The (r)evolution of MaxSAT solving

Ruben Martins

University of Oxford

April 2, 2015

Software Package Upgradeability Problem

Package	Dependencies	Conflicts
p_1	$\{p_2 \vee p_3\}$	$\{p_4\}$
p_2	$\{p_3\}$	$\{\}$
p_3	$\{p_2\}$	$\{p_4\}$
p_4	$\{p_2 \wedge p_3\}$	$\{\}$

- Set of packages we want to install: $\{p_1, p_2, p_3, p_4\}$
- Each package p_i has a set of dependencies:
  - Packages that **must be** installed for p_i to be installed
- Each package p_i has a set of conflicts:
  - Packages that **cannot** be installed for p_i to be installed

Solving the Software Package Upgradeability Problem

Package	Dependencies	Conflicts
p_1	$\{p_2 \vee p_3\}$	$\{p_4\}$
p_2	$\{p_3\}$	$\{\}$
p_3	$\{p_2\}$	$\{p_4\}$
p_4	$\{p_2 \wedge p_3\}$	$\{\}$

Encode the problem to Propositional Satisfiability

- A literal l_i is either a Boolean variable x_i or $\neg x_i$:
- A clause $\omega = \bigvee_i l_i$:
  - $\omega_1 = (x_1); \omega_2 = (\neg x_1 \vee x_2 \vee x_3); \omega_3 = (\neg x_2 \vee \neg x_3)$
- CNF formula $\varphi = \bigwedge_j \omega_j$:
  - $\varphi = (\omega_1 \wedge \omega_2 \wedge \omega_3)$

Solving the Software Package Upgradeability Problem

Package	Dependencies	Conflicts
p_1	$\{p_2 \vee p_3\}$	$\{p_4\}$
p_2	$\{p_3\}$	$\{\}$
p_3	$\{p_2\}$	$\{p_4\}$
p_4	$\{p_2 \wedge p_3\}$	$\{\}$

Encode the problem to Propositional Satisfiability

- Encoding dependencies:
  - $p_1 \Rightarrow (p_2 \vee p_3) \equiv (\neg p_1 \vee p_2 \vee p_3)$
  - $p_2 \Rightarrow p_3 \equiv (\neg p_2 \vee p_3)$
  - $p_3 \Rightarrow p_2 \equiv (\neg p_3 \vee p_2)$
  - $p_4 \Rightarrow (p_2 \wedge p_3) \equiv (\neg p_4 \vee p_2) \wedge (\neg p_4 \vee p_3)$

Solving the Software Package Upgradeability Problem

Package	Dependencies	Conflicts
p_1	$\{p_2 \vee p_3\}$	$\{p_4\}$
p_2	$\{p_3\}$	$\{\}$
p_3	$\{p_2\}$	$\{p_4\}$
p_4	$\{p_2 \wedge p_3\}$	$\{\}$

Encode the problem to Propositional Satisfiability

- Encoding conflicts:
  - $p_1 \Rightarrow \neg p_4 \equiv (\neg p_1 \vee \neg p_4)$
  - $p_3 \Rightarrow \neg p_4 \equiv (\neg p_3 \vee \neg p_4)$

Solving the Software Package Upgradeability Problem

Package	Dependencies	Conflicts
p_1	$\{p_2 \vee p_3\}$	$\{p_4\}$
p_2	$\{p_3\}$	$\{\}$
p_3	$\{p_2\}$	$\{p_4\}$
p_4	$\{p_2 \wedge p_3\}$	$\{\}$

Encode the problem to Propositional Satisfiability

- Encoding installing all packages:
  - $(p_1) \wedge (p_2) \wedge (p_3) \wedge (p_4)$

Solving the Software Package Upgradeability Problem

CNF Formula:

$$\neg p_1 \vee p_2 \vee p_3 \qquad \neg p_2 \vee p_3 \qquad \neg p_3 \vee p_2$$

$$\neg p_4 \vee p_2 \qquad \neg p_4 \vee p_3 \qquad \neg p_1 \vee \neg p_4 \qquad \neg p_3 \vee \neg p_4$$

$$p_1 \qquad p_2 \qquad p_3 \qquad p_4$$

- Propositional Satisfiability (SAT):
  - Decide if the formula is satisfiable or unsatisfiable

Solving the Software Package Upgradeability Problem

CNF Formula:

$\neg p_1 \vee p_2 \vee p_3$	$\neg p_2 \vee p_3$	$\neg p_3 \vee p_2$	
$\neg p_4 \vee p_2$	$\neg p_4 \vee p_3$	$\neg p_1 \vee \neg p_4$	$\neg p_3 \vee \neg p_4$
p_1	p_2	p_3	p_4

- Formula is unsatisfiable

Solving the Software Package Upgradeability Problem

CNF Formula:

$\neg p_1 \vee p_2 \vee p_3$	$\neg p_2 \vee p_3$	$\neg p_3 \vee p_2$	
$\neg p_4 \vee p_2$	$\neg p_4 \vee p_3$	$\neg p_1 \vee \neg p_4$	$\neg p_3 \vee \neg p_4$
p_1	p_2	p_3	p_4

- Formula is unsatisfiable
- We cannot install all packages
- How many packages can we install?

What is Maximum Satisfiability?

- Maximum Satisfiability (MaxSAT):
  - Optimized version of SAT
  - All clauses in the formula are soft
  - Minimize number of unsatisfied soft clauses

What is Maximum Satisfiability?

- Maximum Satisfiability (MaxSAT):
  - Optimized version of SAT
  - All clauses in the formula are soft
  - Minimize number of unsatisfied soft clauses
- Partial MaxSAT:
  - Clauses in the formula are soft or hard
  - Hard clauses must be satisfied
  - Minimize number of unsatisfied soft clauses
- Weighted Partial MaxSAT:
  - Clauses in the formula are soft or hard
  - Weights associated with soft clauses
  - Minimize sum of weights of unsatisfied soft clauses

Software Package Upgradeability Problem as MaxSAT

Partial MaxSAT Formula:

φ_h (Hard):	$\neg p_1 \lor p_2 \lor p_3$	$\neg p_2 \lor p_3$	$\neg p_3 \lor p_2$	
	$\neg p_4 \lor p_2$	$\neg p_4 \lor p_3$	$\neg p_1 \lor \neg p_4$	$\neg p_3 \lor \neg p_4$
φ_s (Soft):	p_1	p_2	p_3	p_4

- Dependencies and conflicts are encoded as hard clauses
- Installation of packages are encoded as soft clauses
- **Goal:** maximize the number of installed packages

Software Package Upgradeability Problem as MaxSAT

Partial MaxSAT Formula:

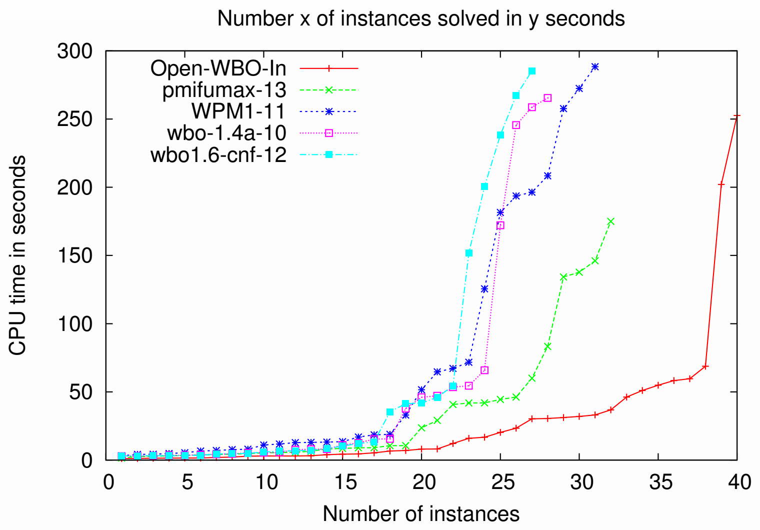
φ_h (Hard):	$\neg p_1 \vee p_2 \vee p_3$	$\neg p_2 \vee p_3$	$\neg p_3 \vee p_2$	
	$\neg p_4 \vee p_2$	$\neg p_4 \vee p_3$	$\neg p_1 \vee \neg p_4$	$\neg p_3 \vee \neg p_4$
φ_s (Soft):	p_1	p_2	p_3	p_4

- Dependencies and conflicts are encoded as hard clauses
- Installation of packages are encoded as soft clauses
- **Optimal solution** (3 out 4 packages are installed):
  - $\mu = \{p_1 = 1, p_2 = 1, p_3 = 1, p_4 = 0\}$

Why is MaxSAT Important?

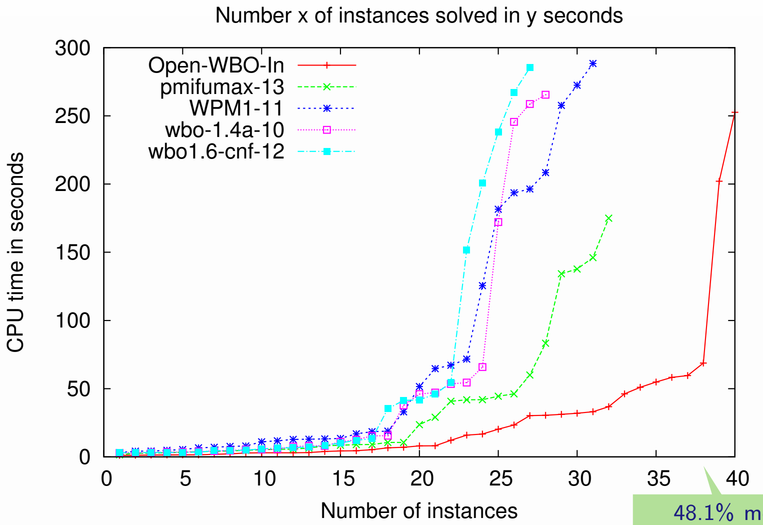
- Many real-world applications can be encoded to MaxSAT:
    - Software package upgradeability:
        - Eclipse platform uses MaxSAT for managing the plugins dependencies
    - Error localization in C code
    - Debugging of hardware designs
    - Haplotyping with pedigrees
    - Reasoning over Biological Networks
    - Course timetabling
    - Combinatorial auctions
    - ...
- MaxSAT algorithms are effective for solving real-word problems

The MaxSAT (r)evolution – plain industrial instances



Source: [MaxSAT 2014 organizers]

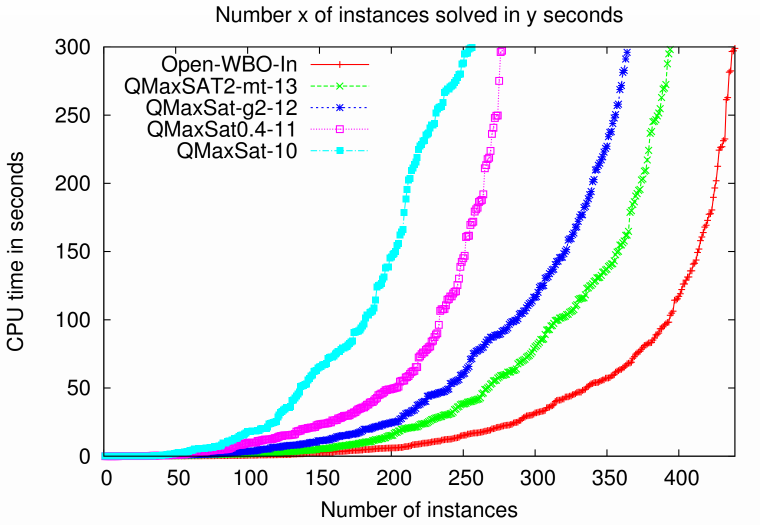
The MaxSAT (r)evolution – plain industrial instances



Source: [MaxSAT 2014 organizers]

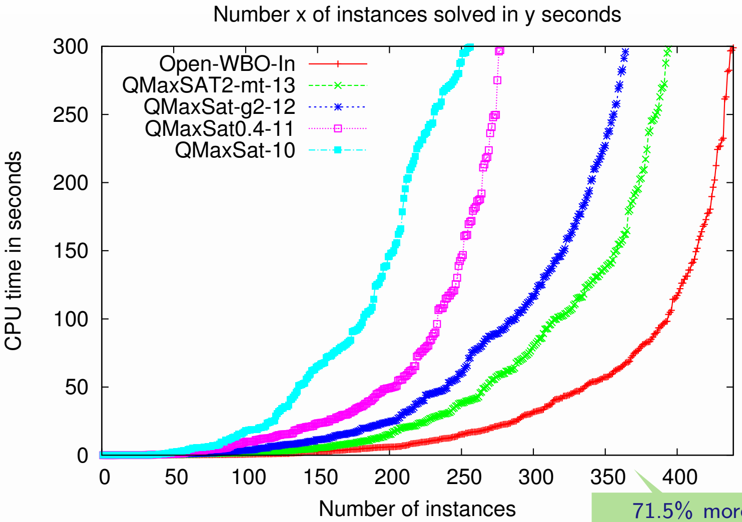
48.1% more instances solved!

The MaxSAT (r)evolution – partial



Source: [MaxSAT 2014 organizers]

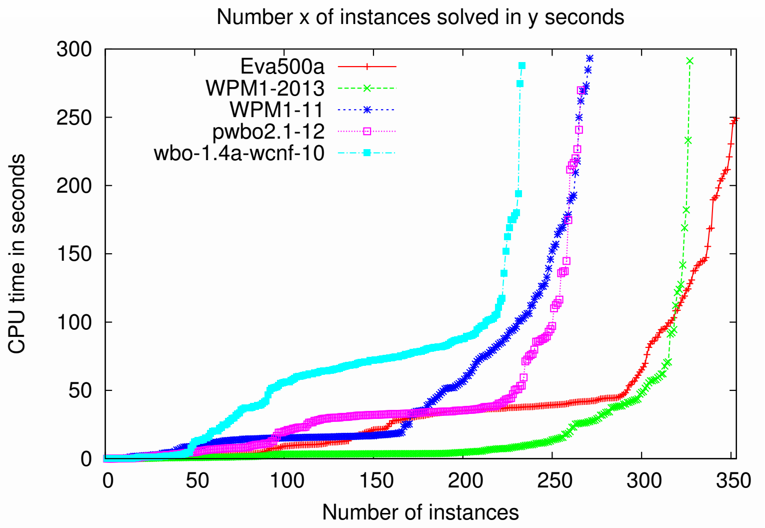
The MaxSAT (r)evolution – partial



71.5% more instances solved!

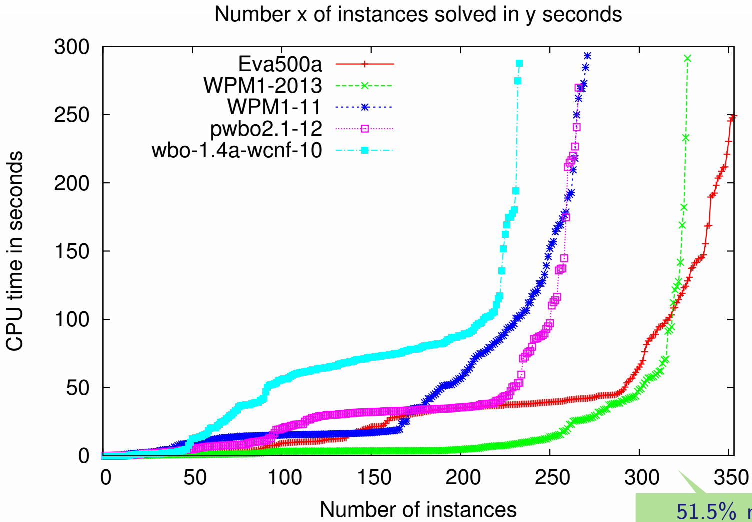
Source: [MaxSAT 2014 organizers]

The MaxSAT (r)evolution – weighted partial



Source: [MaxSAT 2014 organizers]

The MaxSAT (r)evolution – weighted partial



Source: [MaxSAT 2014 organizers]

51.5% more instances solved**!**

Outline

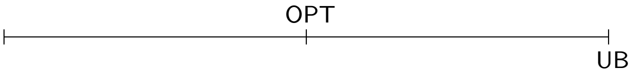
- MaxSAT Algorithms:
  - Linear search algorithms
  - Unsatisfiability-based algorithms
- Incremental solving in MaxSAT:
  - Keep the state of the SAT solver between MaxSAT calls
- Partitioning in MaxSAT:
  - Use the structure of the problem to guide the search

Outline

- MaxSAT Algorithms:
  - Linear search algorithms
  - Unsatisfiability-based algorithms
- Incremental solving in MaxSAT:
  - Keep the state of the SAT solver between MaxSAT calls
- Partitioning in MaxSAT:
  - Use the structure of the problem to guide the search

MaxSAT algorithms

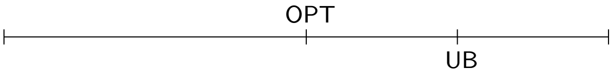
SAT-UNSAT Linear Search algorithm:



- Optimum solution (OPT):
  - Assignment with **minimum** cost
- Upper Bound (UB) value:
  - Cost **greater than or equal** to OPT
- SAT-UNSAT Linear search algorithms:
  - Iterative calls to a SAT solver
  - Refine UB value until OPT is found

MaxSAT algorithms

SAT-UNSAT Linear Search algorithm:

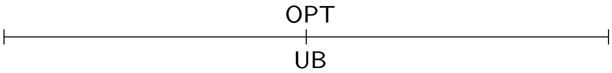


- Optimum solution (OPT):
  - Assignment with **minimum** cost
- Upper Bound (UB) value:
  - Cost **greater than or equal** to OPT

- SAT-UNSAT Linear search algorithms:
  - Iterative calls to a SAT solver
  - Refine UB value until OPT is found

MaxSAT algorithms

SAT-UNSAT Linear Search algorithm:



- Optimum solution (OPT):
  - Assignment with **minimum** cost
- Upper Bound (UB) value:
  - Cost **greater than or equal** to OPT

- SAT-UNSAT Linear search algorithms:
  - Iterative calls to a SAT solver
  - Refine UB value until OPT is found

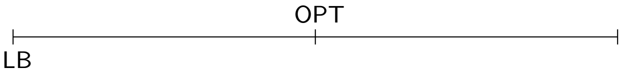
MaxSAT algorithms

SAT-UNSAT Linear Search algorithm:

- Algorithm is **incremental**
  - Only **one** SAT solver must be created
  - New variables and constraints are added between SAT solver calls
  - SAT solver calls are performed on refinements of the previous formula
- In incremental algorithms:
  - No need to rebuild the SAT solver between iterations
  - Keep all learned clauses
  - Keep internal state of the SAT solver between calls

MaxSAT algorithms

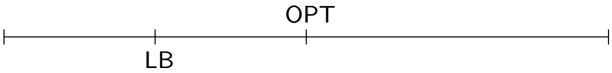
Unsatisfiability-based algorithms:



- Lower Bound (LB) value:
  - Cost **smaller than or equal** to OPT
- Unsatisfiability-based algorithms:
  - Iteratively increase the LB until a satisfiable call is performed
  - Use unsatisfiable subformulas to refine LB value until OPT is found
- These algorithms are not incremental
  - SAT solver is rebuilt at each iteration

MaxSAT algorithms

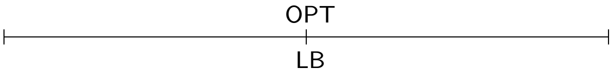
Unsatisfiability-based algorithms:



- Lower Bound (LB) value:
  - Cost **smaller than or equal** to OPT
- Unsatisfiability-based algorithms:
  - Iteratively increase the LB until a satisfiable call is performed
  - Use unsatisfiable subformulas to refine LB value until OPT is found
- These algorithms are not incremental
  - SAT solver is rebuilt at each iteration

MaxSAT algorithms

Unsatisfiability-based algorithms:



- Lower Bound (LB) value:
  - Cost **smaller than or equal** to OPT
- Unsatisfiability-based algorithms:
  - Iteratively increase the LB until a satisfiable call is performed
  - Use unsatisfiable subformulas to refine LB value until OPT is found
- These algorithms are not incremental
  - SAT solver is rebuilt at each iteration

MaxSAT Algorithms

- MaxSAT algorithms build on SAT solver technology:
  - Unsatisfiable subformulas (or cores)
- MaxSAT algorithms use constraints not defined in causal form:
  - AtMost1 constraints, $\sum_{j=1}^{n} x_j \leq 1$
  - General cardinality constraints, $\sum_{j=1}^{n} x_j \leq k$
  - Pseudo-Boolean constraints, $\sum_{j=1}^{n} a_j x_j \leq k$
- Efficient encodings to CNF

CNF Encodings

Naive encoding for AtMost1 Constraints:

- $x_1 + x_2 + x_3 \leq 1$:
  - $(\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3)$
- For a general AtMost1 constraint $r_1 + r_2 + \ldots + r_n \leq 1$:
  - For each pair (r_i, r_j) add the clause $(\neg r_i \vee \neg r_j)$
- Complexity: $\mathcal{O}(n^2)$ clauses
- More efficient encodings can be used! (PBLib'15)

CNF Encodings

Sequential counters (Sinz [CP'05])

- AtMost1 constraints:
  - Clauses/Variables: $\mathcal{O}(n)$
- General cardinality constraints:
  - Clauses/Variables: $\mathcal{O}(n\ k)$

Sequential weighted counters (Hölldobler et al. [KI'12])

- Pseudo-Boolean constraints:
  - Clauses/Variables: $\mathcal{O}(n\ k)$

Linear Search Algorithms SAT-UNSAT

Partial MaxSAT Formula:

φ_h (Hard):		$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	
φ_s (Soft):	x_1	x_3	$x_2 \vee \bar{x}_1$	$\bar{x}_3 \vee x_1$

Linear Search Algorithms SAT-UNSAT

Partial MaxSAT Formula:

$$\varphi_h : \quad \bar{x}_2 \vee \bar{x}_1 \qquad x_2 \vee \bar{x}_3$$

$$\varphi_s : \quad x_1 \vee \boxed{r_1} \qquad x_3 \vee \boxed{r_2} \qquad x_2 \vee \bar{x}_1 \vee \boxed{r_3} \qquad \bar{x}_3 \vee x_1 \vee \boxed{r_4}$$

- Relax all soft clauses
- Relaxation variables:
  - $V_R = \{r_1, r_2, r_3, r_4\}$
  - If a soft clause ω_i is **unsatisfied**, then $r_i = 1$
  - If a soft clause ω_i is **satisfied**, then $r_i = 0$

Linear Search Algorithms SAT-UNSAT

Partial MaxSAT Formula:

φ_h :		$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	
φ_s :	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \bar{x}_1 \vee r_3$	$\bar{x}_3 \vee x_1 \vee r_4$

$V_R = \{r_1, r_2, r_3, r_4\}$

- Formula is satisfiable
  - $\nu = \{x_1 = 1, x_2 = 0, x_3 = 0, r_1 = 0, r_2 = 1, r_3 = 1, r_4 = 0\}$
- **Goal:** Minimize the number of relaxation variables assigned to 1

Linear Search Algorithms SAT-UNSAT

Partial MaxSAT Formula:

φ_h :		$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	
φ_s :	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \bar{x}_1 \vee r_3$	$\bar{x}_3 \vee x_1 \vee r_4$

$\mu = 2 \qquad V_R = \{r_1, r_2, r_3, r_4\}$

- r_2 and r_3 were assigned truth value 1:
  - Current solution unsatisfies 2 soft clauses
- Can less than 2 soft clauses be unsatisfied?

Linear Search Algorithms SAT-UNSAT

Partial MaxSAT Formula:

$$\varphi_h : \quad \bar{x}_2 \vee \bar{x}_1 \qquad x_2 \vee \bar{x}_3 \qquad \text{CNF}(\textstyle\sum_{r_i \in V_R} r_i \leq 1)$$

$$\varphi_s : \quad x_1 \vee r_1 \qquad x_3 \vee r_2 \qquad x_2 \vee \bar{x}_1 \vee r_3 \qquad \bar{x}_3 \vee x_1 \vee r_4$$

$$\mu = 2 \qquad V_R = \{r_1, r_2, r_3, r_4\}$$

- Add cardinality constraint that excludes solutions that unsatisfies 2 or more soft clauses:
  - CNF($r_1 + r_2 + r_3 + r_4 \leq 1$)

Linear Search Algorithms SAT-UNSAT

Partial MaxSAT Formula:

$$\varphi_h : \quad \bar{x}_2 \vee \bar{x}_1 \qquad x_2 \vee \bar{x}_3 \qquad \mathrm{CNF}(\textstyle\sum_{r_i \in V_R} r_i \leq 1)$$

$$\varphi_s : \quad x_1 \vee r_1 \qquad x_3 \vee r_2 \qquad x_2 \vee \bar{x}_1 \vee r_3 \qquad \bar{x}_3 \vee x_1 \vee r_4$$

$$\mu = 2 \qquad V_R = \{r_1, r_2, r_3, r_4\}$$

- Formula is unsatisfiable:
  - There are no solutions that unsatisfy 1 or less soft clauses

Linear Search Algorithms SAT-UNSAT

Partial MaxSAT Formula:

φ_h:		$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	
φ_s:	x_1	x_3	$x_2 \vee \bar{x}_1$	$\bar{x}_3 \vee x_1$

$\mu = 2 \qquad V_R = \{r_1, r_2, r_3, r_4\}$

- **Optimal solution**: given by the last model and corresponds to unsatisfying 2 soft clauses:
  - $\nu = \{x_1 = 1, x_2 = 0, x_3 = 0\}$
- The same procedure can be generalized to weighted

Linear Search Algorithms UNSAT-SAT

Partial MaxSAT Formula:

$$\varphi_h : \quad \bar{x}_2 \vee \bar{x}_1 \qquad x_2 \vee \bar{x}_3$$

$$\varphi_s : \quad x_1 \vee \boxed{r_1} \qquad x_3 \vee \boxed{r_2} \qquad x_2 \vee \bar{x}_1 \vee \boxed{r_3} \qquad \bar{x}_3 \vee x_1 \vee \boxed{r_4}$$

- Relax all soft clauses
- Relaxation variables:
  - $V_R = \{r_1, r_2, r_3, r_4\}$
  - If a soft clause ω_i is **unsatisfied**, then $r_i = 1$
  - If a soft clause ω_i is **satisfied**, then $r_i = 0$

Linear Search Algorithms UNSAT-SAT

Partial MaxSAT Formula:

φ_h : $\bar{x}_2 \vee \bar{x}_1$ $\quad x_2 \vee \bar{x}_3$ $\quad \boxed{\text{CNF}(\sum_{r_i \in V_R} r_i \leq 0)}$

φ_s : $x_1 \vee r_1$ $\quad x_3 \vee r_2$ $\quad x_2 \vee \bar{x}_1 \vee r_3$ $\quad \bar{x}_3 \vee x_1 \vee r_4$

$\mu = 2 \qquad V_R = \{r_1, r_2, r_3, r_4\}$

- Add cardinality constraint that excludes solutions that unsatisfies 1 or more soft clauses:
  - CNF($r_1 + r_2 + r_3 + r_4 \leq 0$)

Linear Search Algorithms UNSAT-SAT

Partial MaxSAT Formula:

φ_h : $\bar{x}_2 \vee \bar{x}_1$ $\quad x_2 \vee \bar{x}_3$ $\quad \text{CNF}(\sum_{r_i \in V_R} r_i \leq 0)$

φ_s : $x_1 \vee r_1$ $\quad x_3 \vee r_2$ $\quad x_2 \vee \bar{x}_1 \vee r_3$ $\quad \bar{x}_3 \vee x_1 \vee r_4$

- Formula is unsatisfiable:
  - There are no solutions that unsatisfy 0 or less soft clauses
- Add cardinality constraint that excludes solutions that unsatisfies 2 or more soft clauses:
  - $\text{CNF}(r_1 + r_2 + r_3 + r_4 \leq 1)$

Linear Search Algorithms UNSAT-SAT

Partial MaxSAT Formula:

$$\varphi_h: \quad \bar{x}_2 \vee \bar{x}_1 \qquad x_2 \vee \bar{x}_3 \qquad \text{CNF}(\textstyle\sum_{r_i \in V_R} r_i \leq 1)$$

$$\varphi_s: \quad x_1 \vee r_1 \qquad x_3 \vee r_2 \qquad x_2 \vee \bar{x}_1 \vee r_3 \qquad \bar{x}_3 \vee x_1 \vee r_4$$

- Formula is unsatisfiable:
  - There are no solutions that unsatisfy 1 or less soft clauses
- Add cardinality constraint that excludes solutions that unsatisfies 3 or more soft clauses:
  - $\text{CNF}(r_1 + r_2 + r_3 + r_4 \leq 2)$

Linear Search Algorithms UNSAT-SAT

Partial MaxSAT Formula:

φ_h :	$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	$\text{CNF}(\sum_{r_i \in V_R} r_i \leq 2)$	
φ_s :	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \bar{x}_1 \vee r_3$	$\bar{x}_3 \vee x_1 \vee r_4$

- Formula is satisfiable:
  - $\mu = \{x_1 = 1, x_2 = 0, x_3 = 0, r_1 = 0, r_2 = 1, r_3 = 1, r_4 = 0\}$
- Optimal solution unsatisfies 2 soft clauses
- The same procedure can be generalized to weighted

Unsatisfiability-based Algorithms (MSU3: Marques-Silva&Planes'07)

Partial MaxSAT Formula:

φ_h (Hard):		$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	
φ_s (Soft):	x_1	x_3	$x_2 \vee \bar{x}_1$	$\bar{x}_3 \vee x_1$

Unsatisfiability-based Algorithms (MSU3: Marques-Silva&Planes'07)

Partial MaxSAT Formula:

φ_h:		$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	
φ_s:	x_1	x_3	$x_2 \vee \bar{x}_1$	$\bar{x}_3 \vee x_1$

- Formula is unsatisfiable

Unsatisfiability-based Algorithms (MSU3: Marques-Silva&Planes'07)

Partial MaxSAT Formula:

φ_h:		$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	
φ_s:	x_1	x_3	$x_2 \vee \bar{x}_1$	$\bar{x}_3 \vee x_1$

- Formula is unsatisfiable
- Identify an unsatisfiable core

Unsatisfiability-based Algorithms (MSU3: Marques-Silva&Planes'07)

Partial MaxSAT Formula:

φ_h: $\bar{x}_2 \vee \bar{x}_1$ $\quad x_2 \vee \bar{x}_3$ $\quad \boxed{\text{CNF}(r_1 + r_2 \leq 1)}$

φ_s: $x_1 \vee \boxed{r_1}$ $\quad x_3 \vee \boxed{r_2}$ $\quad x_2 \vee \bar{x}_1$ $\quad \bar{x}_3 \vee x_1$

- Relax non-relaxed soft clauses in unsatisfiable core:
  - Add cardinality constraint that excludes solutions that unsatisfies 2 or more soft clauses:
    - CNF($r_1 + r_2 \leq 1$)
  - Relaxation on demand instead of relaxing all soft clauses eagerly

Unsatisfiability-based Algorithms (MSU3: Marques-Silva&Planes'07)

Partial MaxSAT Formula:

φ_h:	$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	$\text{CNF}(r_1 + r_2 \leq 1)$	
φ_s:	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \bar{x}_1$	$\bar{x}_3 \vee x_1$

- Formula is unsatisfiable

Unsatisfiability-based Algorithms (MSU3: Marques-Silva&Planes'07)

Partial MaxSAT Formula:

φ_h:	$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	$\mathrm{CNF}(r_1 + r_2 \leq 1)$	
φ_s:	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \bar{x}_1$	$\bar{x}_3 \vee x_1$

- Formula is unsatisfiable
- Identify an unsatisfiable core

Unsatisfiability-based Algorithms (MSU3: Marques-Silva&Planes'07)

Partial MaxSAT Formula:

φ_h: $\bar{x}_2 \vee \bar{x}_1$ $x_2 \vee \bar{x}_3$ $\mathrm{CNF}(r_1 + r_2 + r_3 + r_4 \leq 2)$

φ_s: $x_1 \vee r_1$ $x_3 \vee r_2$ $x_2 \vee \bar{x}_1 \vee r_3$ $\bar{x}_3 \vee x_1 \vee r_4$

- Relax non-relaxed soft clauses in unsatisfiable core:
  - Add cardinality constraint that excludes solutions that unsatisfies 3 or more soft clauses:
    - $\mathrm{CNF}(r_1 + r_2 + r_3 + r_4 \leq 2)$
  - Relaxation on demand instead of relaxing all soft clauses eagerly

Unsatisfiability-based Algorithms (MSU3: Marques-Silva&Planes'07)

Partial MaxSAT Formula:

$$\varphi_h : \quad \bar{x}_2 \vee \bar{x}_1 \qquad x_2 \vee \bar{x}_3 \qquad \text{CNF}(r_1 + r_2 + r_3 + r_4 \leq 2)$$

$$\varphi_s : \quad x_1 \vee r_1 \qquad x_3 \vee r_2 \qquad x_2 \vee \bar{x}_1 \vee r_3 \qquad \bar{x}_3 \vee x_1 \vee r_4$$

- Formula is satisfiable:
  - $\mu = \{x_1 = 1, x_2 = 0, x_3 = 0, r_1 = 0, r_2 = 1, r_3 = 1, r_4 = 0\}$
- Optimal solution unsatisfies 2 soft clauses
- The same procedure can be generalized to weighted

Unsatisfiability-based Algorithms (Fu&Malik [SAT'06])

Partial MaxSAT Formula:

φ_h (Hard):		$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	
φ_s (Soft):	x_1	x_3	$x_2 \vee \bar{x}_1$	$\bar{x}_3 \vee x_1$

Unsatisfiability-based Algorithms (Fu&Malik [SAT'06])

Partial MaxSAT Formula:

φ_h:		$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	
φ_s:	x_1	x_3	$x_2 \vee \bar{x}_1$	$\bar{x}_3 \vee x_1$

- Formula is unsatisfiable

Unsatisfiability-based Algorithms (Fu&Malik [SAT'06])

Partial MaxSAT Formula:

φ_h:		$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	
φ_s:	x_1	x_3	$x_2 \vee \bar{x}_1$	$\bar{x}_3 \vee x_1$

- Formula is unsatisfiable
- Identify an unsatisfiable core

Unsatisfiability-based Algorithms (Fu&Malik [SAT'06])

Partial MaxSAT Formula:

φ_h:	$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	$\boxed{\mathrm{CNF}(r_1 + r_2 \leq 1)}$	
φ_s:	$x_1 \vee \boxed{r_1}$	$x_3 \vee \boxed{r_2}$	$x_2 \vee \bar{x}_1$	$\bar{x}_3 \vee x_1$

- Relax unsatisfiable core:
  - Add relaxation variables
  - Add AtMost1 constraint

Unsatisfiability-based Algorithms (Fu&Malik [SAT'06])

Partial MaxSAT Formula:

φ_h: $\bar{x}_2 \vee \bar{x}_1$ $\quad x_2 \vee \bar{x}_3$ $\quad \mathrm{CNF}(r_1 + r_2 \leq 1)$

φ_s: $x_1 \vee r_1$ $\quad x_3 \vee r_2$ $\quad x_2 \vee \bar{x}_1$ $\quad \bar{x}_3 \vee x_1$

- Formula is unsatisfiable

Unsatisfiability-based Algorithms (Fu&Malik [SAT'06])

Partial MaxSAT Formula:

φ_h:	$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	$\text{CNF}(r_1 + r_2 \leq 1)$	
φ_s:	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \bar{x}_1$	$\bar{x}_3 \vee x_1$

- Formula is unsatisfiable
- Identify an unsatisfiable core

Unsatisfiability-based Algorithms (Fu&Malik [SAT'06])

Partial MaxSAT Formula:

φ_h: $\bar{x}_2 \vee \bar{x}_1$ $\quad x_2 \vee \bar{x}_3$ $\quad \mathrm{CNF}(r_1 + r_2 \leq 1)$ $\quad \boxed{\mathrm{CNF}(r_3 + \ldots + r_6 \leq 1)}$

φ_s: $x_1 \vee r_1 \vee \boxed{r_3}$ $\quad x_3 \vee r_2 \vee \boxed{r_4}$ $\quad x_2 \vee \bar{x}_1 \vee \boxed{r_5}$ $\quad \bar{x}_3 \vee x_1 \vee \boxed{r_6}$

- Relax unsatisfiable core:
  - Add relaxation variables
  - Add AtMost1 constraint
- Soft clauses may be relaxed multiple times

Unsatisfiability-based Algorithms (Fu&Malik [SAT'06])

Partial MaxSAT Formula:

φ_h:	$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	$\text{CNF}(r_1 + r_2 \leq 1)$	$\text{CNF}(r_3 + \ldots + r_6 \leq 1)$
φ_s:	$x_1 \vee r_1 \vee r_3$	$x_3 \vee r_2 \vee r_4$	$x_2 \vee \bar{x}_1 \vee r_5$	$\bar{x}_3 \vee x_1 \vee r_6$

- Formula is satisfiable
- An optimal solution would be:
  - $\nu = \{x_1 = 1, x_2 = 0, x_3 = 0\}$

Unsatisfiability-based Algorithms (Fu&Malik [SAT'06])

Partial MaxSAT Formula:

φ_h:		$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	
φ_s:	x_1	x_3	$x_2 \vee \bar{x}_1$	$\bar{x}_3 \vee x_1$

- Formula is satisfiable
- An optimal solution would be:
  - $\nu = \{x_1 = 1, x_2 = 0, x_3 = 0\}$
- This assignment unsatisfies 2 soft clauses

Unsatisfiability-based Algorithms (Fu&Malik [SAT'06])

Partial MaxSAT Formula:

φ_h:		$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	
φ_s:	x_1	x_3	$x_2 \vee \bar{x}_1$	$\bar{x}_3 \vee x_1$

- Formula is satisfiable
- An optimal solution would be:
  - $\nu = \{x_1 = 1, x_2 = 0, x_3 = 0\}$
- This assignment unsatisfies 2 soft clauses
- How can this procedure be generalized to weighted?

(Manquinho et al. [SAT'09])

Unsatisfiability-based Algorithms (Manquinho et al. [SAT'09])

Weighted Partial MaxSAT Formula:

φ_h (Hard): $\bar{x}_2 \vee \bar{x}_1$ $\quad x_2 \vee \bar{x}_3$

φ_s (Soft): $(x_1, 2)$ $\quad (x_3, 3)$ $\quad (x_2 \vee \bar{x}_1, 1)$ $\quad (\bar{x}_3 \vee x_1, 1)$

Unsatisfiability-based Algorithms (Manquinho et al. [SAT'09])

Weighted Partial MaxSAT Formula:

φ_h (Hard): $\bar{x}_2 \vee \bar{x}_1$ $\quad x_2 \vee \bar{x}_3$

φ_s (Soft): $(x_1, 2)$ $\quad (x_3, 3)$ $\quad (x_2 \vee \bar{x}_1, 1)$ $\quad (\bar{x}_3 \vee x_1, 1)$

- Naive approach:
  - For each soft clause (ω, w) create w copies of weight 1
  - **Problem:** Does not scale when the size of the weights increase

Unsatisfiability-based Algorithms (Manquinho et al. [SAT'09])

Weighted Partial MaxSAT Formula:

φ_h (Hard): $\bar{x}_2 \vee \bar{x}_1$ $\quad x_2 \vee \bar{x}_3$

φ_s (Soft): $(x_1, 2)$ $\quad (x_3, 3)$ $\quad (x_2 \vee \bar{x}_1, 1)$ $\quad (\bar{x}_3 \vee x_1, 1)$

- **Solution:**
  - Create copies only when needed
  - Use the weight of the unsatisfiable core to split the soft clauses

Unsatisfiability-based Algorithms (Manquinho et al. [SAT'09])

Weighted Partial MaxSAT Formula:

$$\varphi_h: \quad \bar{x}_2 \vee \bar{x}_1 \quad x_2 \vee \bar{x}_3$$

$$\varphi_s: \quad (x_1, 2) \quad (x_3, 3) \quad (x_2 \vee \bar{x}_1, 1) \quad (\bar{x}_3 \vee x_1, 1)$$

- Formula is unsatisfiable

Unsatisfiability-based Algorithms (Manquinho et al. [SAT'09])

Weighted Partial MaxSAT Formula:

$$\varphi_h : \quad \bar{x}_2 \vee \bar{x}_1 \quad x_2 \vee \bar{x}_3$$

$$\varphi_s : \quad (x_1, 2) \quad (x_3, 3) \quad (x_2 \vee \bar{x}_1, 1) \quad (\bar{x}_3 \vee x_1, 1)$$

- Formula is unsatisfiable
- Identify an unsatisfiable core

Unsatisfiability-based Algorithms (Manquinho et al. [SAT'09])

Weighted Partial MaxSAT Formula:

$$\varphi_h: \quad \bar{x}_2 \vee \bar{x}_1 \quad x_2 \vee \bar{x}_3$$

$$\varphi_s: \quad (x_1 \vee r_1, 2) \quad \boxed{(x_3, 1)} \quad (x_2 \vee \bar{x}_1, 1) \quad (\bar{x}_3 \vee x_1, 1)$$

$$\boxed{(x_3, 2)}$$

- Core weight (c_w): 2 (smallest weight of the soft clauses in the core)
- Split soft clauses with weight larger than the core weight:
  - $(\omega, w) \rightarrow (\omega, w - c_w) \wedge (\omega, c_w)$

Unsatisfiability-based Algorithms (Manquinho et al. [SAT'09])

Weighted Partial MaxSAT Formula:

φ_h : $\bar{x}_2 \vee \bar{x}_1$ $\quad$ $x_2 \vee \bar{x}_3$ $\quad$ $\mathrm{CNF}(r_1 + r_2 \leq 1)$

φ_s : $(x_1 \vee r_1, 2)$ $\quad$ $(x_3, 1)$ $\quad$ $(x_2 \vee \bar{x}_1, 1)$ $\quad$ $(\bar{x}_3 \vee x_1, 1)$

$(x_3 \vee r_2, 2)$

- Core weight (c_w): 2 (smallest weight soft clauses in the core)
- Split soft clauses with weight larger than the core weight:
  - $(\omega, w) \rightarrow (\omega, w - c_w) \wedge (\omega, c_w)$
- Relax soft clauses with weight equal to c_w, add AtMost1 constraint

Unsatisfiability-based Algorithms (Manquinho et al. [SAT'09])

Weighted Partial MaxSAT Formula:

φ_h :	$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	$\mathrm{CNF}(r_1 + r_2 \leq 1)$	
φ_s :	$(x_1 \vee r_1, 2)$	$(x_3, 1)$	$(x_2 \vee \bar{x}_1, 1)$	$(\bar{x}_3 \vee x_1, 1)$
	$(x_3 \vee r_2, 2)$			

- Formula is unsatisfiable

Unsatisfiability-based Algorithms (Manquinho et al. [SAT'09])

Weighted Partial MaxSAT Formula:

φ_h :	$\bar{x}_2 \vee \bar{x}_1$	$x_2 \vee \bar{x}_3$	$\text{CNF}(r_1 + r_2 \leq 1)$	
φ_s :	$(x_1 \vee r_1, 2)$	$(x_3, 1)$	$(x_2 \vee \bar{x}_1, 1)$	$(\bar{x}_3 \vee x_1, 1)$
	$(x_3 \vee r_2, 2)$			

- Formula is unsatisfiable
- Identify unsatisfiable core

Unsatisfiability-based Algorithms (Manquinho et al. [SAT'09])

Weighted Partial MaxSAT Formula:

$$\begin{array}{lcccc} \varphi_h: & \bar{x}_2 \vee \bar{x}_1 & x_2 \vee \bar{x}_3 & \mathrm{CNF}(r_1 + r_2 \leq 1) & \boxed{\mathrm{CNF}(r_3 + r_4 \leq 1)} \\ \varphi_s: & (x_1 \vee r_1, 2) & \boxed{(x_3 \vee r_3, 1)} & (x_2 \vee \bar{x}_1, 1) & \boxed{(\bar{x}_3 \vee x_1 \vee r_4, 1)} \\ & (x_3 \vee r_2, 2) & & & \end{array}$$

- Formula is unsatisfiable
- Identify unsatisfiable core
- Relax unsatisfiable core

Unsatisfiability-based Algorithms (Manquinho et al. [SAT'09])

Weighted Partial MaxSAT Formula:

$$\varphi_h : \quad \bar{x}_2 \vee \bar{x}_1 \qquad x_2 \vee \bar{x}_3 \qquad \text{CNF}(r_1 + r_2 \leq 1) \qquad \text{CNF}(r_3 + r_4 \leq 1)$$

$$\varphi_s : \quad (x_1 \vee r_1, 2) \quad (x_3 \vee r_3, 1) \qquad (x_2 \vee \bar{x}_1, 1) \qquad (\bar{x}_3 \vee x_1 \vee r_4, 1)$$

$$(x_3 \vee r_2, 2)$$

- Formula is satisfiable
  - $\nu = \{x_1 = 0, x_2 = 1, x_3 = 1, r_1 = 1, r_2 = 0, r_3 = 0, r_4 = 1\}$
- **Optimal cost:** 3

Outline

- MaxSAT Algorithms:
  - Linear search algorithms
  - Unsatisfiability-based algorithms
- Incremental solving in MaxSAT:
  - Keep the state of the SAT solver between MaxSAT calls
- Partitioning in MaxSAT:
  - Use the structure of the problem to guide the search

Incremental MaxSAT algorithms

Martins et al. [CP'14]

- Non-incremental algorithms
  - Working formula is rebuilt at each SAT solver call
  - MaxSAT algorithms need to update constraints between calls to the SAT solver
  - For soundness reasons, SAT solvers only allow to add new variables and new constraints

Incremental MaxSAT algorithms

Martins et al. [CP'14]

- Non-incremental algorithms
  - Working formula is rebuilt at each SAT solver call
  - MaxSAT algorithms need to update constraints between calls to the SAT solver
  - For soundness reasons, SAT solvers only allow to add new variables and new constraints

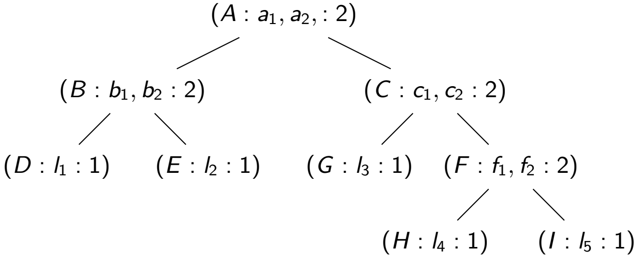
- **Goal:** Make Unsatisfiability-based algorithms incremental

Incremental MaxSAT algorithms

Martins et al. [CP'14]

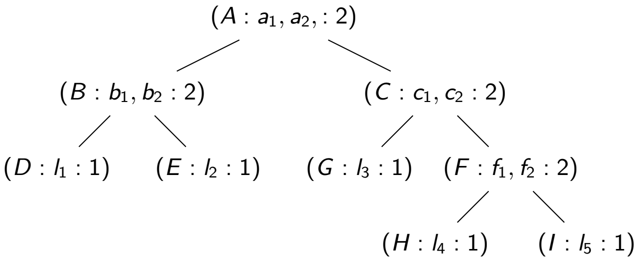
- SAT solver calls allow to specify **assumptions**:
  - Assumptions are literals that are set to true in the returned model
  - Assumptions can be changed between calls to the SAT solver
  - Soundness of the solver is maintained
- **Iterative encoding** for cardinality constraints:
  - Grow the encoding as needed
  - Adds new relaxation variables to encoding only when necessary
  - Use assumptions to fix the value of k for the current iteration

Iterative Encoding - Totalizer Encoding Martins et al. [CP'14]



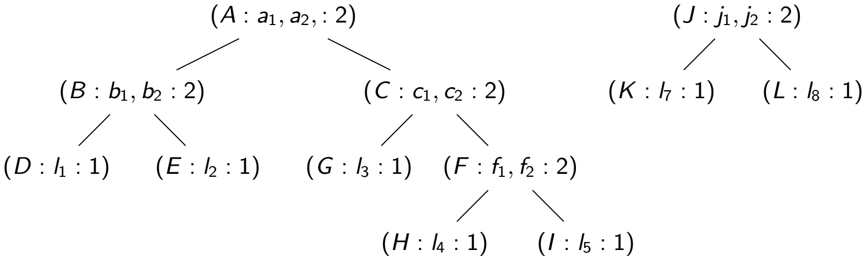
- Encoding of $l_1 + l_2 + l_3 + l_4 + l_5 \leq 1$

Iterative Encoding - Totalizer Encoding Martins et al. [CP'14]



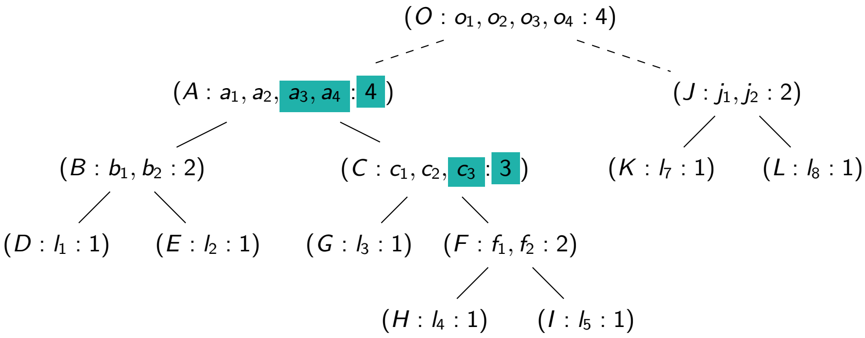
- Encoding of $l_1 + l_2 + l_3 + l_4 + l_5 \leq 1$
- Change it to $l_1 + l_2 + l_3 + l_4 + l_5 + \boxed{l_7 + l_8} \leq \boxed{3}$
  - Add two more literals (l_7 and l_8) to the left-hand side
  - Increase the right-hand side by 2

Iterative Encoding - Totalizer Encoding Martins et al. [CP'14]



- Encoding of $l_1 + l_2 + l_3 + l_4 + l_5 \leq 1$
- Change it to $l_1 + l_2 + l_3 + l_4 + l_5 + \boxed{l_7 + l_8} \leq \boxed{3}$
  - Add two more literals (l_7 and l_8) to the left-hand side
  - Increase the right-hand side by 2
- Extend the representation

Iterative Encoding - Totalizer Encoding Martins et al. [CP'14]



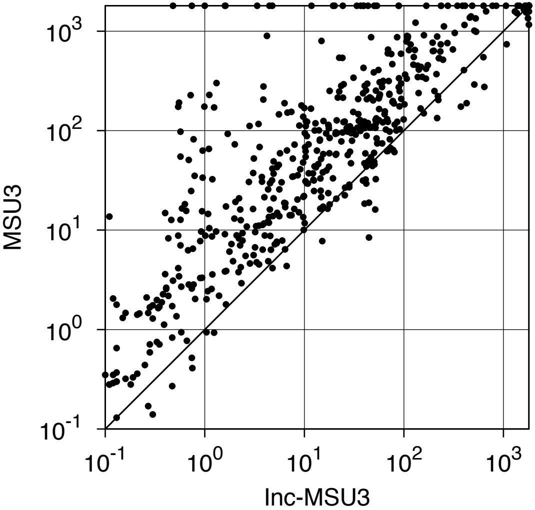
- Encoding of $l_1 + l_2 + l_3 + l_4 + l_5 \leq 1$
- Change it to $l_1 + l_2 + l_3 + l_4 + l_5 + \boxed{l_7 + l_8} \leq \boxed{3}$
  - Add two more literals (l_7 and l_8) to the left-hand side
  - Increase the right-hand side by 2
- Extend the representation

Experimental Results

- Open-WBO:
  - `http://sat.inesc-id.pt/open-wbo/`
  - Open-source MaxSAT solver
  - Non-incremental MSU3 and incremental MSU3
- Benchmarks: unweighted (55) and partial (568) MaxSAT instances from the industrial category of the MaxSAT Evaluation 2014
- AMD Opteron 6272 processors (2.3 GHz) running Fedora Core 18;
- Timeout: 1,800 seconds

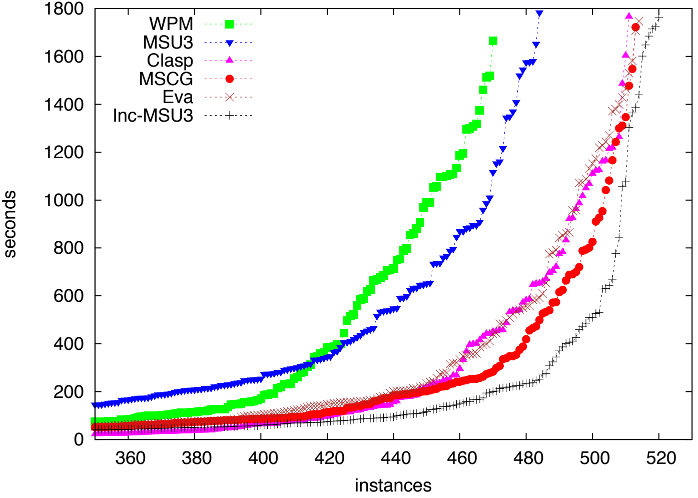
Experimental Results

- Non-incremental vs. Incremental MSU3



Experimental Results

- Running times of state-of-the-art MaxSAT solvers



Incremental MaxSAT Solving

- Iterative encoding can be used in other more sophisticated MaxSAT algorithms
- Experimental results clearly show the effectiveness of using incrementality
- Due to incrementality, Open-WBO won the best solver award for unweighted and partial MaxSAT at the MaxSAT Evaluation 2014

Outline

- MaxSAT Algorithms:
  - Linear search algorithms
  - Unsatisfiability-based algorithms
- Incremental solving in MaxSAT:
  - Keep the state of the SAT solver between MaxSAT calls
- Partitioning in MaxSAT:
  - Use the structure of the problem to guide the search

Partitioning in MaxSAT

- Unsatisfiability-based algorithms are very effective on industrial benchmarks
- However, performance is related with the unsatisfiable cores given by the SAT solver:
  - Some unsatisfiable cores may be unnecessarily large
  - **Solution:** Partition the soft clauses

Partitioning in MaxSAT

Martins et al. [ECAI'12]

(1) Partition the soft clauses

γ_1	γ_2	γ_3

Partitioning in MaxSAT

Martins et al. [ECAI'12]

(1) Partition the soft clauses

(2) Add a new partition to the formula

$\boxed{\gamma_1}$ $\boxed{\gamma_2}$ $\boxed{\gamma_3}$

Partitioning in MaxSAT

Martins et al. [ECAI'12]

(1) Partition the soft clauses

(2) Add a new partition to the formula

(3) While the formula is unsatisfiable:

- Relax unsatisfiable core

γ_1 γ_2 γ_3

Partitioning in MaxSAT

Martins et al. [ECAI'12]

(1) Partition the soft clauses

(2) Add a new partition to the formula

(3) While the formula is unsatisfiable:
   - Relax unsatisfiable core

(4) The formula is satisfiable:
   - If there are no more partitions:
     - ▷ Optimum found
   - Otherwise, **go back to 2**

γ_1	γ_2	γ_3

Partitioning in MaxSAT

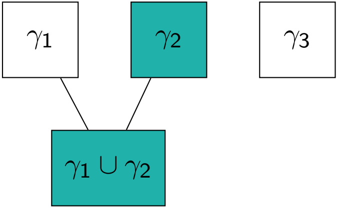
Martins et al. [ECAI'12]

(1) Partition the soft clauses

(2) Add a new partition to the formula

(3) While the formula is unsatisfiable:
   - Relax unsatisfiable core

(4) The formula is satisfiable:
   - If there are no more partitions:
     - ▷ Optimum found
   - Otherwise, go back to 2



Partitioning in MaxSAT

Martins et al. [ECAI'12]

(1) Partition the soft clauses

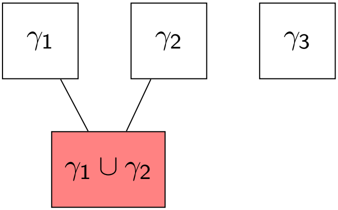
(2) Add a new partition to the formula

(3) While the formula is unsatisfiable:

- Relax unsatisfiable core

(4) The formula is satisfiable:

- If there are no more partitions:
  ▷ Optimum found
- Otherwise, go back to 2



Partitioning in MaxSAT

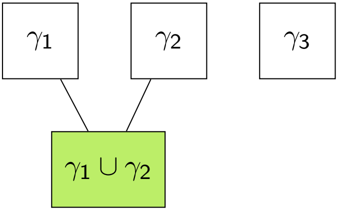
Martins et al. [ECAI'12]

(1) Partition the soft clauses

(2) Add a new partition to the formula

(3) While the formula is unsatisfiable:
  - Relax unsatisfiable core

(4) The formula is satisfiable:
  - If there are no more partitions:
    - ▷ Optimum found
  - Otherwise, **go back to 2**



Partitioning in MaxSAT

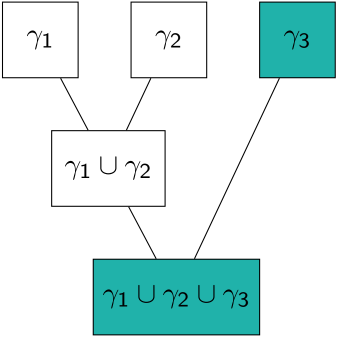
Martins et al. [ECAI'12]

(1) Partition the soft clauses

(2) Add a new partition to the formula

(3) While the formula is unsatisfiable:
   - Relax unsatisfiable core

(4) The formula is satisfiable:
   - If there are no more partitions:
     - ▷ Optimum found
   - Otherwise, go back to 2



Partitioning in MaxSAT

Martins et al. [ECAI'12]

(1) Partition the soft clauses

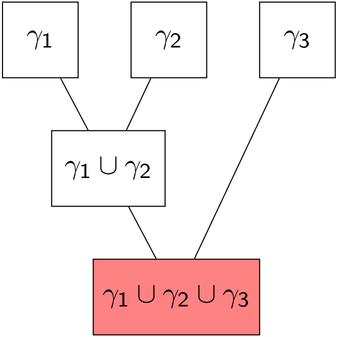
(2) Add a new partition to the formula

(3) While the formula is unsatisfiable:

- Relax unsatisfiable core

(4) The formula is satisfiable:

- If there are no more partitions:
  - ▷ Optimum found
- Otherwise, go back to 2



Partitioning in MaxSAT

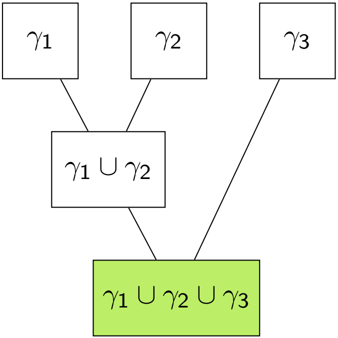
Martins et al. [ECAI'12]

(1) Partition the soft clauses

(2) Add a new partition to the formula

(3) While the formula is unsatisfiable:
- Relax unsatisfiable core

(4) The formula is satisfiable:
- If there are no more partitions:
  ▷ **Optimum found**
- Otherwise, go back to 2



How to partition the soft clauses?

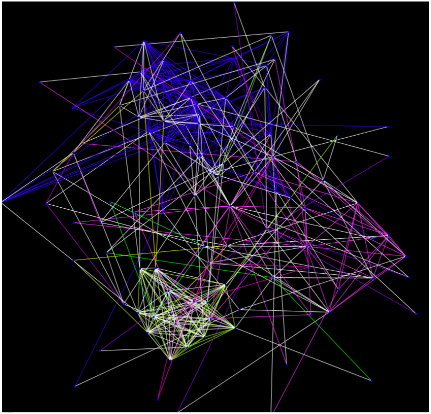
Use the structure of the problem to guide the search:

- Weighted partial MaxSAT: Martins et al. [ECAI'12]
  - Weight-based partitioning

- Partial MaxSAT: Martins et al. [SAT'13]
  - All soft clauses have weight 1
  - Graph-based partitioning:
    - Hypergraph
    - Variable Incidence Graph
    - Clause-Variable Incidence Graph

Exploiting the community structure!

SATGraf — https://bitbucket.org/znewsham/satgraf



(normalized-f20c10b_001_area_delay.wcnf)

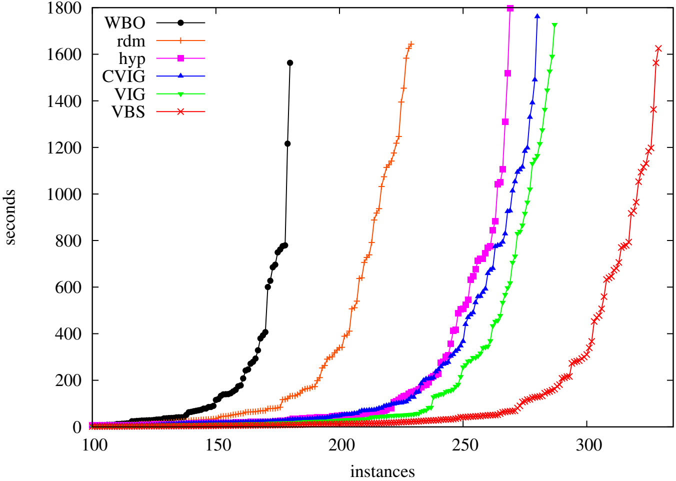
Experimental Results (Partial MaxSAT)

- Benchmarks:
  - 504 industrial partial MaxSAT instances

- Solvers:
  - `WBO`
  - `rdm` (Random partitioning – 16 partitions)
  - `hyp` (Hypergraph partitioning – 16 partitions)
  - `VIG` (Community partitioning – Variable Incidence Graph)
  - `CVIG` (Community partitioning – Clause-Variable Incidence Graph)
  - `VBS` (Virtual Best Solver)

Experimental Results (Partial MaxSAT)

- Running times of solvers for industrial partial MaxSAT instances



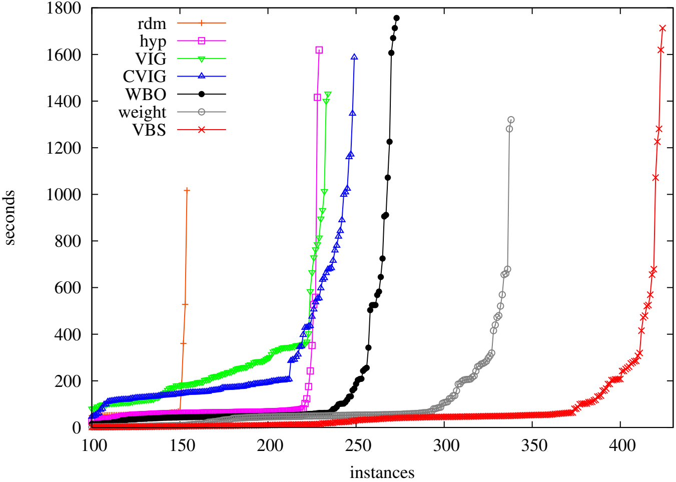
Experimental Results (Weighted Partial MaxSAT)

- Benchmarks:
  - 598 weighted partial MaxSAT instances

- Solvers:
  - `wbo`
  - `weight` (Weight-based partitioning)
  - `rdm` (Random partitioning – 16 partitions)
  - `hyp` (Hypergraph partitioning – 16 partitions)
  - `vig` (Community partitioning – Variable Incidence Graph)
  - `cvig` (Community partitioning – Clause-Variable Incidence Graph)
  - `vbs` (Virtual Best Solver)

Experimental Results (Weighted Partial MaxSAT)

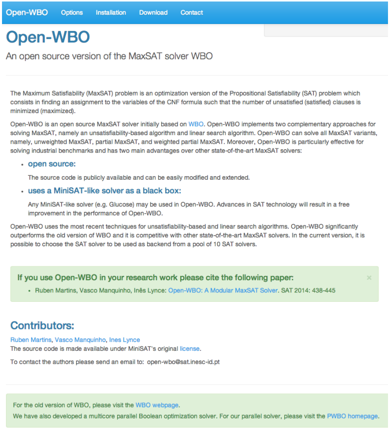
- Running times of solvers for weighted partial MaxSAT instances



Partitioning in MaxSAT

- Partitioning approaches outperform `WBO` on most instances:
  - Finds smaller unsatisfiable cores
- Weight-based partitioning is the best for weighted partial MaxSAT
- All algorithms contribute to the `VBS`:
  - Different graph-based partition methods solve different instances
  - Using the structure of the formula improves the partitioning
- Partitioning idea may be applied to other algorithms and fields!

Want to try MaxSAT solving?


Open-WBO Options Installation Download Contact

Open-WBO

An open source version of the MaxSAT solver WBO

The Maximum Satisfiability (MaxSAT) problem is an optimization version of the Propositional Satisfiability (SAT) problem which consists in finding an assignment to the variables of the CNF formula such that the number of unsatisfied (satisfied) clauses is minimized (maximized).

Open-WBO is an open source MaxSAT solver initially based on WBO. Open-WBO implements two complementary approaches for solving MaxSAT, namely an unsatisfiability-based algorithm and linear search algorithm. Open-WBO can solve all MaxSAT variants, namely, unweighted MaxSAT, partial MaxSAT, and weighted partial MaxSAT. Moreover, Open-WBO is particularly effective for solving industrial benchmarks and has two main advantages over other state-of-the-art MaxSAT solvers:

- **open source:**

  The source code is publicly available and can be easily modified and extended.
- **uses a MiniSAT-like solver as a black box:**

  Any MiniSAT-like solver (e.g. Glucose) may be used in Open-WBO. Advances in SAT technology will result in a free improvement in the performance of Open-WBO.

Open-WBO uses the most recent techniques for unsatisfiability-based and linear search algorithms. Open-WBO significantly outperforms the old version of WBO and it is competitive with other state-of-the-art MaxSAT solvers. In the current version, it is possible to choose the SAT solver to be used as backend from a pool of 10 SAT solvers.

If you use Open-WBO in your research work please cite the following paper:

- Ruben Martins, Vasco Manquinho, Inês Lynce: Open-WBO: A Modular MaxSAT Solver. SAT 2014: 438-445

Contributors:

Ruben Martins, Vasco Manquinho, Ines Lynce

The source code is made available under MiniSAT's original license.

To contact the authors please send an email to: open-wbo@sat.inesc-id.pt

For the old version of WBO, please visit the WBO webpage.

We have also developed a multicore parallel Boolean optimization solver. For our parallel solver, please visit the PWBO homepage.

Try out Open-WBO!

webpage:
http://sat.inesc-id.pt/open-wbo/

contact:
open-wbo@sat.inesc-id.pt

Comments and suggestions are welcome and will help to improve Open-WBO!

References

MaxSAT algorithms:

Z. Fu, S. Malik. On Solving the Partial MAX-SAT Problem. SAT 2006: 252-265.

V. Manquinho, J. Marques-Silva, J. Planes. Algorithms for Weighted Boolean Optimization. SAT 2009: 495-508

J. Marques-Silva, J. Planes. On using unsatisfiability for solving Maximum Satisfiability. Technical report 2007

R. Martins, S. Joshi, V. Manquinho, I. Lynce. Incremental Cardinality Constraints for MaxSAT. CP 2014: 531-548

R. Martins, V. Manquinho, I. Lynce. Open-WBO: A Modular MaxSAT Solver. SAT 2014: 438-445

R. Martins, V. Manquinho, I. Lynce. Community-Based Partitioning for MaxSAT Solving. SAT 2013: 182-191

R. Martins, V. Manquinho, I. Lynce. On Partitioning for Maximum Satisfiability. ECAI 2012: 913-914

References

Cardinality and Pseudo-Boolean Encodings:

C. Sinz. Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. CP 2005: 827-831

N. Manthey, T. Philipp, P. Steinke. A More Compact Translation of Pseudo-Boolean Constraints into CNF Such That Generalized Arc Consistency Is Maintained. KI 2014: 123-134

P. Steinke. A C++ Toolkit for Encoding Pseudo-Boolean Constraints into CNF. `http://tools.computational-logic.org/content/pblib.php`

Web pages of interest:

MaxSAT Evaluation: `http://www.maxsat.udl.cat/`

Open-WBO: `http://sat.inesc-id.pt/open-wbo/`