On Incremental Core-Guided MaxSAT Solving

X. Si[1], X. Zhang[1], V. Manquinho[2], M. Janota[3], **A. Ignatiev**[4,5], and M. Naik[1]

September 6, 2016

[1] Georgia Institute of Technology, USA
[2] INESC-ID, IST, University of Lisbon, Portugal
[3] Microsoft Research, Cambridge, UK
[4] LaSIGE, FC, University of Lisbon, Portugal
[5] ISDCT SB RAS, Irkutsk, Russia

Table of contents

Maximum satisfiability

Maximum satisfiability

$$\text{given } F = F_{hard} \wedge F_{soft} \models \bot,$$

Maximum satisfiability

$$\text{given } F = F_{hard} \wedge F_{soft} \models \bot,$$

$$\text{satisfy } F_{hard} \text{ and maximize } \sum_{c \in F_{soft}} weight(c)$$

Maximum satisfiability

$$\text{given } F = F_{hard} \wedge F_{soft} \models \bot,$$

$$\text{satisfy } F_{hard} \text{ and maximize } \sum_{c \in F_{soft}} weight(c)$$

$$\begin{array}{lcccc} F_{hard} & = & (\neg x \vee \neg y) & (\neg x \vee \neg z) & (\neg y \vee \neg z) \\ F_{soft} & = & (10, x) & (20, y) & (40, z) \end{array}$$

Maximum satisfiability

$$\text{given } F = F_{hard} \wedge F_{soft} \models \bot,$$

$$\text{satisfy } F_{hard} \text{ and maximize } \sum_{c \in F_{soft}} weight(c)$$

$$
\begin{array}{lcccc}
F_{hard} & = & (\neg x \vee \neg y) & (\neg x \vee \neg z) & (\neg y \vee \neg z) \\
F_{soft} & = & (10, x) & (20, y) & (40, z)
\end{array}
$$

Maximum satisfiability

$$\text{given } F = F_{hard} \wedge F_{soft} \models \bot,$$

$$\text{satisfy } F_{hard} \text{ and maximize } \sum_{c \in F_{soft}} weight(c)$$

$$\begin{array}{lcccc} F_{hard} & = & (\neg x \vee \neg y) & (\neg x \vee \neg z) & (\neg y \vee \neg z) \\ F_{soft} & = & (10, x) & (20, y) & (40, z) \end{array}$$

Maximum satisfiability

$$\text{given } F = F_{hard} \wedge F_{soft} \models \bot,$$

$$\text{satisfy } F_{hard} \text{ and maximize } \sum_{c \in F_{soft}} weight(c)$$

$$\begin{array}{lcccc} F_{hard} & = & (\neg x \vee \neg y) & (\neg x \vee \neg z) & (\neg y \vee \neg z) \\ F_{soft} & = & (10, x) & (20, y) & (40, z) \end{array}$$

Maximum satisfiability

given $F = F_{hard} \wedge F_{soft} \models \bot$,

satisfy F_{hard} and maximize $\sum_{c \in F_{soft}} weight(c)$

$$F_{hard} = (\neg x \vee \neg y) \quad (\neg x \vee \neg z) \quad (\neg y \vee \neg z)$$
$$F_{soft} = (10, x) \quad (20, y) \quad (40, z)$$

$$F_{hard} = (\neg x \vee \neg y) \quad (\neg x \vee \neg z) \quad (\neg y \vee \neg z)$$
$$F_{soft} = (10, x) \quad (20, y) \quad (40, z)$$

Scenario

Scenario

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

$|F_i|$ — up to 10^8

(e.g. Markov Logic Networks[1])

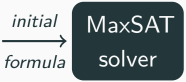
[1] R. Mangal, X. Zhang, A. Kamath, A. Nori, M. Naik: *Scaling Relational Inference Using Proofs and Refutations*. AAAI 2016: 3278–3286

Scenario

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

$|F_i|$ — up to 10^8

(e.g. Markov Logic Networks[1])



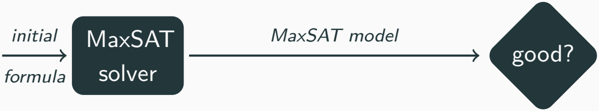
[1]R. Mangal, X. Zhang, A. Kamath, A. Nori, M. Naik: *Scaling Relational Inference Using Proofs and Refutations*. AAAI 2016: 3278–3286

Scenario

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

$|F_i|$ — up to 10^8

(e.g. Markov Logic Networks[1])



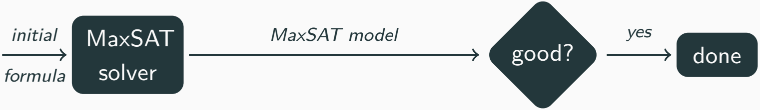
[1]R. Mangal, X. Zhang, A. Kamath, A. Nori, M. Naik: *Scaling Relational Inference Using Proofs and Refutations*. AAAI 2016: 3278–3286

Scenario

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

$|F_i|$ — up to 10^8

(e.g. Markov Logic Networks[1])



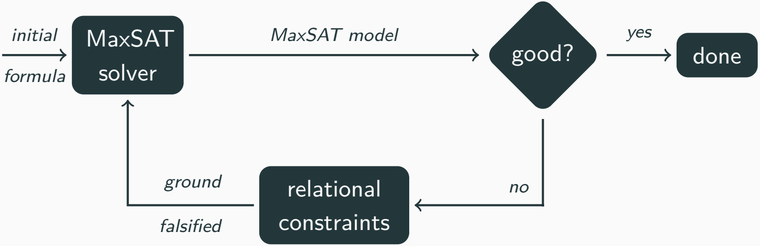
[1] R. Mangal, X. Zhang, A. Kamath, A. Nori, M. Naik: *Scaling Relational Inference Using Proofs and Refutations*. AAAI 2016: 3278–3286

Scenario

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

$|F_i|$ — up to 10^8

(e.g. Markov Logic Networks[1])



[1] R. Mangal, X. Zhang, A. Kamath, A. Nori, M. Naik: *Scaling Relational Inference Using Proofs and Refutations*. AAAI 2016: 3278–3286

Fu&Malik algorithm for MaxSAT

Fu&Malik algorithm for MaxSAT (without weights)

Input: $\phi = \phi_H \cup \phi_S$
Output: optimal solution to ϕ

$cost \leftarrow 0$
while true**:**
  $(\text{st}, \nu, \phi_{core}) \leftarrow \texttt{SAT}(\phi)$
  if $\text{st} = \text{SAT}$**: return** ν, $cost$
  $cost \leftarrow cost + 1$
  $V_R \leftarrow \emptyset$ // relax variables of the core
  foreach $c \in \phi_{core}$**:**
    if $c \in \phi_S$**:**
      $V_R \leftarrow V_R \cup \{r\}$ // r is a fresh relaxation variable
      $\phi \leftarrow \phi \setminus \{c\} \cup \{c \vee r\}$
  if $V_R = \emptyset$**: return** UNSAT // no soft clauses in the core
  $\phi \leftarrow \phi \cup CNF(\sum_{r \in V_R} r \leqslant 1)$ // add hard cardinality constraint

Fu&Malik algorithm for weighted MaxSAT

Input: $\phi = \phi_H \cup \phi_S$

Output: optimal solution to ϕ

$cost \leftarrow 0$

while true**:**

  $(\mathrm{st}, \nu, \phi_{core}) \leftarrow \mathtt{SAT}(\phi)$

  if st = SAT**: return** ν, $cost$

  $// cost \leftarrow cost + 1$

  $w_{min} \leftarrow min\{w \mid c \in \phi_C \wedge (w, c) \in \phi_S\}$ `// weight of UNSAT core`

  $cost \leftarrow cost + w_{min}$

  $V_R \leftarrow \emptyset$

  foreach $c \in \phi_{core}$**:**

    if $(w, c) \in \phi_S$**:**

      $V_R \leftarrow V_R \cup \{r\}$

      $// \phi \leftarrow \phi \setminus \{c\} \cup \{c \vee r\}$

      $\phi \leftarrow \phi \setminus \{(w, c)\} \cup \{(w - w_{min}, c), (w_{min}, c \vee r)\}$ `// split`

  if $V_R = \emptyset$**: return** UNSAT

  $\phi \leftarrow \phi \cup CNF(\sum_{r \in V_R} r \leqslant 1)$

Fu&Malik algorithm for MaxSAT

$$F_{hard} = (\neg x \vee \neg y) \quad (\neg x \vee \neg z) \quad (\neg y \vee \neg z)$$

$$F_{soft} = (10, x) \quad (20, y) \quad (40, z)$$

Fu&Malik algorithm for MaxSAT

$$F_{hard} = (\neg x \vee \neg y) \quad (\neg x \vee \neg z) \quad (\neg y \vee \neg z)$$

$$F_{soft} = (10, x) \quad (20, y) \quad (40, z)$$

Fu&Malik algorithm for MaxSAT

$$
\begin{array}{llccc}
F_{hard} & = & (\neg x \vee \neg y) & (\neg x \vee \neg z) & (\neg y \vee \neg z) \\
 & & r_1 + r_2 \leqslant 1 & & \\
F_{soft} & = & \cancel{(10, x)} & \cancel{(20, y)} & (40, z) \\
 & & (10, x \vee r_1) & (10, y \vee r_2) & \\
 & & & (10, y) & \\
\end{array}
$$

$$
cost = 10
$$

Fu&Malik algorithm for MaxSAT

$$F_{hard} = (\neg x \vee \neg y) \quad (\neg x \vee \neg z) \quad (\neg y \vee \neg z)$$

$$r_1 + r_2 \leqslant 1$$

$$F_{soft} = \cancel{(10, x)} \quad \cancel{(20, y)} \quad (40, z)$$

$$(10, x \vee r_1) \quad (10, y \vee r_2)$$

$$(10, y)$$

$$cost = 10$$

Fu&Malik algorithm for MaxSAT

$$F_{hard} = (\neg x \vee \neg y) \quad (\neg x \vee \neg z) \quad (\neg y \vee \neg z)$$

$$r_1 + r_2 \leqslant 1 \quad r_3 + r_4 + r_5 \leqslant 1$$

$$F_{soft} = \cancel{(10, x)} \quad \cancel{(20, y)} \quad \cancel{(40, z)}$$

$$\cancel{(10, x \vee r_1)} \quad \cancel{(10, y \vee r_2)}$$

$$(10, y)$$

$$(10, x \vee r_1 \vee r_3) \quad (10, y \vee r_2 \vee r_4) \quad (10, z \vee r_5)$$

$$(30, z)$$

$$cost = 20$$

Fu&Malik algorithm for MaxSAT

$$
\begin{array}{llccc}
F_{hard} & = & (\neg x \vee \neg y) & (\neg x \vee \neg z) & (\neg y \vee \neg z) \\
 & & r_1 + r_2 \leqslant 1 & r_3 + r_4 + r_5 \leqslant 1 & \\
F_{soft} & = & \cancel{(10, x)} & \cancel{(20, y)} & \cancel{(40, z)} \\
 & & \cancel{(10, x \vee r_1)} & \cancel{(10, y \vee r_2)} & \\
 & & & (10, y) & \\
 & & (10, x \vee r_1 \vee r_3) & (10, y \vee r_2 \vee r_4) & (10, z \vee r_5) \\
 & & & & (30, z) \\
 & & & & \\
cost & = & 20 & &
\end{array}
$$

Fu&Malik algorithm for MaxSAT

$$
\begin{array}{llccc}
F_{hard} & = & (\neg x \vee \neg y) & (\neg x \vee \neg z) & (\neg y \vee \neg z) \\
 & & r_1 + r_2 \leqslant 1 & r_3 + r_4 + r_5 \leqslant 1 & r_6 + r_7 \leqslant 1 \\
F_{soft} & = & \cancel{(10, x)} & \cancel{(20, y)} & \cancel{(40, z)} \\
 & & \cancel{(10, x \vee r_1)} & \cancel{(10, y \vee r_2)} & \\
 & & & \cancel{(10, y)} & \\
 & & (10, x \vee r_1 \vee r_3) & (10, y \vee r_2 \vee r_4) & (10, z \vee r_5) \\
 & & & & \cancel{(30, z)} \\
 & & & (10, y \vee r_6) & (10, z \vee r_7) \\
 & & & & (20, z) \\
cost & = & 30 & &
\end{array}
$$

Fu&Malik algorithm for MaxSAT

F_{hard}	$=$	$(\neg x \vee \neg y)$	$(\neg x \vee \neg z)$	$(\neg y \vee \neg z)$
		$r_1 + r_2 \leqslant 1$	$r_3 + r_4 + r_5 \leqslant 1$	$r_6 + r_7 \leqslant 1$
F_{soft}	$=$	~~$(10, x)$~~	~~$(20, y)$~~	~~$(40, z)$~~
		~~$(10, x \vee r_1)$~~	~~$(10, y \vee r_2)$~~	
			~~$(10, y)$~~	
		$(10, x \vee r_1 \vee r_3)$	$(10, y \vee r_2 \vee r_4)$	$(10, z \vee r_5)$
				~~$(30, z)$~~
			$(10, y \vee r_6)$	$(10, z \vee r_7)$
				$(20, z)$
$cost$	$=$	30		

Fu&Malik algorithm for MaxSAT

$$F_{hard} = (\neg x \vee \neg y) \quad (\neg x \vee \neg z) \quad (\neg y \vee \neg z)$$

$$F_{soft} = (10, x) \quad (20, y) \quad (40, z)$$

$$cost = 30$$

Incremental approach

Incremental approach

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

Incremental approach

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

straightforward approach:

- solve each F_i independently

Incremental approach

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

straightforward approach:

- solve each F_i independently
- $F_{i+1} = F_i \cup \delta_i$ — formulas are similar

Incremental approach

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

straightforward approach:

- solve each F_i independently
- $F_{i+1} = F_i \cup \delta_i$ — formulas are similar
- MaxSAT solver **repeats** its work (SAT calls included)

Incremental approach

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

straightforward approach:

- solve each F_i independently
- $F_{i+1} = F_i \cup \delta_i$ — formulas are similar
- MaxSAT solver **repeats** its work (SAT calls included)
- inefficient

Incremental approach

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

straightforward approach:

- solve each F_i independently
- $F_{i+1} = F_i \cup \delta_i$ — formulas are similar
- MaxSAT solver **repeats** its work (SAT calls included)
- inefficient

incremental approach:

- $F_i \rightarrow \texttt{MaxSAT} \rightarrow F_i'$

Incremental approach

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

straightforward approach:

- solve each F_i independently
- $F_{i+1} = F_i \cup \delta_i$ — formulas are similar
- MaxSAT solver **repeats** its work (SAT calls included)
- inefficient

incremental approach:

- $F_i \rightarrow \texttt{MaxSAT} \rightarrow F_i'$
- F_i' is satisfiable

Incremental approach

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

straightforward approach:

- solve each F_i independently
- $F_{i+1} = F_i \cup \delta_i$ — formulas are similar
- MaxSAT solver **repeats** its work (SAT calls included)
- inefficient

incremental approach:

- $F_i \to \texttt{MaxSAT} \to F'_i$
- F'_i is satisfiable
- $F_{i+1} = F'_i \cup \delta_i$

Incremental approach

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

straightforward approach:

- solve each F_i independently
- $F_{i+1} = F_i \cup \delta_i$ — formulas are similar
- MaxSAT solver **repeats** its work (SAT calls included)
- inefficient

incremental approach:

- $F_i \to \texttt{MaxSAT} \to F_i'$
- F_i' is satisfiable
- $F_{i+1} = F_i' \cup \delta_i$
- MaxSAT solver **continues** working (no repetition)

Incremental approach

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

straightforward approach:

- solve each F_i independently
- $F_{i+1} = F_i \cup \delta_i$ — formulas are similar
- MaxSAT solver **repeats** its work (SAT calls included)
- inefficient

incremental approach:

- $F_i \rightarrow \texttt{MaxSAT} \rightarrow F'_i$
- F'_i is satisfiable
- $F_{i+1} = F'_i \cup \delta_i$
- MaxSAT solver **continues** working (no repetition)
- F'_i is reused at step $i+1$

Incremental approach

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

straightforward approach:

- solve each F_i independently
- $F_{i+1} = F_i \cup \delta_i$ — formulas are similar
- MaxSAT solver **repeats** its work (SAT calls included)
- inefficient

incremental approach:

- $F_i \rightarrow \texttt{MaxSAT} \rightarrow F_i'$
- F_i' is satisfiable
- $F_{i+1} = F_i' \cup \delta_i$
- MaxSAT solver **continues** working (no repetition)
- F_i' is reused at step $i + 1$
- **two** incrementality levels:

Incremental approach

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

straightforward approach:

- solve each F_i independently
- $F_{i+1} = F_i \cup \delta_i$ — formulas are similar
- MaxSAT solver **repeats** its work (SAT calls included)
- inefficient

incremental approach:

- $F_i \rightarrow \texttt{MaxSAT} \rightarrow F_i'$
- F_i' is satisfiable
- $F_{i+1} = F_i' \cup \delta_i$
- MaxSAT solver **continues** working (no repetition)
- F_i' is reused at step $i + 1$
- **two** incrementality levels:
  - **MaxSAT** (unsat cores)

Incremental approach

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots \subseteq F_n$$

straightforward approach:

- solve each F_i independently
- $F_{i+1} = F_i \cup \delta_i$ — formulas are similar
- MaxSAT solver **repeats** its work (SAT calls included)
- inefficient

incremental approach:

- $F_i \rightarrow \texttt{MaxSAT} \rightarrow F_i'$
- F_i' is satisfiable
- $F_{i+1} = F_i' \cup \delta_i$
- MaxSAT solver **continues** working (no repetition)
- F_i' is reused at step $i + 1$
- **two** incrementality levels:
  - **MaxSAT** (unsat cores)
  - **SAT** (learnt clauses)

Incrementality at MaxSAT level

Input: $\phi = \phi_H \cup \phi_S$

Output: optimal solution to ϕ

1 $cost \leftarrow 0$

2 **while** true**:**

3 $(\text{st}, \nu, \phi_{core}) \leftarrow \texttt{SAT}(\phi)$

4 **if** st = SAT**:**

5 //**return** ν, $cost$

6 output ν, $cost$ `// output and wait new inputs`

7 $\delta \leftarrow$read new hard or soft clauses

8 $\phi \leftarrow \phi \cup \delta$

9 goto line 2

10 $w_{min} \leftarrow min\{w \mid c \in \phi_C \wedge (w, c) \in \phi_S\}$ `// weight of UNSAT core`

11 $cost \leftarrow cost + w_{min}$

12 $V_R \leftarrow \emptyset$

13 ...

Incrementality at **SAT level**

Input: $\phi = \phi_H \cup \phi_S$

Output: optimal solution to ϕ

$cost \leftarrow 0$

while true**:**

  $(\text{st}, \nu, \phi_{core}) \leftarrow \texttt{SAT}(\phi)$

  ...

  $\phi \leftarrow \phi \setminus \{c\} \cup \{c \vee r\}$

  ...

Incrementality at SAT level

Input: $\phi = \phi_H \cup \phi_S$
Output: optimal solution to ϕ

$cost \leftarrow 0$
while true**:**
  $(\text{st}, \nu, \phi_{core}) \leftarrow \texttt{SAT}(\phi)$
  $\ldots$
  $\phi \leftarrow \phi \setminus \{c\} \cup \{c \vee r\}$
  $\ldots$

$cost \leftarrow 0$
$\phi \leftarrow \phi_H \cup \{c \vee b_c \mid (w, c) \in \phi_S\}$
$\mathcal{A} \leftarrow \{\neg b_c \mid (w, c) \in \phi_S\}$
while true**:**
  $(\text{st}, \nu, \phi_{core}) \leftarrow \texttt{SAT}(\phi, \mathcal{A})$
  $\ldots$
  $//\phi \leftarrow \phi \setminus \{c\} \cup \{c \vee r\}$
  $\mathcal{A} \leftarrow \mathcal{A} \setminus \{\neg b_c\} \cup \{b_c\}$
  $\phi \leftarrow \phi \cup \{c \vee r \vee b_r\}$
  $\ldots$

Incrementality at **both levels**

Input: $\phi = \phi_H \cup \phi_S$
Output: optimal solution to ϕ

$cost \leftarrow 0$
$\phi_W \leftarrow \phi_H \cup \{c \cup \{\texttt{blockingVar}(c)\} \mid c \in \phi_S\}$ // fresh blocking variables
$\mathcal{A} \leftarrow \{\neg\texttt{blockingVar}(c) \mid c \in \phi_S\}$ // enable all soft clauses
while true**:**
    $(\text{st}, \nu, \phi_C) \leftarrow \texttt{SAT}(\phi_W, \mathcal{A})$
    if st $=$ SAT**: return** ν, $cost$ // optimal solution to ϕ
    $V_R \leftarrow \emptyset$
    $m_C = \min\{\texttt{weight}(c) \mid c \in \phi_C \wedge \texttt{soft}(c)\}$
    $cost \leftarrow cost + m_C$
    foreach $c \in \phi_C \wedge \texttt{soft}(c)$**:**
        $V_R \leftarrow V_R \cup \{r\}$ // r is a fresh relaxation variable
        $c_r \leftarrow (c \setminus \{\texttt{blockingVar}(c)\}) \cup \{r\} \cup \{b_r\}$ // b_r is a fresh variable
        $\mathcal{A} \leftarrow \mathcal{A} \cup \{\neg b_r\}$ // enable c_r
        $\phi_W \leftarrow \phi_W \cup \{c_r\}$
        $\texttt{weight}(c_r) \leftarrow m_C$
        if $\texttt{weight}(c) > m_C$**:** $\texttt{weight}(c) \leftarrow \texttt{weight}(c) - m_C$
        else: $\mathcal{A} \leftarrow (\mathcal{A} \setminus \{\neg\texttt{blockingVar}(c)\}) \cup \{\texttt{blockingVar}(c)\}$ // disable c
    if $V_R = \emptyset$**: return** UNSAT // no soft clauses in the core
    $\phi_W \leftarrow \phi_W \cup \{\texttt{CNF}(\sum_{r \in V_R} r \leqslant 1)\}$

Poor quality cores

Poor quality cores

weighted MaxSAT splits clauses

Poor quality cores

weighted MaxSAT splits clauses

e.g. let $n \in \mathbb{N}$ and $w_1 < w_2 \in \mathbb{N}$:

Poor quality cores

weighted MaxSAT splits clauses

e.g. let $n \in \mathbb{N}$ and $w_1 < w_2 \in \mathbb{N}$:

1 $F_{soft} = \quad (w_1, b \vee \bigvee_{i=1}^{n} a_i) \quad (w_2, \neg b) \quad \bigwedge_{i=1}^{n} (w_2, \neg a_i)$

Poor quality cores

weighted MaxSAT splits clauses

e.g. let $n \in \mathbb{N}$ and $w_1 < w_2 \in \mathbb{N}$:

1 $F_{soft} = \quad (w_1, b \vee \bigvee_{i=1}^{n} a_i) \quad (w_2, \neg b) \quad \bigwedge_{i=1}^{n} (w_2, \neg a_i)$

Poor quality cores

weighted MaxSAT splits clauses

e.g. let $n \in \mathbb{N}$ and $w_1 < w_2 \in \mathbb{N}$:

1 $F_{soft} =$ ~~$(w_1, b \vee \bigvee_{i=1}^{n} a_i)$~~ ~~$(w_2, \neg b)$~~ ~~$\bigwedge_{i=1}^{n} (w_2, \neg a_i)$~~

$(w_1, b \vee \bigvee_{i=1}^{n} a_i \vee r^1)$ $(w_1, \neg b \vee r_1^2)$ $\bigwedge_{i=1}^{n} (w_1, \neg a_i \vee r_i^3)$

Poor quality cores

weighted MaxSAT splits clauses

e.g. let $n \in \mathbb{N}$ and $w_1 < w_2 \in \mathbb{N}$:

$$F_{soft} = \quad \cancel{(w_1, b \vee \bigvee_{i=1}^{n} a_i)} \quad \cancel{(w_2, \neg b)} \quad \cancel{\bigwedge_{i=1}^{n} (w_2, \neg a_i)}$$

$$(w_1, b \vee \bigvee_{i=1}^{n} a_i \vee r^1) \quad (w_1, \neg b \vee r_1^2) \quad \bigwedge_{i=1}^{n} (w_1, \neg a_i \vee r_i^3)$$

$$(w_2 - w_1, \neg b) \quad \bigwedge_{i=1}^{n} (w_2 - w_1, \neg a_i)$$

Poor quality cores

weighted MaxSAT splits clauses

e.g. let $n \in \mathbb{N}$ and $w_1 < w_2 \in \mathbb{N}$:

1 $F_{soft} =$ ~~$(w_1, b \lor \bigvee_{i=1}^{n} a_i)$~~ ~~$(w_2, \neg b)$~~ ~~$\bigwedge_{i=1}^{n} (w_2, \neg a_i)$~~

$(w_1, b \lor \bigvee_{i=1}^{n} a_i \lor r^1)$ $(w_1, \neg b \lor r_1^2)$ $\bigwedge_{i=1}^{n} (w_1, \neg a_i \lor r_i^3)$

$(w_2 - w_1, \neg b)$ $\bigwedge_{i=1}^{n} (w_2 - w_1, \neg a_i)$

2 $F_{hard} =$ (b)

Poor quality cores

weighted MaxSAT splits clauses

e.g. let $n \in \mathbb{N}$ and $w_1 < w_2 \in \mathbb{N}$:

1 $F_{soft} =$ ~~$(w_1, b \vee \bigvee_{i=1}^{n} a_i)$~~ ~~$(w_2, \neg b)$~~ ~~$\bigwedge_{i=1}^{n} (w_2, \neg a_i)$~~

$(w_1, b \vee \bigvee_{i=1}^{n} a_i \vee r^1)$ $(w_1, \neg b \vee r_1^2)$ $\bigwedge_{i=1}^{n} (w_1, \neg a_i \vee r_i^3)$

$(w_2 - w_1, \neg b)$ $\bigwedge_{i=1}^{n} (w_2 - w_1, \neg a_i)$

2 $F_{hard} =$ (b)

Poor quality cores

weighted MaxSAT splits clauses

e.g. let $n \in \mathbb{N}$ and $w_1 < w_2 \in \mathbb{N}$:

1 $F_{soft} =$ ~~$(w_1, b \vee \bigvee_{i=1}^{n} a_i)$~~ ~~$(w_2, \neg b)$~~ ~~$\bigwedge_{i=1}^{n} (w_2, \neg a_i)$~~

$(w_1, b \vee \bigvee_{i=1}^{n} a_i \vee r^1)$ $(w_1, \neg b \vee r_1^2)$ $\bigwedge_{i=1}^{n} (w_1, \neg a_i \vee r_i^3)$

~~$(w_2 - w_1, \neg b)$~~ $\bigwedge_{i=1}^{n} (w_2 - w_1, \neg a_i)$

$(w_2 - w_1, \neg b \vee r_2^2)$

2 $F_{hard} =$ (b)

Poor quality cores

weighted MaxSAT splits clauses

e.g. let $n \in \mathbb{N}$ and $w_1 < w_2 \in \mathbb{N}$:

1 $F_{soft} = \quad (w_1, b \vee \bigvee_{i=1}^{n} a_i) \quad (w_2, \neg b) \quad \bigwedge_{i=1}^{n} (w_2, \neg a_i)$

2 $F_{hard} = \quad (b)$

Poor quality cores

weighted MaxSAT splits clauses

e.g. let $n \in \mathbb{N}$ and $w_1 < w_2 \in \mathbb{N}$:

1 $F_{soft} = \quad (w_1, b \vee \bigvee_{i=1}^{n} a_i) \quad \sout{(w_2, \neg b)} \quad \bigwedge_{i=1}^{n} (w_2, \neg a_i)$

$(w_2, \neg b \vee r_1^2)$

2 $F_{hard} = \quad (b)$

Poor quality cores

weighted MaxSAT splits clauses

e.g. let $n \in \mathbb{N}$ and $w_1 < w_2 \in \mathbb{N}$:

1 $F_{soft} = \quad (w_1, b \vee \bigvee_{i=1}^{n} a_i) \quad \cancel{(w_2, \neg b)} \quad \bigwedge_{i=1}^{n} (w_2, \neg a_i)$

$(w_2, \neg b \vee r_1^2)$

2 $F_{hard} = \quad (b)$

MaxSAT restarts can help!

Poor quality cores

weighted MaxSAT splits clauses

e.g. let $n \in \mathbb{N}$ and $w_1 < w_2 \in \mathbb{N}$:

1 $F_{soft} = \quad (w_1, b \vee \bigvee_{i=1}^{n} a_i) \quad \sout{(w_2, \neg b)} \quad \bigwedge_{i=1}^{n} (w_2, \neg a_i)$

$(w_2, \neg b \vee r_1^2)$

2 $F_{hard} = \quad (b)$

MaxSAT restarts can help!

$(split\_lim_c \leqslant k \quad \forall c \in F_{soft})$

Experimental results

Experimental evaluation

- Applications:
  1. abstraction refinement

Experimental evaluation

- Applications:
  1. abstraction refinement +
  2. user-guided analysis

Experimental evaluation

- Applications:
  1. abstraction refinement +
  2. user-guided analysis +
  3. statistical relational inference

Experimental evaluation

- Applications:
  1. abstraction refinement $+$
  2. user-guided analysis $+$
  3. statistical relational inference

     $=$ 74 sequential MaxSAT problems

     $=$ 669 individual MaxSAT instances (avg. 10^6 clauses)

Experimental evaluation

- Applications:
  1. abstraction refinement +
  2. user-guided analysis +
  3. statistical relational inference

     = 74 sequential MaxSAT problems

     = 669 individual MaxSAT instances (avg. 10^6 clauses)

- new approach in Open-WBO — state of the art
  1. non-incremental
  2. incremental-without-restarts
  3. incremental (clause split 2, 5, 10, 15)

Experimental evaluation

- Applications:
  1. abstraction refinement +
  2. user-guided analysis +
  3. statistical relational inference

  = 74 sequential MaxSAT problems

  = 669 individual MaxSAT instances (avg. 10^6 clauses)

- new approach in Open-WBO — state of the art
  1. non-incremental
  2. incremental-without-restarts
  3. incremental (clause split 2, 5, 10, 15)

- Machine configuration:
  - 3GHz CPU

Experimental evaluation

- Applications:
  1. abstraction refinement +
  2. user-guided analysis +
  3. statistical relational inference

  = 74 sequential MaxSAT problems

  = 669 individual MaxSAT instances (avg. 10^6 clauses)

- new approach in Open-WBO — state of the art
  1. non-incremental
  2. incremental-without-restarts
  3. incremental (clause split 2, 5, 10, 15)

- Machine configuration:
  - 3GHz CPU
  - running Linux

Experimental evaluation

- Applications:
  1. abstraction refinement +
  2. user-guided analysis +
  3. statistical relational inference

     = 74 sequential MaxSAT problems

     = 669 individual MaxSAT instances (avg. 10^6 clauses)

- new approach in Open-WBO — state of the art
  1. non-incremental
  2. incremental-without-restarts
  3. incremental (clause split 2, 5, 10, 15)

- Machine configuration:
  - 3GHz CPU
  - running Linux
  - 30m timeout

Experimental evaluation

- Applications:
    1. abstraction refinement +
    2. user-guided analysis +
    3. statistical relational inference

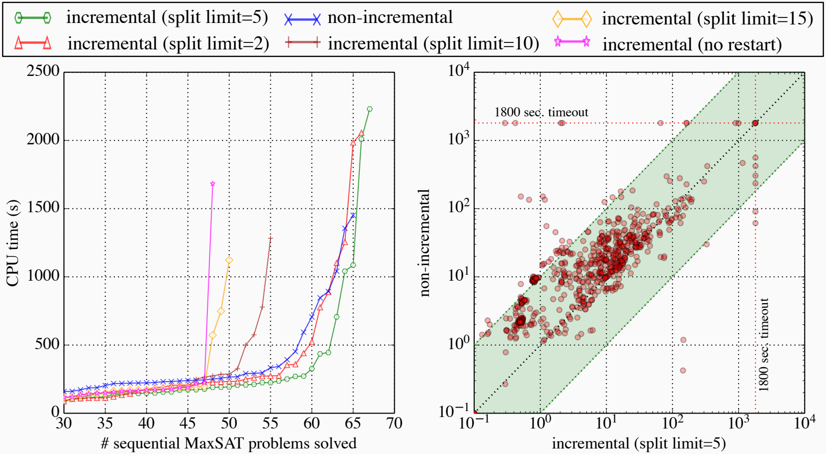
       = 74 sequential MaxSAT problems

       = 669 individual MaxSAT instances (avg. 10^6 clauses)

- new approach in Open-WBO — state of the art
    1. non-incremental
    2. incremental-without-restarts
    3. incremental (clause split 2, 5, 10, 15)

- Machine configuration:
    - 3GHz CPU
    - running Linux
    - 30m timeout
    - 32GB memout

Experimental results



(a) sequential MaxSAT problems ***(b)*** individual MaxSAT instances

Speedup over non-incremental approach

Split limit 5 vs. **non-incremental**:

- average speedup — 1.8×
- best speedup — 296× !

Summary and future work

Summary and future work

- new incremental approach to sequential MaxSAT:

Summary and future work

- new incremental approach to sequential MaxSAT:
    - incremental MaxSAT calls

Summary and future work

- new incremental approach to sequential MaxSAT:
  - incremental MaxSAT calls +
  - incremental SAT calls inside MaxSAT

Summary and future work

- new incremental approach to sequential MaxSAT:
    - incremental MaxSAT calls +
    - incremental SAT calls inside MaxSAT +
    - adaptive restarts

Summary and future work

- new incremental approach to sequential MaxSAT:
  - incremental MaxSAT calls +
  - incremental SAT calls inside MaxSAT +
  - adaptive restarts

- better restart strategies

Summary and future work

- new incremental approach to sequential MaxSAT:
  - incremental MaxSAT calls +
  - incremental SAT calls inside MaxSAT +
  - adaptive restarts

- better restart strategies
- state-of-the-art MaxSAT algorithms

Summary and future work

- new incremental approach to sequential MaxSAT:
  - incremental MaxSAT calls +
  - incremental SAT calls inside MaxSAT +
  - adaptive restarts

- better restart strategies
- state-of-the-art MaxSAT algorithms
- not only add but also delete clauses

Questions?