphi \wedge \varphi_S$ `# QBF to decide iteratively`
  - **find** $\mathcal{A}_E \in \mathcal{M}(\psi)$ that maximizes $\#(\varphi_S, \mathcal{A}_E)$

Definition

Formula $\varphi_C = \varphi \wedge \varphi'_S$, $\varphi'_S \subseteq \varphi_S$, is an ***unsatisfiable core*** of ψ' if $\exists E\ \overrightarrow{Q}.\ \varphi_C$ is false.

Algorithm QMSU1

- Based on the **Fu&Malik**'s algorithm (a.k.a. MSU1 or WPM1)

Algorithm QMSU1

- Based on the **Fu&Malik**'s algorithm (a.k.a. MSU1 or WPM1)

  Differences:

  - **QBF** oracle instead of SAT oracle
  - hard part ***can be*** in non-CNF

Algorithm QMSU1

- Based on the **Fu&Malik**'s algorithm (a.k.a. MSU1 or WPM1)

  Differences:

  - **QBF** oracle instead of SAT oracle
  - hard part ***can be*** in non-CNF

- Basic principles:

$$\psi'_R \leftarrow \psi' = \exists E\ \overrightarrow{Q}.\ \varphi \wedge \varphi_S$$

Algorithm QMSU1

- Based on the **Fu&Malik**'s algorithm (a.k.a. MSU1 or WPM1)

  Differences:

  - **QBF** oracle instead of SAT oracle
  - hard part ***can be*** in non-CNF

- Basic principles:

$\psi'_R \leftarrow \psi' = \exists E\ \overrightarrow{Q}.\ \varphi \wedge \varphi_S$

while ψ'_R *is* false: # ask a QBF oracle

Algorithm QMSU1

- Based on the **Fu&Malik**'s algorithm (a.k.a. MSU1 or WPM1)

  Differences:

  - **QBF** oracle instead of SAT oracle
  - hard part ***can be*** in non-CNF

- Basic principles:

$\psi'_R \leftarrow \psi' = \exists E\ \overrightarrow{Q}.\ \varphi \wedge \varphi_S$

while ψ'_R *is* false: # ask a QBF oracle

    extract unsatisfiable core φ_C

Algorithm QMSU1

- Based on the **Fu&Malik**'s algorithm (a.k.a. MSU1 or WPM1)

  Differences:

  - **QBF** oracle instead of SAT oracle
  - hard part ***can be*** in non-CNF

- Basic principles:

$\psi'_R \leftarrow \psi' = \exists E\ \overrightarrow{Q}.\ \varphi \wedge \varphi_S$
while ψ'_R *is* false: # ask a QBF oracle
    extract unsatisfiable core φ_C
    relax soft part of φ_C

Algorithm QMSU1

- Based on the **Fu&Malik**'s algorithm (a.k.a. MSU1 or WPM1)

  Differences:

  - **QBF** oracle instead of SAT oracle
  - hard part ***can be*** in non-CNF

- Basic principles:

$\psi'_R \leftarrow \psi' = \exists E\ \overrightarrow{Q}.\ \varphi \wedge \varphi_S$
while ψ'_R *is* false**:** # ask a QBF oracle
    extract unsatisfiable core φ_C
    relax soft part of φ_C
    update ψ'_R

Algorithm QMSU1

input : $\psi = \exists E\ \overrightarrow{Q}.\ \varphi$, and φ_S

$R_{all} \leftarrow \emptyset$ # set of all relaxation variables
while true:
  $\psi'_R = \exists E\ \exists R_{all}\ \overrightarrow{Q}.\ \varphi \wedge \varphi_S$
  $(\mathrm{st}, \varphi_C, \mathcal{A}_E) \leftarrow \mathtt{QBF}(\psi'_R)$ # calling a QBF oracle

  if st $=$ true:
    return $\mathcal{A}_E$

  $R \leftarrow \emptyset$
  foreach $c \in \mathtt{Soft}(\varphi_C)$: # relaxing the core
    let r be a new relaxation variable
    $R \leftarrow R \cup \{r\}$
    $\varphi_S \leftarrow \varphi_S \setminus \{c\} \cup \{c \vee r\}$

  $\varphi \leftarrow \varphi \wedge \mathtt{CNF}(\sum_{r \in R} r \leqslant 1)$ # updating the hard part
  $R_{all} \leftarrow R_{all} \cup R$

Algorithm QMSU1

input : $\psi = \exists E\, \overrightarrow{Q}.\ \varphi$, and φ_S

$R_{\mathtt{all}} \leftarrow \emptyset$ `# set of all relaxation variables`
while true:
  $\psi'_R = \exists E\, \exists R_{\mathtt{all}}\, \overrightarrow{Q}.\ \varphi \wedge \varphi_S$
  $(\mathrm{st}, \varphi_C, \mathcal{A}_E) \leftarrow \mathtt{QBF}(\psi'_R)$ `# calling a QBF oracle`

  if st $=$ true:
    return $\mathcal{A}_E$

  $R \leftarrow \emptyset$
  foreach $c \in \mathtt{Soft}(\varphi_C)$: `# relaxing the core`
    let r be a new relaxation variable
    $R \leftarrow R \cup \{r\}$
    $\varphi_S \leftarrow \varphi_S \setminus \{c\} \cup \{c \vee r\}$

  $\varphi \leftarrow \varphi \wedge \mathtt{CNF}(\sum_{r \in R} r \leqslant 1)$ `# updating the hard part`
  $R_{\mathtt{all}} \leftarrow R_{\mathtt{all}} \cup R$

Algorithm QMSU1

input : $\psi = \exists E\ \overrightarrow{Q}.\ \varphi$, and φ_S

$R_{all} \leftarrow \emptyset$ # set of all relaxation variables
while true:
  $\psi'_R = \exists E\ \exists R_{all}\ \overrightarrow{Q}.\ \varphi \wedge \varphi_S$
  $(st, \varphi_C, \mathcal{A}_E) \leftarrow \mathtt{QBF}(\psi'_R)$ # calling a QBF oracle

  if $st =$ true:
    return $\mathcal{A}_E$

  $R \leftarrow \emptyset$
  foreach $c \in \mathtt{Soft}(\varphi_C)$: # relaxing the core
    let r be a new relaxation variable
    $R \leftarrow R \cup \{r\}$
    $\varphi_S \leftarrow \varphi_S \setminus \{c\} \cup \{c \vee r\}$

  $\varphi \leftarrow \varphi \wedge \mathtt{CNF}(\sum_{r \in R} r \leqslant 1)$ # updating the hard part
  $R_{all} \leftarrow R_{all} \cup R$

CEGAR-based 2QBF

$$\exists X \forall Y.\ \varphi_H \wedge \varphi_S$$

CEGAR-based 2QBF

$$\exists X \forall Y.\ \varphi_H \wedge \varphi_S$$

$$\Updownarrow$$

Full expansion: $$\exists X.\ \bigwedge_{\nu \in \{0,1\}^{|Y|}} \left(\varphi_H \wedge \varphi_S\right)\Big|_{\nu}$$

CEGAR-based 2QBF

$$\exists X \forall Y.\ \varphi_H \wedge \varphi_S$$

$$\Updownarrow$$

Full expansion: $\exists X.\ \bigwedge_{\nu \in \{0,1\}^{|Y|}} (\varphi_H \wedge \varphi_S)\big|_\nu$

Partial expansion: $\exists X.\ \bigwedge_{\nu \in W} (\varphi_H \wedge \varphi_S)\big|_\nu \qquad W \subseteq \{0,1\}^{|Y|}$

CEGAR-based 2QBF

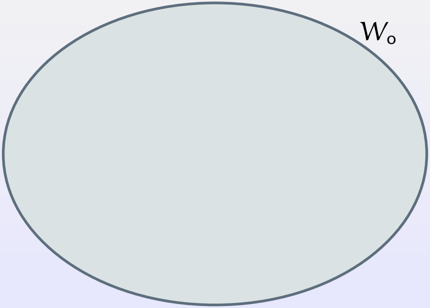
$$\exists X \forall Y.\ \varphi_H \wedge \varphi_S$$

$$\Updownarrow$$

Full expansion: $\exists X.\ \bigwedge_{\nu \in \{0,1\}^{|Y|}} (\varphi_H \wedge \varphi_S)\big|_{\nu}$

Partial expansion: $\exists X.\ \bigwedge_{\nu \in W} (\varphi_H \wedge \varphi_S)\big|_{\nu} \qquad W \subseteq \{0,1\}^{|Y|}$

Gradual strengthening of abstractions until a solution is found



CEGAR-based 2QBF

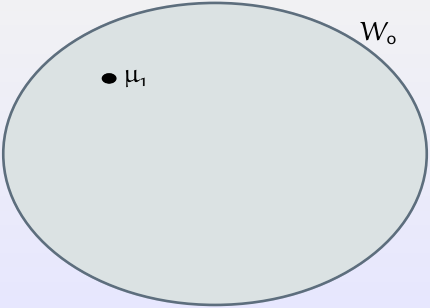
$$\exists X \forall Y.\ \varphi_H \wedge \varphi_S$$

$$\Updownarrow$$

Full expansion: $\exists X.\ \bigwedge_{\nu \in \{0,1\}^{|Y|}} (\varphi_H \wedge \varphi_S)\big|_\nu$

Partial expansion: $\exists X.\ \bigwedge_{\nu \in W} (\varphi_H \wedge \varphi_S)\big|_\nu \qquad W \subseteq \{0,1\}^{|Y|}$

Gradual strengthening of abstractions until a solution is found



CEGAR-based 2QBF

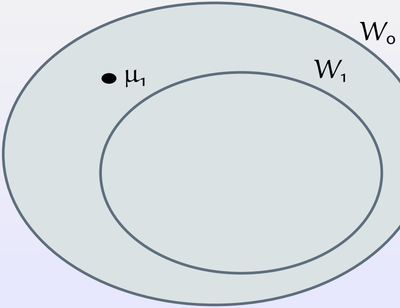
$$\exists X \forall Y.\ \varphi_H \wedge \varphi_S$$

$$\Updownarrow$$

Full expansion: $\exists X.\ \bigwedge_{\nu \in \{0,1\}^{|Y|}} (\varphi_H \wedge \varphi_S)\big|_{\nu}$

Partial expansion: $\exists X.\ \bigwedge_{\nu \in W} (\varphi_H \wedge \varphi_S)\big|_{\nu} \qquad W \subseteq \{0,1\}^{|Y|}$

Gradual strengthening of abstractions until a solution is found



CEGAR-based 2QBF

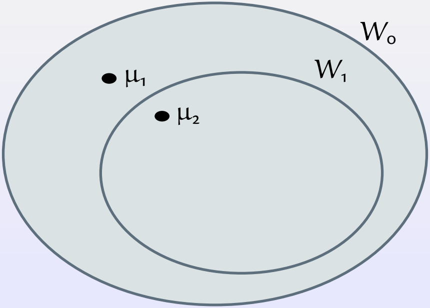
$$\exists X \forall Y.\ \varphi_H \wedge \varphi_S$$

$$\Updownarrow$$

Full expansion: $\exists X.\ \bigwedge_{\nu \in \{0,1\}^{|Y|}} (\varphi_H \wedge \varphi_S)\big|_\nu$

Partial expansion: $\exists X.\ \bigwedge_{\nu \in W} (\varphi_H \wedge \varphi_S)\big|_\nu \qquad W \subseteq \{0,1\}^{|Y|}$

Gradual strengthening of abstractions until a solution is found



CEGAR-based 2QBF

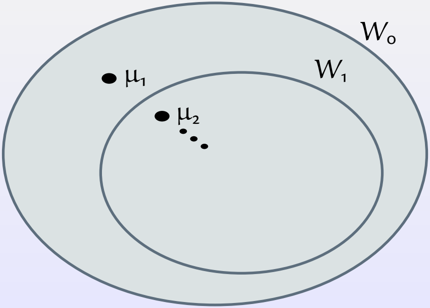
$$\exists X \forall Y.\ \varphi_H \wedge \varphi_S$$

$$\Updownarrow$$

Full expansion: $\exists X.\ \bigwedge_{\nu \in \{0,1\}^{|Y|}} (\varphi_H \wedge \varphi_S)\big|_\nu$

Partial expansion: $\exists X.\ \bigwedge_{\nu \in W} (\varphi_H \wedge \varphi_S)\big|_\nu \qquad W \subseteq \{0,1\}^{|Y|}$

Gradual strengthening of abstractions until a solution is found



CEGAR-based 2QBF

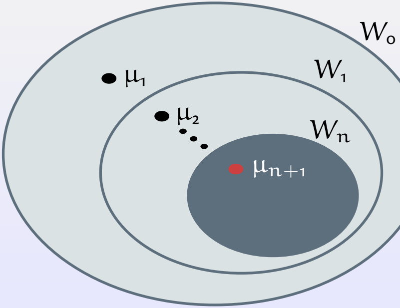
$$\exists X \forall Y.\ \varphi_H \wedge \varphi_S$$

$$\Updownarrow$$

Full expansion: $\exists X.\ \bigwedge_{\nu \in \{0,1\}^{|Y|}} (\varphi_H \wedge \varphi_S)\big|_\nu$

Partial expansion: $\exists X.\ \bigwedge_{\nu \in W} (\varphi_H \wedge \varphi_S)\big|_\nu \qquad W \subseteq \{0,1\}^{|Y|}$

Gradual strengthening of abstractions until a solution is found



Computing Cores in CEGAR-based 2QBF

input : $\exists X \forall Y.\ \varphi_H \wedge \varphi_S$

$\omega \leftarrow \emptyset$

while true:

$\quad \varphi \leftarrow \mathtt{CNF}(\bigwedge_{\nu \in \omega} \varphi_H|_\nu) \cup \bigwedge_{\nu \in \omega} \varphi_S|_\nu$

$\quad (st_1, \mu, \varphi_C) \leftarrow \mathtt{SAT}(\varphi)$ `# candidate`

$\quad$ **if** $st_1 =$ false:

$\quad\quad \varphi'_S \leftarrow \{c \in \varphi_S \mid c' \in \varphi_C, \nu \in \omega, c' = c|_\nu\}$

$\quad\quad$ **return** (false, $\varphi_H \wedge \varphi'_S$) `# no candidate found`

$\quad (st_2, \nu) \leftarrow \mathtt{SAT}\left(\neg(\varphi_H \wedge \varphi_S)|_\mu\right)$ `# counterexample`

$\quad$ **if** $st_2 =$ false:

$\quad\quad$ **return** (true, μ) `# solution found`

$\quad \omega \leftarrow \omega \cup \{\nu\}$

Computing Cores in CEGAR-based 2QBF

input : $\exists X \forall Y.\ \varphi_H \wedge \varphi_S$

$\omega \leftarrow \emptyset$

while true:

  $\varphi \leftarrow \mathtt{CNF}(\bigwedge_{\nu \in \omega} \varphi_H|_\nu) \cup \bigwedge_{\nu \in \omega} \varphi_S|_\nu$

  $(st_1, \mu, \varphi_C) \leftarrow \mathtt{SAT}(\varphi)$ `# candidate`

  if $st_1 =$ false:

    $\varphi'_S \leftarrow \{c \in \varphi_S \mid c' \in \varphi_C, \nu \in \omega, c' = c|_\nu\}$

    return (false, $\varphi_H \wedge \varphi'_S$) `# no candidate found`

  $(st_2, \nu) \leftarrow \mathtt{SAT}\left(\neg(\varphi_H \wedge \varphi_S)|_\mu\right)$ `# counterexample`

  if $st_2 =$ false:

    return (true, μ) `# solution found`

  $\omega \leftarrow \omega \cup \{\nu\}$

Computing Cores in CEGAR-based 2QBF

input : $\exists X \forall Y.\ \varphi_H \wedge \varphi_S$

$\omega \leftarrow \emptyset$
while true:
  $\varphi \leftarrow \mathtt{CNF}(\bigwedge_{\nu \in \omega} \varphi_H|_\nu) \cup \bigwedge_{\nu \in \omega} \varphi_S|_\nu$
  $(st_1, \mu, \varphi_C) \leftarrow \mathtt{SAT}(\varphi)$ # candidate
  if $st_1 =$ false:
    $\varphi'_S \leftarrow \{c \in \varphi_S \mid c' \in \varphi_C, \nu \in \omega, c' = c|_\nu\}$
    return (false, $\varphi_H \wedge \varphi'_S$) # no candidate found
  $(st_2, \nu) \leftarrow \mathtt{SAT}\left(\neg(\varphi_H \wedge \varphi_S)|_\mu\right)$ # counterexample
  if $st_2 =$ false:
    return (true, μ) # solution found
  $\omega \leftarrow \omega \cup \{\nu\}$

Smallest MUS Problem

Definition

Formula ψ^*, $\psi^* \subseteq \varphi$, is called a ***smallest MUS*** of φ if

1. ψ^* is unsatisfiable
2. for any MUS ψ, $\psi \subseteq \varphi$, the following holds $|\psi^*| \leqslant |\psi|$

Example

$$\varphi = \{\ x_2 \vee \neg x_3 \vee \neg x_4,\ \ x_1 \vee x_2,\ \ x_3,\ \ \neg x_1,\ \ x_4,\ \ \neg x_2\ \}$$

φ has 2 MUSes:

Smallest MUS Problem

Definition

Formula ψ^*, $\psi^* \subseteq \varphi$, is called a ***smallest MUS*** of φ if

1. ψ^* is unsatisfiable
2. for any MUS ψ, $\psi \subseteq \varphi$, the following holds $|\psi^*| \leqslant |\psi|$

Example

$$\varphi = \{\ x_2 \vee \neg x_3 \vee \neg x_4,\ \ x_1 \vee x_2,\ \ x_3,\ \ \neg x_1,\ \ x_4,\ \ \neg x_2\ \}$$

φ has 2 MUSes:

- $|\psi_1| = 4$

Smallest MUS Problem

Definition

Formula ψ^*, $\psi^* \subseteq \varphi$, is called a ***smallest MUS*** of φ if

1. ψ^* is unsatisfiable
2. for any MUS ψ, $\psi \subseteq \varphi$, the following holds $|\psi^*| \leqslant |\psi|$

Example

$\varphi = \{\ x_2 \lor \neg x_3 \lor \neg x_4,\ \ x_1 \lor x_2,\ \ x_3,\ \ \neg x_1,\ \ x_4,\ \ \neg x_2\ \}$

φ has 2 MUSes:

- $|\psi_1| = 4$
- $|\psi_2| = 3$

Smallest MUS Problem

Definition

Formula ψ^*, $\psi^* \subseteq \varphi$, is called a ***smallest MUS*** of φ if

1. ψ^* is unsatisfiable
2. for any MUS ψ, $\psi \subseteq \varphi$, the following holds $|\psi^*| \leqslant |\psi|$

Example

$$\varphi = \{\ x_2 \vee \neg x_3 \vee \neg x_4,\ \ x_1 \vee x_2,\ \ x_3,\ \ \neg x_1,\ \ x_4,\ \ \neg x_2\ \}$$

φ has 2 MUSes:

- $|\psi_1| = 4$
- $|\psi_2| = 3\ < |\psi_1| \Rightarrow \psi_2 = \psi^*$

SMUS as QMaxSAT

- Original formula $\varphi = \{c_1, \ldots, c_m\}$

SMUS as QMaxSAT

- Original formula $\varphi = \{c_1, \ldots, c_m\}$
- QBF formulation of SMUS:

SMUS as QMaxSAT

- Original formula $\varphi = \{c_1, \ldots, c_m\}$
- QBF formulation of SMUS:
  - $S = \{s_1, \ldots, s_m\}$ `# selection variables`
  - $\varphi_R = \{c_1 \vee \neg s_1, \ldots, c_m \vee \neg s_m\}$ `# extended formula`

SMUS as QMaxSAT

- Original formula $\varphi = \{c_1, \ldots, c_m\}$

- QBF formulation of SMUS:
  - $S = \{s_1, \ldots, s_m\}$ `# selection variables`
  - $\varphi_R = \{c_1 \vee \neg s_1, \ldots, c_m \vee \neg s_m\}$ `# extended formula`
  - $\varphi_{unsat} = \exists S\, \forall X.\ \neg \varphi_R$ `# QBF for finding UNSAT subformula`

SMUS as QMaxSAT

- Original formula $\varphi = \{c_1, \ldots, c_m\}$
- QBF formulation of SMUS:
  - $S = \{s_1, \ldots, s_m\}$ # selection variables
  - $\varphi_R = \{c_1 \vee \neg s_1, \ldots, c_m \vee \neg s_m\}$ # extended formula
  - $\varphi_{unsat} = \exists S \, \forall X. \; \neg \varphi_R$ # QBF for finding UNSAT subformula
  - $f(s_1, \ldots, s_m) = \sum_{i=1}^{m} s_i$ # objective function

SMUS as QMaxSAT

- Original formula $\varphi = \{c_1, \ldots, c_m\}$

- QBF formulation of SMUS:
  - $S = \{s_1, \ldots, s_m\}$ # selection variables
  - $\varphi_R = \{c_1 \vee \neg s_1, \ldots, c_m \vee \neg s_m\}$ # extended formula
  - $\varphi_{unsat} = \exists S\, \forall X.\ \neg \varphi_R$ # QBF for finding UNSAT subformula
  - $f(s_1, \ldots, s_m) = \sum_{i=1}^{m} s_i$ # objective function

  - **find** $\mathcal{A}_S \in \mathcal{M}(\varphi_{unsat})$ s. t. $\forall \mathcal{B}_S \in \mathcal{M}(\varphi_{unsat})$: $f(\mathcal{A}_S) \leqslant f(\mathcal{B}_S)$

SMUS as QMaxSAT

- Original formula $\varphi = \{c_1, \ldots, c_m\}$
- QBF formulation of SMUS:
  - $S = \{s_1, \ldots, s_m\}$ `# selection variables`
  - $\varphi_R = \{c_1 \vee \neg s_1, \ldots, c_m \vee \neg s_m\}$ `# extended formula`
  - $\varphi_{unsat} = \exists S\, \forall X.\ \neg \varphi_R$ `# QBF for finding UNSAT subformula`
  - $f(s_1, \ldots, s_m) = \sum_{i=1}^{m} s_i$ `# objective function`
  - **find** $\mathcal{A}_S \in \mathcal{M}(\varphi_{unsat})$ s. t. $\forall \mathcal{B}_S \in \mathcal{M}(\varphi_{unsat})$: $f(\mathcal{A}_S) \leqslant f(\mathcal{B}_S)$
- Result QBF to decide iteratively:
  - $\exists S\, \forall X.\ \neg \varphi_R \wedge (f(s_1, \ldots, s_m) \leqslant k)$ `# linear search`

SMUS as QMaxSAT

- Original formula $\varphi = \{c_1, \dots, c_m\}$

- QBF formulation of SMUS:
  - $S = \{s_1, \dots, s_m\}$ # selection variables
  - $\varphi_R = \{c_1 \vee \neg s_1, \dots, c_m \vee \neg s_m\}$ # extended formula
  - $\varphi_{unsat} = \exists S\, \forall X.\ \neg \varphi_R$ # QBF for finding UNSAT subformula
  - $f(s_1, \dots, s_m) = \sum_{i=1}^{m} s_i$ # objective function

  - **find** $\mathcal{A}_S \in \mathcal{M}(\varphi_{unsat})$ s. t. $\forall \mathcal{B}_S \in \mathcal{M}(\varphi_{unsat})$: $f(\mathcal{A}_S) \leqslant f(\mathcal{B}_S)$

- Result QBF to decide iteratively:
  - $\exists S\, \forall X.\ \neg \varphi_R \wedge (f(s_1, \dots, s_m) \leqslant k)$ # linear search

  - $\exists S\, \forall X.\ \neg \varphi_R \wedge \varphi_S$, # core-guided search
    where $\varphi_S = \{\neg s_1, \dots, \neg s_m\}$ # soft constraints

Improvements of the Approach

Digger — state of the art for SMUS.

Improvements of the Approach

Digger — state of the art for SMUS.

Disjoint MCS enumeration can help

Improvements of the Approach

Digger — state of the art for SMUS.

Disjoint MCS enumeration can help — it improves the lower bound.

Improvements of the Approach

Digger — state of the art for SMUS.

Disjoint MCS enumeration can help — it improves the lower bound.

Definition

Formula ψ, $\psi \subseteq \varphi$, is called a ***minimal correction set*** of φ if

1. φ is unsatisfiable
2. $\varphi \setminus \psi$ is satisfiable
3. for any clause $c \in \psi$: $\varphi \setminus \psi \cup \{c\}$ is unsatisfiable.

Improvements of the Approach

Digger — state of the art for SMUS.

Disjoint MCS enumeration can help — it improves the lower bound.

Definition

Formula ψ, $\psi \subseteq \varphi$, is called a ***minimal correction set*** of φ if

1. φ is unsatisfiable
2. $\varphi \setminus \psi$ is satisfiable
3. for any clause $c \in \psi$: $\varphi \setminus \psi \cup \{c\}$ is unsatisfiable.

Fact

Any MUS of φ is a *minimal hitting set* of the complete set of MCSes of φ.

Improvements of the Approach

Digger — state of the art for SMUS.

Disjoint MCS enumeration can help — it improves the lower bound.

Definition

Formula ψ, $\psi \subseteq \varphi$, is called a ***minimal correction set*** of φ if

1. φ is unsatisfiable
2. $\varphi \setminus \psi$ is satisfiable
3. for any clause $c \in \psi$: $\varphi \setminus \psi \cup \{c\}$ is unsatisfiable.

Fact

Any MUS of φ is a *minimal hitting set* of the complete set of MCSes of φ.

Core-guided search cannot use lower bounds, but

Improvements of the Approach

Digger — state of the art for SMUS.

Disjoint MCS enumeration can help — it improves the lower bound.

Definition

Formula ψ, $\psi \subseteq \varphi$, is called a ***minimal correction set*** of φ if

1. φ is unsatisfiable
2. $\varphi \setminus \psi$ is satisfiable
3. for any clause $c \in \psi$: $\varphi \setminus \psi \cup \{c\}$ is unsatisfiable.

Fact

Any MUS of φ is a *minimal hitting set* of the complete set of MCSes of φ.

Core-guided search cannot use lower bounds, but

- find *unit* MCSes

Improvements of the Approach

Digger — state of the art for SMUS.

Disjoint MCS enumeration can help — it improves the lower bound.

Definition

Formula ψ, $\psi \subseteq \varphi$, is called a ***minimal correction set*** of φ if

1. φ is unsatisfiable
2. $\varphi \setminus \psi$ is satisfiable
3. for any clause $c \in \psi$: $\varphi \setminus \psi \cup \{c\}$ is unsatisfiable.

Fact

Any MUS of φ is a *minimal hitting set* of the complete set of MCSes of φ.

Core-guided search cannot use lower bounds, but

- find *unit* MCSes $\Rightarrow$ SMUS contains all of them

Improvements of the Approach

Digger — state of the art for SMUS.

Disjoint MCS enumeration can help — it improves the lower bound.

Definition

Formula ψ, $\psi \subseteq \varphi$, is called a ***minimal correction set*** of φ if

1. φ is unsatisfiable
2. $\varphi \setminus \psi$ is satisfiable
3. for any clause $c \in \psi$: $\varphi \setminus \psi \cup \{c\}$ is unsatisfiable.

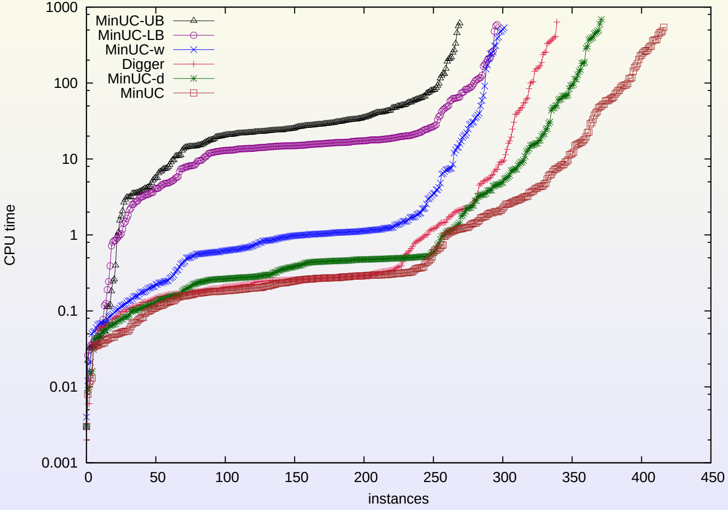
Fact

Any MUS of φ is a *minimal hitting set* of the complete set of MCSes of φ.

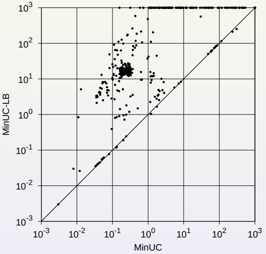
Core-guided search cannot use lower bounds, but

- find *unit* MCSes $\Rightarrow$ SMUS contains all of them
- **any** MCS is an unsatisfiable core of $\exists S\, \forall X.\ \neg\varphi_R \wedge \varphi_S$ `# see the paper`

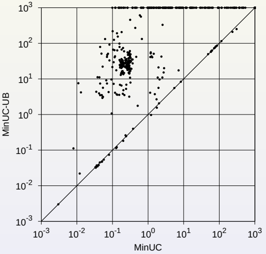
Performance Comparison: MinUC vs Digger



Performance Comparison: Linear Search vs Core-Guided

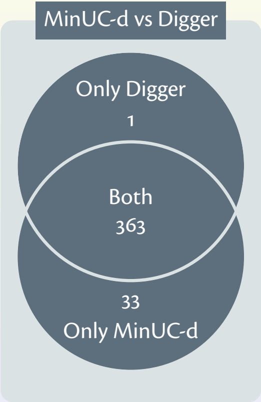


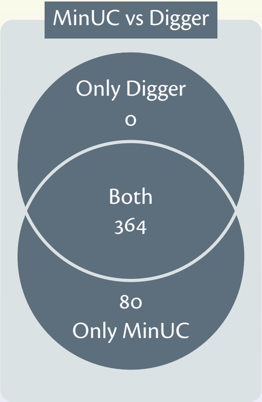
(a) MinUC vs MinUC-LB



(b) MinUC vs MinUC-UB

Number of Solved Instances





Total number of instances — 682.

Summary and Future Work

- Novel core-guided algorithm for QMaxSAT

Summary and Future Work

- Novel core-guided algorithm for QMaxSAT
- Generation of unsatisfiable cores with CEGAR-based QBF

Summary and Future Work

- Novel core-guided algorithm for QMaxSAT
- Generation of unsatisfiable cores with CEGAR-based QBF
- Smallest MUS solved as QMaxSAT

Summary and Future Work

- Novel core-guided algorithm for QMaxSAT
- Generation of unsatisfiable cores with CEGAR-based QBF
- Smallest MUS solved as QMaxSAT

- Other quantified optimization problems

Summary and Future Work

- Novel core-guided algorithm for QMaxSAT
- Generation of unsatisfiable cores with CEGAR-based QBF
- Smallest MUS solved as QMaxSAT

- Other quantified optimization problems
- More core-guided algorithms for QMaxSAT

Summary and Future Work

- Novel core-guided algorithm for QMaxSAT
- Generation of unsatisfiable cores with CEGAR-based QBF
- Smallest MUS solved as QMaxSAT

- Other quantified optimization problems
- More core-guided algorithms for QMaxSAT
- Integration of DPLL-based QBF solvers

Summary and Future Work

- Novel core-guided algorithm for QMaxSAT
- Generation of unsatisfiable cores with CEGAR-based QBF
- Smallest MUS solved as QMaxSAT

- Other quantified optimization problems
- More core-guided algorithms for QMaxSAT
- Integration of DPLL-based QBF solvers
- *CEGAR-based vs DPLL-based* comparison (unsatisfiable cores)

Thank you for your attention!