Maximum Satisfiability

Ruben Martins

Carnegie Mellon University

http://www.cs.cmu.edu/~mheule/15816-f19/

Automated Reasoning and Satisfiability, October 1, 2019

What is Boolean Satisfiability?

- **Fundamental problem** in Computer Science
  - The first problem to be proven NP-Complete
  - Has a wide range of applications

- Formula:
  - $\varphi = (\neg x_2 \vee \neg x_1) \wedge (x_2 \vee \neg x_3) \wedge (x_1) \wedge (x_3)$

- Boolean Satisfiability (SAT):
  - Is there an assignment of true or false values to variables such that φ evaluates to true?

Software Package Upgradeability Problem

2. utopia@utopia2: ~ (ssh)

```
utopia@utopia2:~$ sudo apt-get install bison++
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  flex-old
The following packages will be REMOVED:
  bison flex libfl-dev
The following NEW packages will be installed:
  bison++ flex-old
0 upgraded, 2 newly installed, 3 to remove and 334 not upgraded.
Need to get 507 kB of archives.
After this operation, 995 kB disk space will be freed.
Do you want to continue? [Y/n]
```

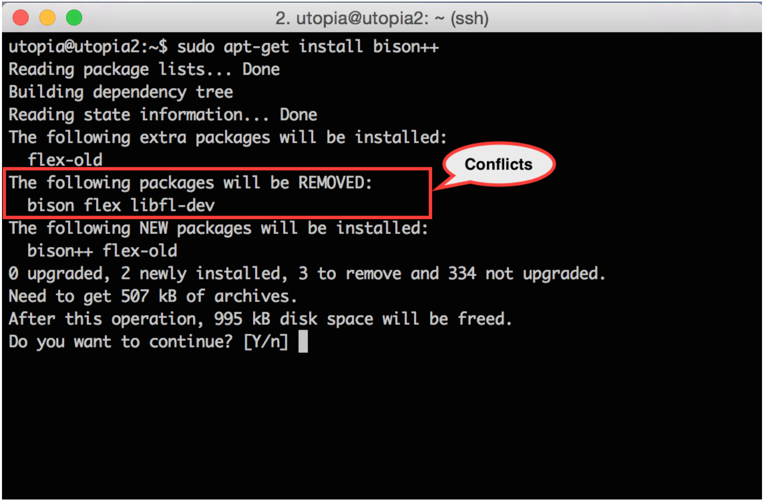
Software Package Upgradeability Problem

2. utopia@utopia2: ~ (ssh)

```
utopia@utopia2:~$ sudo apt-get install bison++
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  flex-old
The following packages will be REMOVED:
  bison flex libfl-dev
The following NEW packages will be installed:
  bison++ flex-old
0 upgraded, 2 newly installed, 3 to remove and 334 not upgraded.
Need to get 507 kB of archives.
After this operation, 995 kB disk space will be freed.
Do you want to continue? [Y/n]
```

Dependencies

Software Package Upgradeability Problem



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

NP Completeness

"I can't find an efficient algorithm, but neither can all these famous people."

NP Completeness

"I can't find an efficient algorithm, but neither can all these famous people."

- Giving up?
  - The problem is NP-hard, so let's develop heuristics or approximation algorithms.

NP Completeness

"I can't find an efficient algorithm, but neither can all these famous people."

- Giving up?
  - The problem is NP-hard, so let's develop heuristics or approximation algorithms.
- No! Current tools can find solutions for **very large** problems!

Software Package Upgradeability Problem as SAT

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

How can we encode this problem to Boolean Satisfiability?

Software Package Upgradeability Problem as SAT

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

How can we encode this problem to Boolean Satisfiability?

(Hint) Encode dependencies, conflicts, and installing all packages

Software Package Upgradeability Problem as SAT

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

How can we encode this problem to Boolean Satisfiability?

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

Software Package Upgradeability Problem as SAT

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

How can we encode this problem to Boolean Satisfiability?

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

Software Package Upgradeability Problem as SAT

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

How can we encode this problem to Boolean Satisfiability?

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

Software Package Upgradeability Problem as SAT

Formula φ:

Dependencies $\neg p_1 \vee p_2 \vee p_3$ $\neg p_2 \vee p_3$ $\neg p_3 \vee p_2$

Software Package Upgradeability Problem as SAT

Formula φ:

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

Software Package Upgradeability Problem as SAT

Formula φ:

Dependencies	$\neg p_1 \vee p_2 \vee p_3$	$\neg p_2 \vee p_3$	$\neg p_3 \vee p_2$	
Conflicts	$\neg p_4 \vee p_2$	$\neg p_4 \vee p_3$	$\neg p_1 \vee \neg p_4$	$\neg p_3 \vee \neg p_4$
Packages	p_1	p_2	p_3	p_4

Software Package Upgradeability Problem as SAT

Formula φ:

Dependencies	$\neg p_1 \vee p_2 \vee p_3$	$\neg p_2 \vee p_3$	$\neg p_3 \vee p_2$	
Conflicts	$\neg p_4 \vee p_2$	$\neg p_4 \vee p_3$	$\neg p_1 \vee \neg p_4$	$\neg p_3 \vee \neg p_4$
Packages	p_1	p_2	p_3	p_4

- $\varphi = (\neg p_1 \vee p_2 \vee p_3) \wedge (\neg p_2 \vee p_3) \wedge (\neg p_3 \vee p_2) \wedge (\neg p_4 \vee p_2) \wedge (\neg p_4 \vee p_3) \wedge (\neg p_1 \vee \neg p_4) \wedge (\neg p_3 \vee \neg p_4) \wedge (p_1) \wedge (p_2) \wedge (p_3) \wedge (p_4)$

Software Package Upgradeability Problem as SAT

Formula φ:

Dependencies	$\neg p_1 \vee p_2 \vee p_3$	$\neg p_2 \vee p_3$	$\neg p_3 \vee p_2$	
Conflicts	$\neg p_4 \vee p_2$	$\neg p_4 \vee p_3$	$\neg p_1 \vee \neg p_4$	$\neg p_3 \vee \neg p_4$
Packages	p_1	p_2	p_3	p_4

- Formula is unsatisfiable 
- Can you find an unsatisfiable subformula?

(Hint) There are several with 3 clauses!

Software Package Upgradeability Problem as SAT

Formula φ:

Dependencies	$\neg p_1 \lor p_2 \lor p_3$	$\neg p_2 \lor p_3$	$\neg p_3 \lor p_2$	
Conflicts	$\neg p_4 \lor p_2$	$\neg p_4 \lor p_3$	$\neg p_1 \lor \neg p_4$	$\neg p_3 \lor \neg p_4$
Packages	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):
  - Clauses in the formula are either **soft** or **hard**
  - Hard clauses: **must** be satisfied
    (e.g. conflicts, dependencies)
  - Soft clauses: **desirable** to be satisfied
    (e.g. package installation)
- **Goal**: Maximize number of satisfied soft clauses

How to encode Software Package Upgradeability?

Software Package Upgradeability problem as MaxSAT:

- What are the hard constraints?
  - **(Hint)** Dependencies, conflicts or installation packages?
- What are the soft constraints?
  - **(Hint)** Dependencies, conflicts or installation packages?

How to encode Software Package Upgradeability?

Software Package Upgradeability problem as MaxSAT:

- What are the hard constraints?
  - Dependencies and conflicts
- What are the soft constraints?
  - Installation of packages

Software Package Upgradeability Problem as MaxSAT

MaxSAT Formula:

φ_h (Hard): $\neg p_1 \vee p_2 \vee p_3$ $\quad \neg p_2 \vee p_3$ $\quad \neg p_3 \vee p_2$

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

φ_s (Soft): p_1 $\quad p_2$ $\quad p_3$ $\quad p_4$

- 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

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)

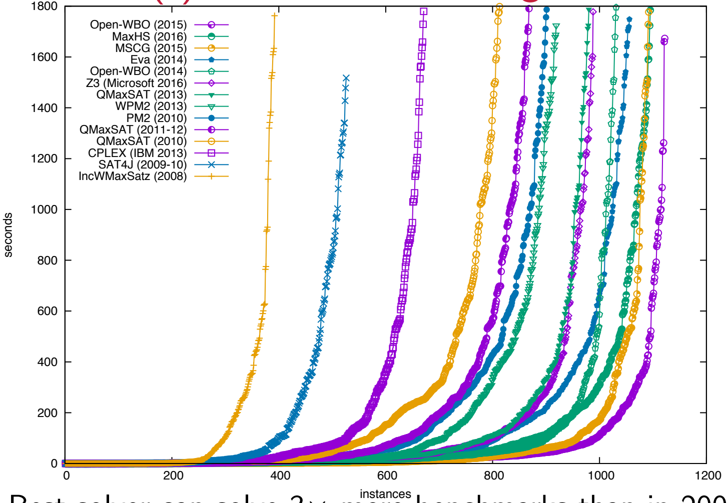
Why is MaxSAT Important?

- Many real-world applications can be encoded to MaxSAT:
  - Software package upgradeability
  - Error localization in C code

    
  - Haplotyping with pedigrees

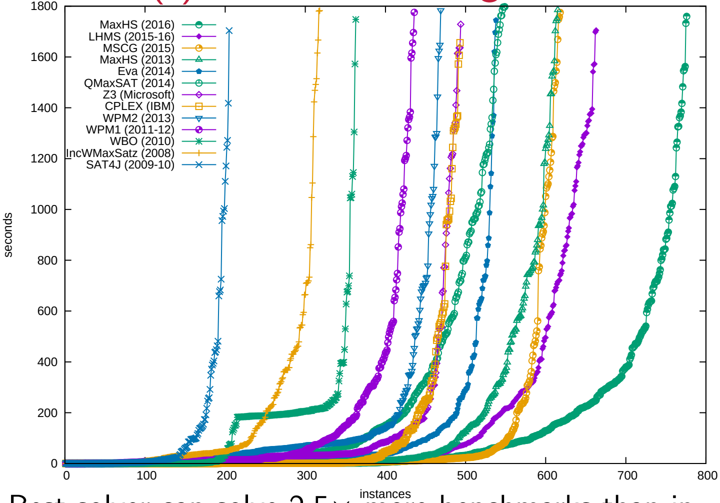
    
  - ...
- MaxSAT algorithms are **very effective** for solving real-word problems

The MaxSAT (r)evolution – Unweighted MaxSAT



- Best solver can solve 3× more benchmarks than in 2008!
- Better than tools like CPLEX (IBM) and Z3 (Microsoft)!

The MaxSAT (r)evolution – Weighted MaxSAT

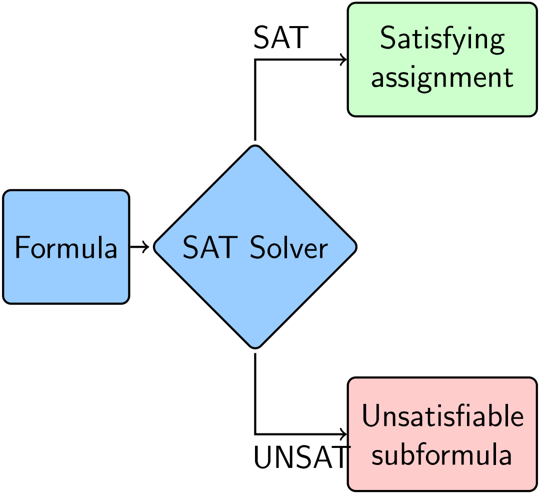


- Best solver can solve 2.5× more benchmarks than in 2008!
- Better than tools like CPLEX (IBM) and Z3 (Microsoft)!

Outline

- MaxSAT Algorithms:
  - **Upper bound search** on the number of unsatisfied soft clauses
  - **Lower bound search** on the number of unsatisfied soft clauses
- **Partitioning** in MaxSAT:
  - Use the structure of the problem to guide the search
- Using MaxSAT solvers

SAT Solvers


SAT
Satisfying assignment
Formula
SAT Solver
Unsatisfiable subformula
UNSAT

Satisfying assignment

Formula:

$$x_1 \qquad x_2 \lor \neg x_1 \qquad \neg x_3 \lor x_1 \qquad \neg x_3 \lor \neg x_1 \qquad x_2 \lor \neg x_3$$

- Satisfying assignment:
  - Assignment to the variables that evaluates the formula to true

Satisfying assignment

Formula:

$$x_1 \qquad x_2 \vee \neg x_1 \qquad \neg x_3 \vee x_1 \qquad \neg x_3 \vee \neg x_1 \qquad x_2 \vee \neg x_3$$

- Satisfying assignment:
  - Assignment to the variables that evaluates the formula to true
  - $\mu = \{x_1 = 1, x_2 = 1, x_3 = 0\}$

Unsatisfiable subformula

Formula:

$$x_1 \qquad x_3 \qquad x_2 \vee \neg x_1 \qquad \neg x_3 \vee x_1 \qquad \neg x_2 \vee \neg x_1 \qquad x_2 \vee \neg x_3$$

- Formula is unsatisfiable

Unsatisfiable subformula

Formula:

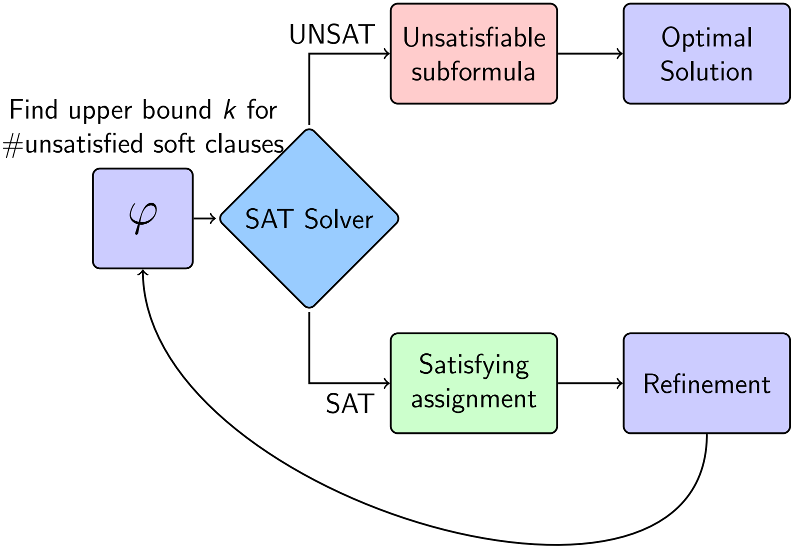
$x_1 \quad x_3 \quad x_2 \vee \neg x_1 \quad \neg x_3 \vee x_1 \quad \neg x_2 \vee \neg x_1 \quad x_2 \vee \neg x_3$

- Formula is unsatisfiable
- Unsatisfiable subformula (core):
  - $\varphi' \subseteq \varphi$, such that φ' is unsatisfiable

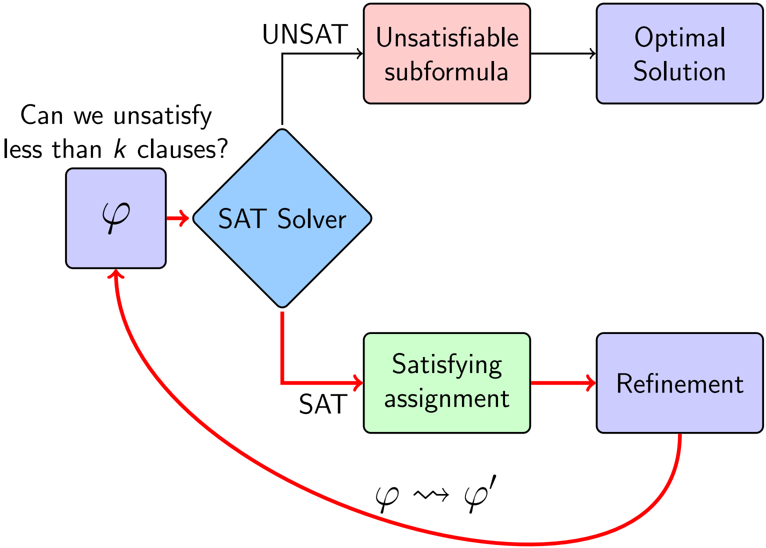
MaxSAT Algorithms

- MaxSAT algorithms build on SAT solver technology
- 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
  - Sinz, Totalizer, ...

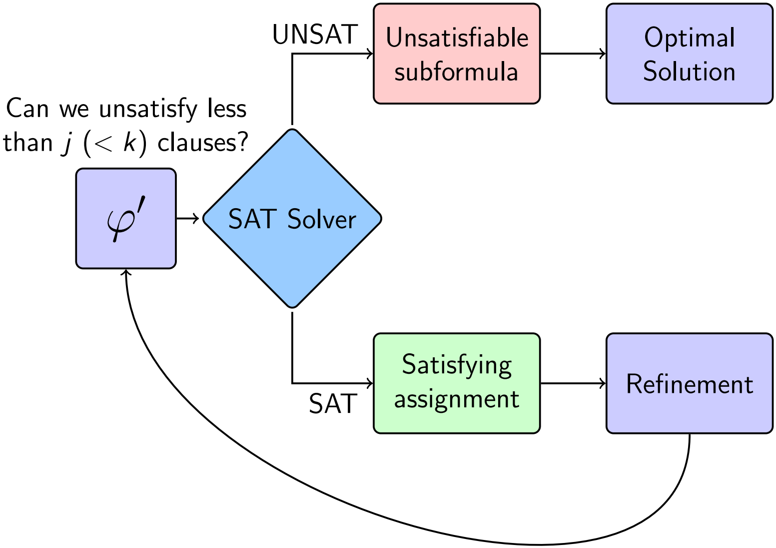
Upper Bound Search for MaxSAT



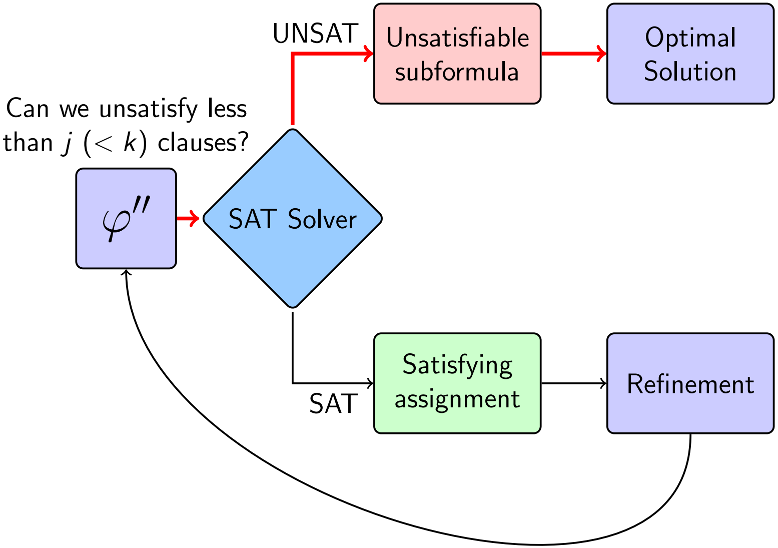
Upper Bound Search for MaxSAT



Upper Bound Search for MaxSAT



Upper Bound Search for MaxSAT



Linear Search Algorithms SAT-UNSAT

Partial MaxSAT Formula:

φ_h (Hard):		$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	
φ_s (Soft):	x_1	x_3	$x_2 \vee \neg x_1$	$\neg x_3 \vee x_1$

Linear Search Algorithms SAT-UNSAT

Partial MaxSAT Formula:

φ_h : $\neg x_2 \vee \neg x_1$ $\quad x_2 \vee \neg x_3$

φ_s : $x_1 \vee \boxed{r_1}$ $\quad x_3 \vee \boxed{r_2}$ $\quad x_2 \vee \neg x_1 \vee \boxed{r_3}$ $\quad \neg x_3 \vee x_1 \vee \boxed{r_4}$

- 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 :		$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	
φ_s :	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \neg x_1 \vee r_3$	$\neg x_3 \vee x_1 \vee r_4$

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

- 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 number of relaxation variables assigned to 1

Can we unsatisfy less than 2 soft clauses?

Partial MaxSAT Formula:

φ_h :		$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	
φ_s :	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \neg x_1 \vee r_3$	$\neg x_3 \vee x_1 \vee r_4$

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

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

Can we unsatisfy less than 2 soft clauses?

Partial MaxSAT Formula:

φ_h : $\neg x_2 \vee \neg x_1$ $\quad x_2 \vee \neg x_3$ $\quad \text{CNF}(\sum_{r_i \in V_R} r_i \leq 1)$

φ_s : $x_1 \vee r_1$ $\quad x_3 \vee r_2$ $\quad x_2 \vee \neg x_1 \vee r_3$ $\quad \neg x_3 \vee x_1 \vee r_4$

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

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

Can we unsatisfy less than 2 soft clauses? No!

Partial MaxSAT Formula:

φ_h :	$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	$\text{CNF}(\sum_{r_i \in V_R} r_i \leq 1)$	
φ_s :	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \neg x_1 \vee r_3$	$\neg x_3 \vee x_1 \vee r_4$

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

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

Can we unsatisfy less than 2 soft clauses? No!

Partial MaxSAT Formula:

φ_h:		$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	
φ_s:	x_1	x_3	$x_2 \vee \neg x_1$	$\neg x_3 \vee x_1$

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

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

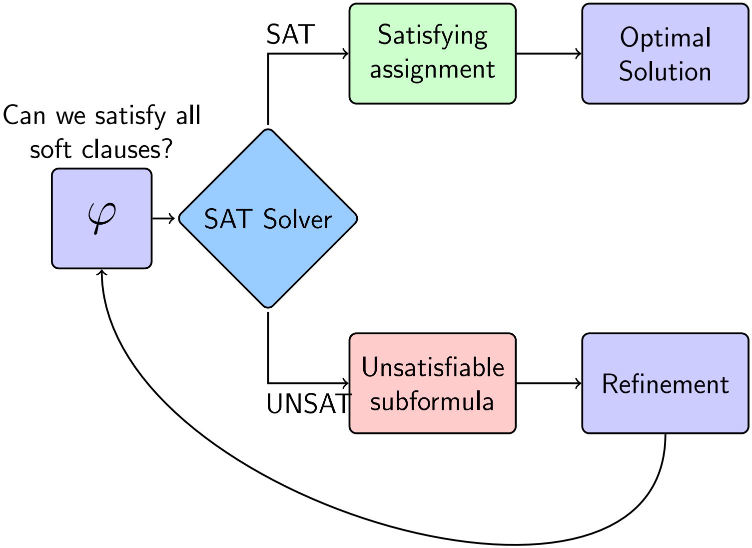
MaxSAT algorithms

- We have just seen a search on the **upper bound**
- What other kind of search can we do to find an optimal solution?

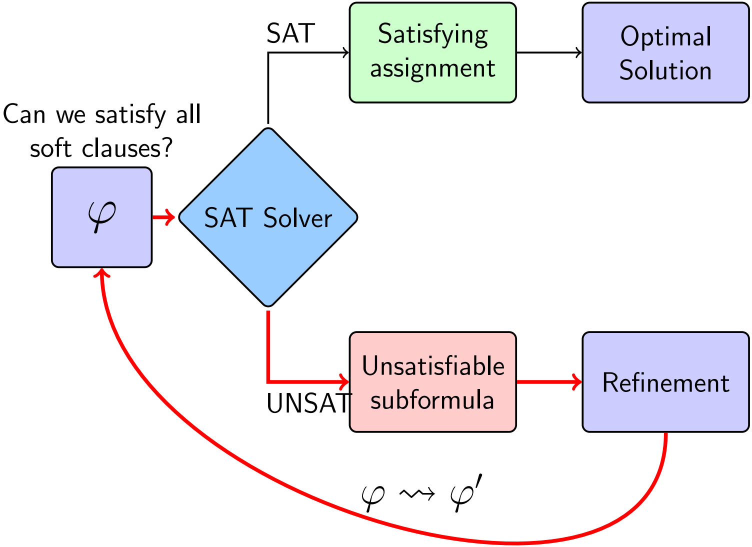
MaxSAT algorithms

- We have just seen a search on the **upper bound**
- What other kind of search can we do to find an optimal solution?
- What if we start searching from the **lower bound**?

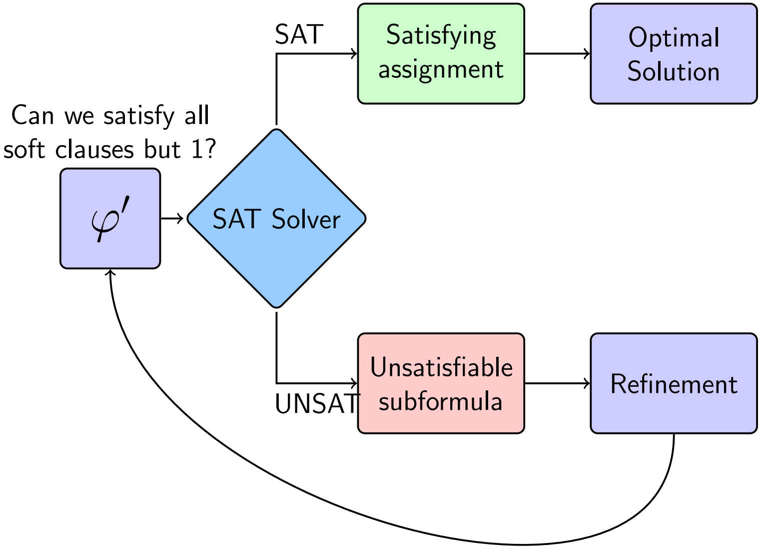
Lower Bound Search for MaxSAT



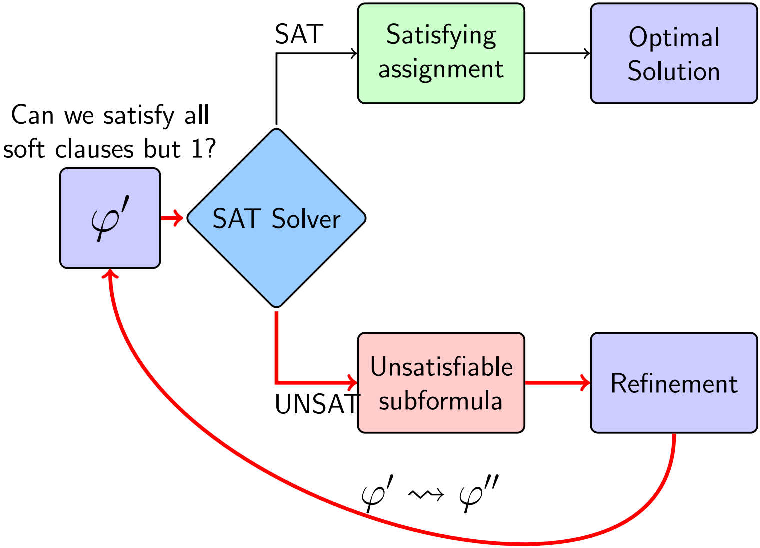
Lower Bound Search for MaxSAT



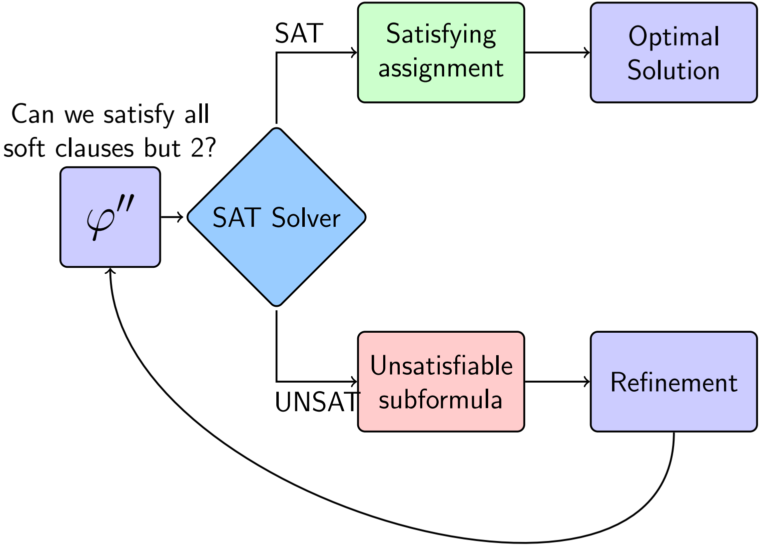
Lower Bound Search for MaxSAT



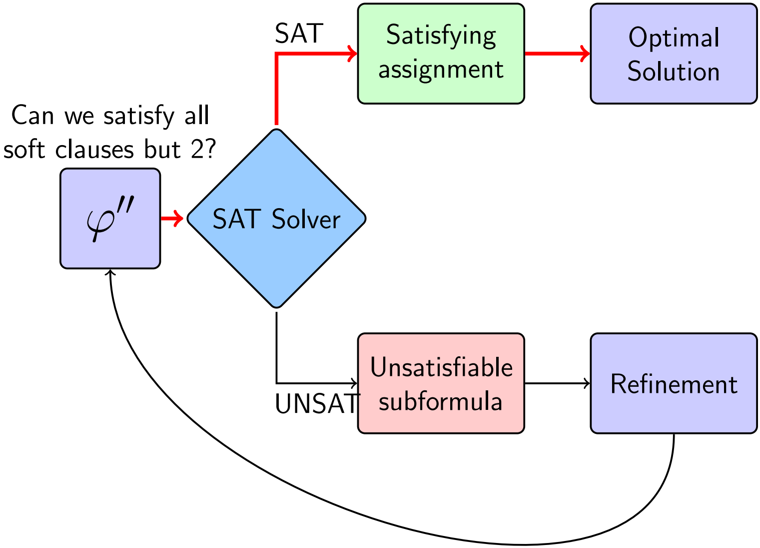
Lower Bound Search for MaxSAT



Lower Bound Search for MaxSAT



Lower Bound Search for MaxSAT



Linear Search Algorithms UNSAT-SAT

Partial MaxSAT Formula:

φ_h : $\neg x_2 \vee \neg x_1$ $\quad x_2 \vee \neg x_3$

φ_s : $x_1 \vee r_1$ $\quad x_3 \vee r_2$ $\quad x_2 \vee \neg x_1 \vee r_3$ $\quad \neg x_3 \vee x_1 \vee r_4$

- 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 $r_i = 0$

Can we satisfy all soft clauses?

Partial MaxSAT Formula:

φ_h : $\neg x_2 \vee \neg x_1$ $x_2 \vee \neg x_3$ $\mathrm{CNF}(\sum_{r_i \in V_R} r_i \leq 0)$

φ_s : $x_1 \vee r_1$ $x_3 \vee r_2$ $x_2 \vee \neg x_1 \vee r_3$ $\neg x_3 \vee x_1 \vee r_4$

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

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

Can we satisfy all soft clauses but 1?

Partial MaxSAT Formula:

φ_h :	$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	$\mathrm{CNF}(\sum_{r_i \in V_R} r_i \leq 0)$	
φ_s :	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \neg x_1 \vee r_3$	$\neg x_3 \vee x_1 \vee r_4$

- 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:
  - $\mathrm{CNF}(r_1 + r_2 + r_3 + r_4 \leq 1)$

Can we satisfy all soft clauses but 2?

Partial MaxSAT Formula:

φ_h :	$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	$\text{CNF}(\sum_{r_i \in V_R} r_i \leq 1)$	
φ_s :	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \neg x_1 \vee r_3$	$\neg x_3 \vee x_1 \vee r_4$

- 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:
  - $\text{CNF}(r_1 + r_2 + r_3 + r_4 \leq 2)$

Can we satisfy all soft clauses but 2? Yes!

Partial MaxSAT Formula:

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

- 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

Unsatisfiability-based Algorithms

- What are the problems of this algorithm?
  (Hint) Number of relaxation variables? Size of the cardinality constraint? Other?

Unsatisfiability-based Algorithms

- What are the problems of this algorithm?
  (Hint) Number of relaxation variables? Size of the cardinality constraint? Other?

- We relax all soft clauses!
- The cardinality constraint contain as many literals as we have soft clauses!
- Can we do better?

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h (Hard):		$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	
φ_s (Soft):	x_1	x_3	$x_2 \vee \neg x_1$	$\neg x_3 \vee x_1$

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h:		$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	
φ_s:	x_1	x_3	$x_2 \vee \neg x_1$	$\neg x_3 \vee x_1$

- Formula is unsatisfiable

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h:		$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	
φ_s:	x_1	x_3	$x_2 \vee \neg x_1$	$\neg x_3 \vee x_1$

- Formula is unsatisfiable
- Identify an unsatisfiable core

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h:	$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	$\mathrm{CNF}(r_1 + r_2 \leq 1)$	
φ_s:	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \neg x_1$	$\neg x_3 \vee x_1$

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

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h:	$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	$\text{CNF}(r_1 + r_2 \leq 1)$	
φ_s:	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \neg x_1$	$\neg x_3 \vee x_1$

- Formula is unsatisfiable

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h:	$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	$\text{CNF}(r_1 + r_2 \leq 1)$	
φ_s:	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \neg x_1$	$\neg x_3 \vee x_1$

- Formula is unsatisfiable
- Identify an unsatisfiable core

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h: $\neg x_2 \vee \neg x_1$ $\quad x_2 \vee \neg x_3$ $\quad \boxed{\text{CNF}(r_1 + \ldots + r_4 \leq 2)}$

φ_s: $x_1 \vee r_1$ $\quad x_3 \vee r_2$ $\quad x_2 \vee \neg x_1 \vee \boxed{r_3}$ $\quad \neg x_3 \vee x_1 \vee \boxed{r_4}$

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

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h : $\neg x_2 \vee \neg x_1$ $\quad x_2 \vee \neg x_3$ $\quad \text{CNF}(r_1 + \ldots + r_4 \leq 2)$

φ_s : $x_1 \vee r_1$ $\quad x_3 \vee r_2$ $\quad x_2 \vee \neg x_1 \vee r_3$ $\quad \neg x_3 \vee x_1 \vee r_4$

- 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

Unsatisfiability-based Algorithms

- What are the problems of this algorithm?
  (Hint) Number of relaxation variables? Size of the cardinality constraint? Other?

Unsatisfiability-based Algorithms

- What are the problems of this algorithm?
  (Hint) Number of relaxation variables? Size of the cardinality constraint? Other?
- We must translate cardinality constraints into CNF!
- If the number of literals is large than we may generate a **very large** formula!
- Can we do better?

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h (Hard):		$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	
φ_s (Soft):	x_1	x_3	$x_2 \vee \neg x_1$	$\neg x_3 \vee x_1$

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h:		$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	
φ_s:	x_1	x_3	$x_2 \vee \neg x_1$	$\neg x_3 \vee x_1$

- Formula is unsatisfiable

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h:		$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	
φ_s:	x_1	x_3	$x_2 \vee \neg x_1$	$\neg x_3 \vee x_1$

- Formula is unsatisfiable
- Identify an unsatisfiable core

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h: $\neg x_2 \vee \neg x_1$ $\quad x_2 \vee \neg x_3$ $\quad \boxed{\text{CNF}(r_1 + r_2 \leq 1)}$

φ_s: $x_1 \vee \boxed{r_1}$ $\quad x_3 \vee \boxed{r_2}$ $\quad x_2 \vee \neg x_1$ $\quad \neg x_3 \vee x_1$

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

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h:	$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	$\text{CNF}(r_1 + r_2 \leq 1)$	
φ_s:	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \neg x_1$	$\neg x_3 \vee x_1$

- Formula is unsatisfiable

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h:	$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	$\text{CNF}(r_1 + r_2 \leq 1)$	
φ_s:	$x_1 \vee r_1$	$x_3 \vee r_2$	$x_2 \vee \neg x_1$	$\neg x_3 \vee x_1$

- Formula is unsatisfiable
- Identify an unsatisfiable core

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h: $\neg x_2 \vee \neg x_1$ $\quad x_2 \vee \neg x_3$ $\quad \mathrm{CNF}(r_1 + r_2 \leq 1)$ $\quad \mathrm{CNF}(r_3 + \ldots + r_6 \leq 1)$

φ_s: $x_1 \vee r_1 \vee r_3$ $\quad x_3 \vee r_2 \vee r_4$ $\quad x_2 \vee \neg x_1 \vee r_5$ $\quad \neg x_3 \vee x_1 \vee r_6$

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

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

$$\varphi_h: \quad \neg x_2 \vee \neg x_1 \quad x_2 \vee \neg x_3 \quad \text{CNF}(r_1 + r_2 \leq 1) \quad \text{CNF}(r_3 + \ldots + r_6 \leq 1)$$

$$\varphi_s: \quad x_1 \vee r_1 \vee r_3 \quad x_3 \vee r_2 \vee r_4 \quad x_2 \vee \neg x_1 \vee r_5 \quad \neg x_3 \vee x_1 \vee r_6$$

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

Unsatisfiability-based Algorithms

Partial MaxSAT Formula:

φ_h:		$\neg x_2 \vee \neg x_1$	$x_2 \vee \neg x_3$	
φ_s:	x_1	x_3	$x_2 \vee \neg x_1$	$\neg x_3 \vee x_1$

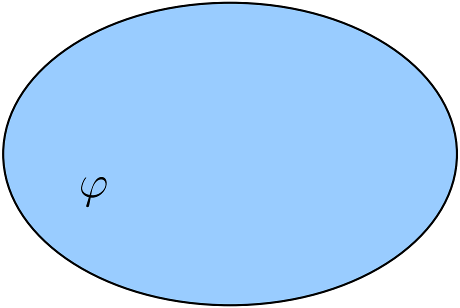
- Formula is satisfiable
- An optimal solution would be:
  - $\nu = \{x_1 = 1, x_2 = 0, x_3 = 0\}$
- This assignment unsatisfies 2 soft clauses

Challenges for Unsatisfiability-based MaxSAT Algorithms

- Unsatisfiable cores found by the SAT solver are **not minimal**

Challenges for Unsatisfiability-based MaxSAT Algorithms

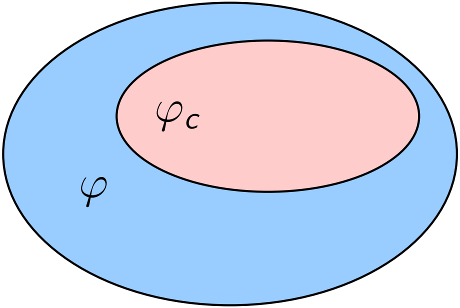
- Unsatisfiable cores found by the SAT solver are **not minimal**



Formula φ

Challenges for Unsatisfiability-based MaxSAT Algorithms

- Unsatisfiable cores found by the SAT solver are **not minimal**

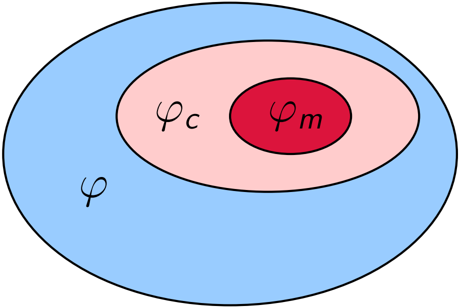


Formula φ

Unsatisfiable core φ_c

Challenges for Unsatisfiability-based MaxSAT Algorithms

- Unsatisfiable cores found by the SAT solver are **not minimal**



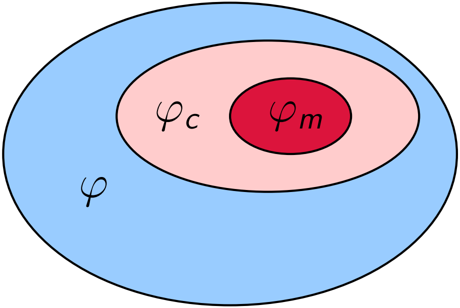
Formula φ

Unsatisfiable core φ_c

Minimal core φ_m

Challenges for Unsatisfiability-based MaxSAT Algorithms

- Unsatisfiable cores found by the SAT solver are **not minimal**



Formula φ

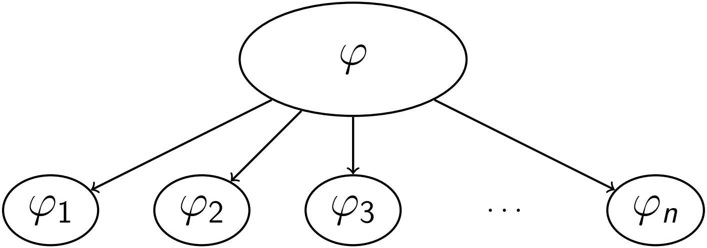
Unsatisfiable core φ_c

Minimal core φ_m

- **Minimizing** unsatisfiable cores is **computationally hard**

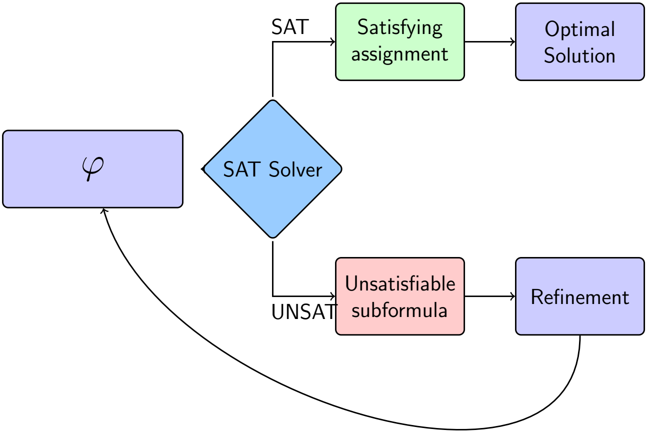
Partitioning in MaxSAT

- Partitioning in MaxSAT:
  - Partition the soft clauses into disjoint sets
  - Iteratively increase the size of the MaxSAT formula

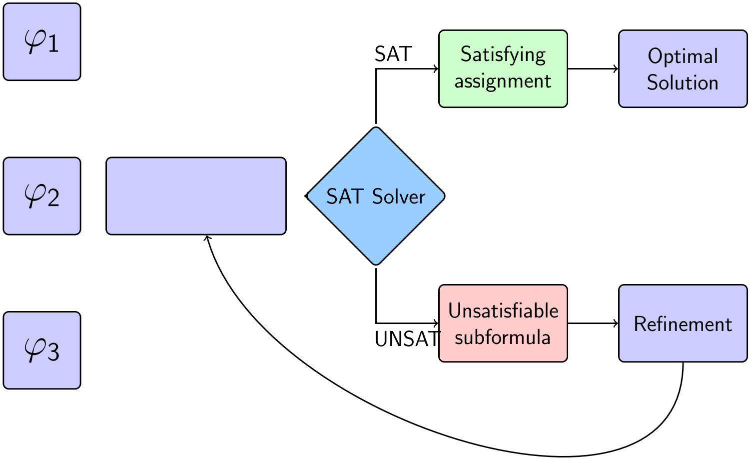


- Advantages:
  - **Easier formulas** for the SAT solver
  - **Smaller unsatisfiable cores** at each iteration

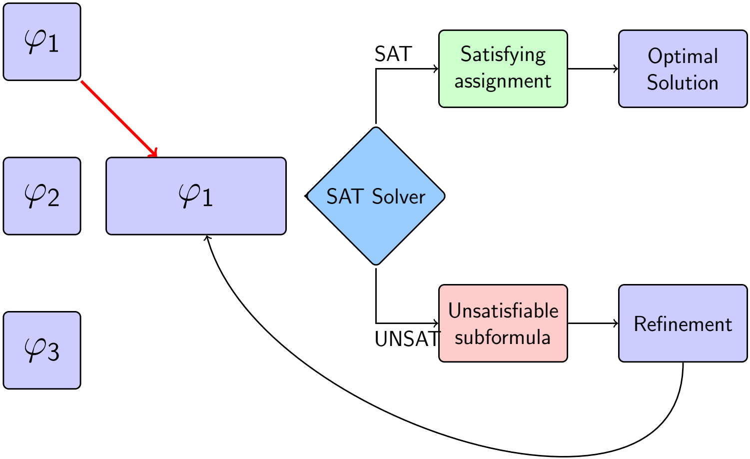
Framework for Partitioning-based MaxSAT Algorithms



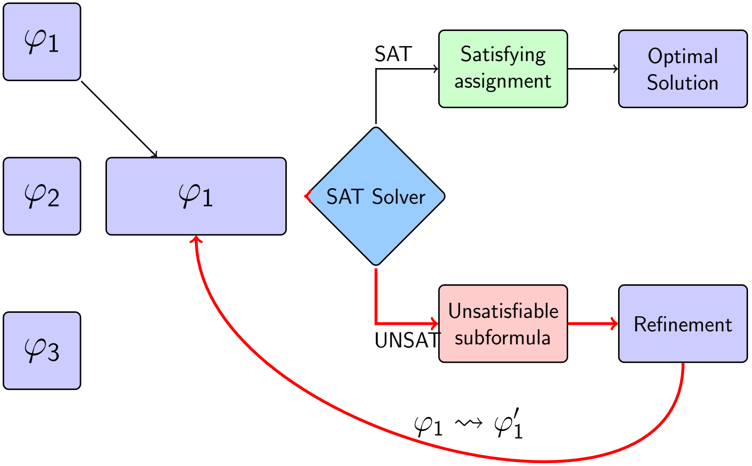
Framework for Partitioning-based MaxSAT Algorithms



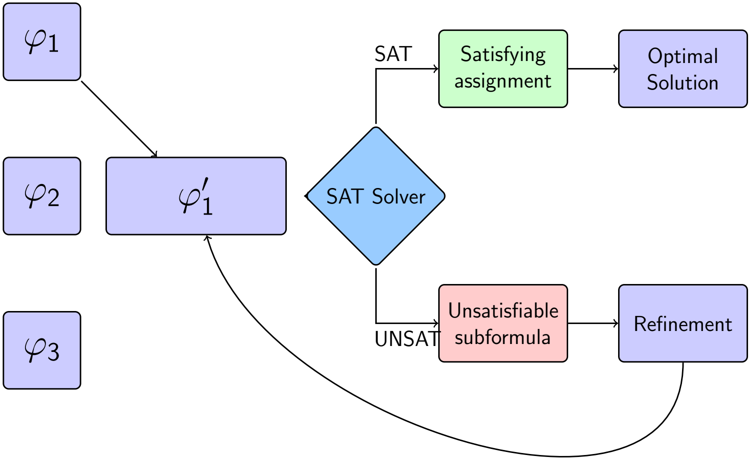
Framework for Partitioning-based MaxSAT Algorithms



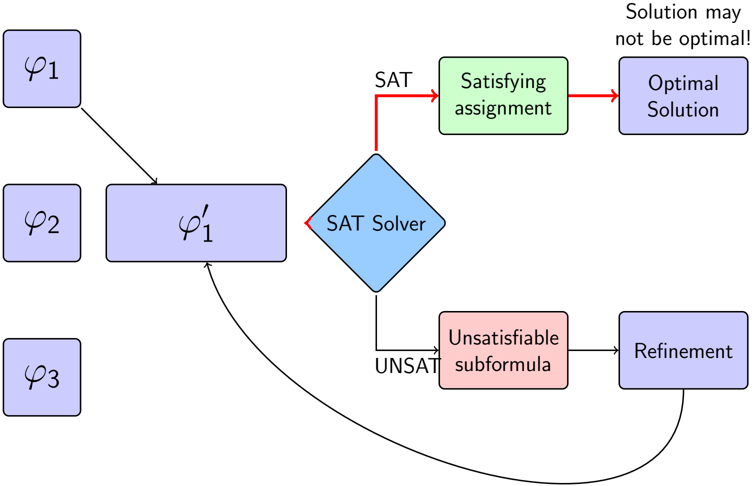
Framework for Partitioning-based MaxSAT Algorithms



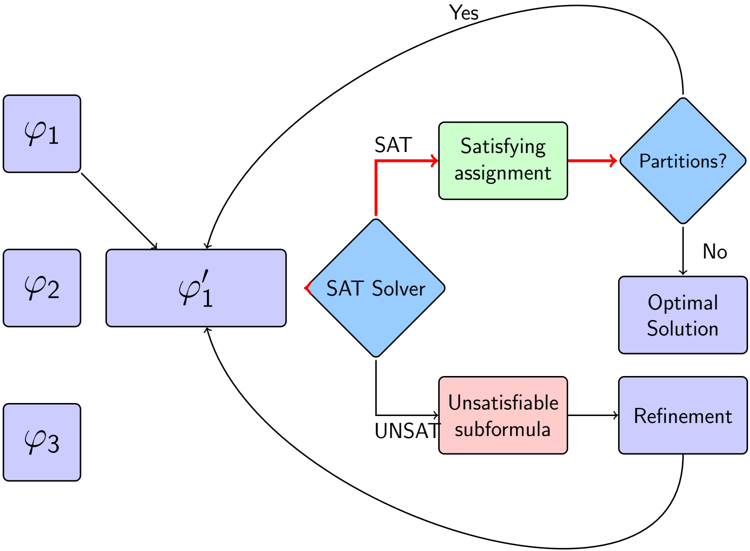
Framework for Partitioning-based MaxSAT Algorithms



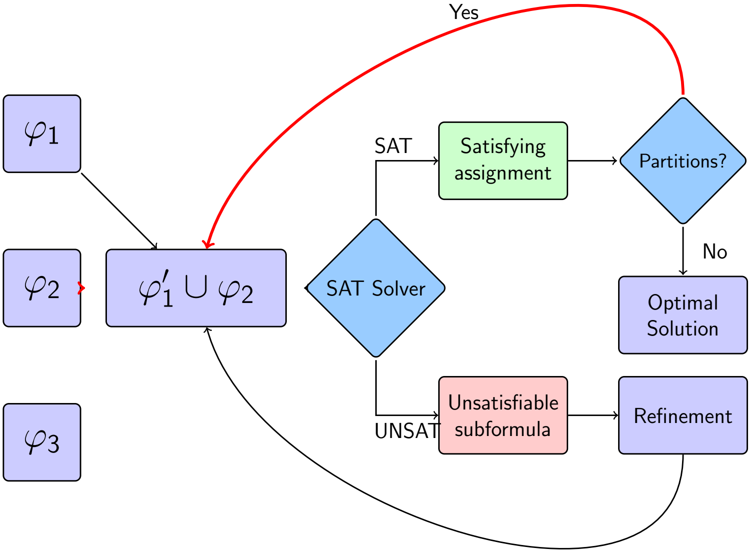
Framework for Partitioning-based MaxSAT Algorithms



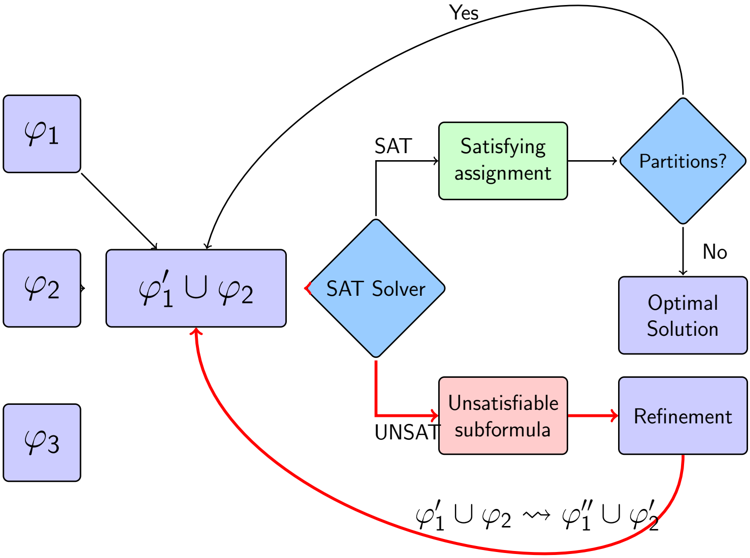
Framework for Partitioning-based MaxSAT Algorithms



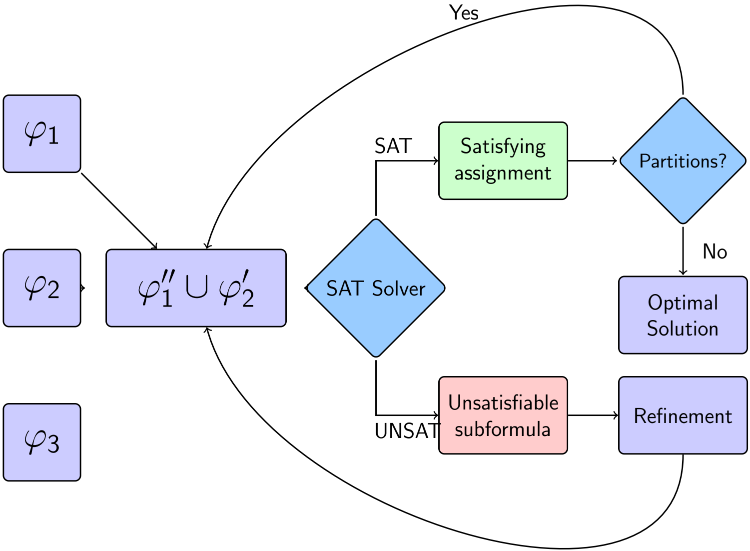
Framework for Partitioning-based MaxSAT Algorithms



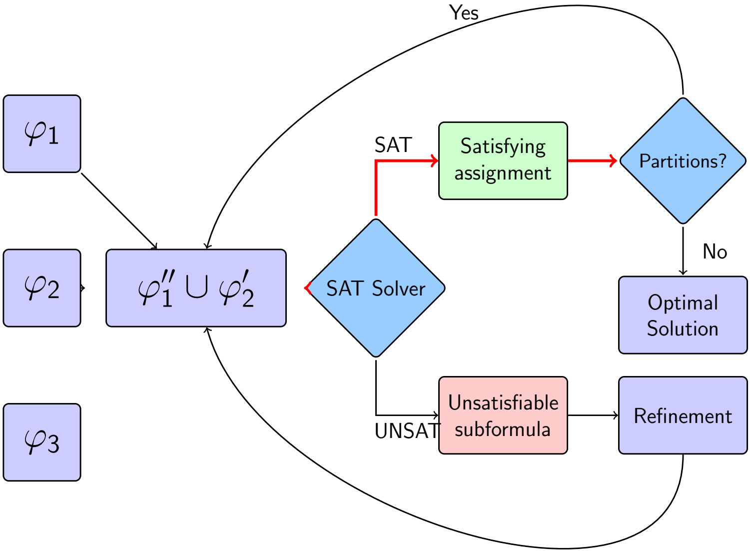
Framework for Partitioning-based MaxSAT Algorithms



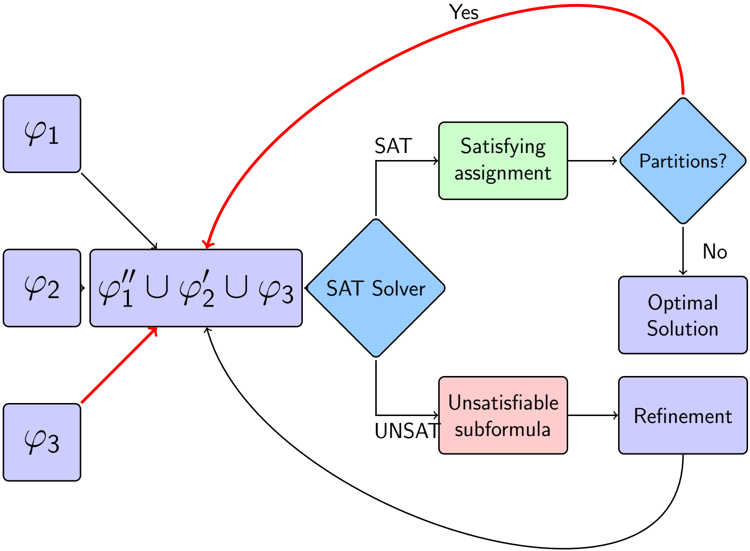
Framework for Partitioning-based MaxSAT Algorithms



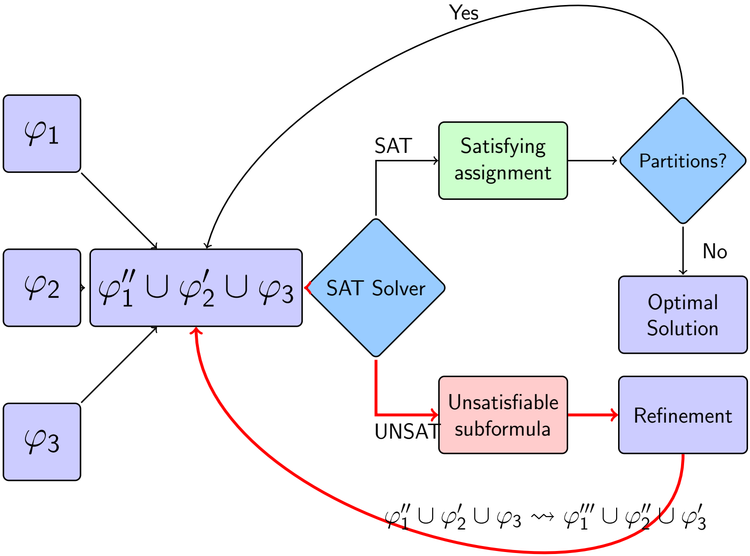
Framework for Partitioning-based MaxSAT Algorithms



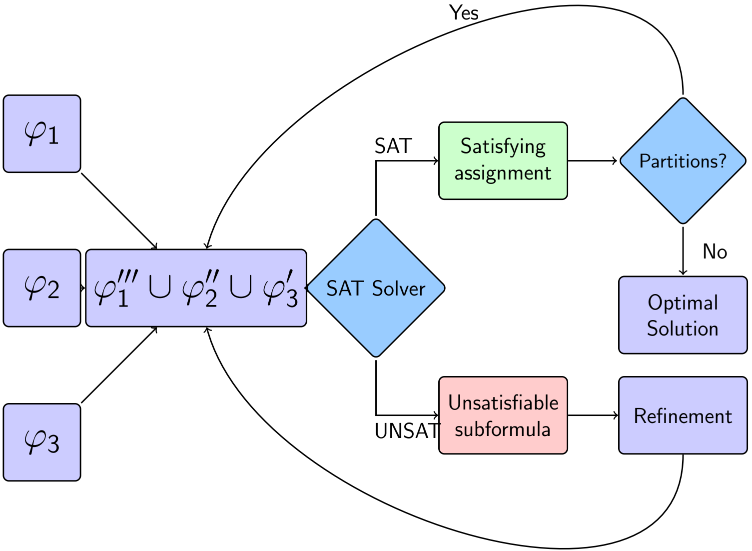
Framework for Partitioning-based MaxSAT Algorithms



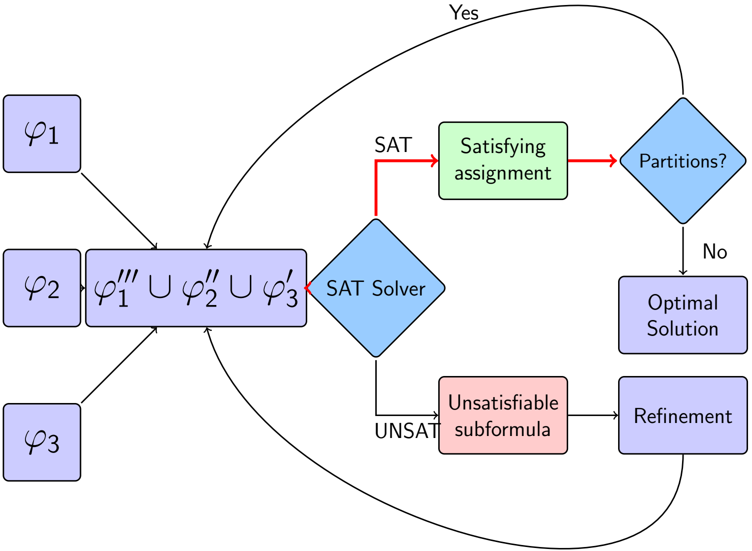
Framework for Partitioning-based MaxSAT Algorithms



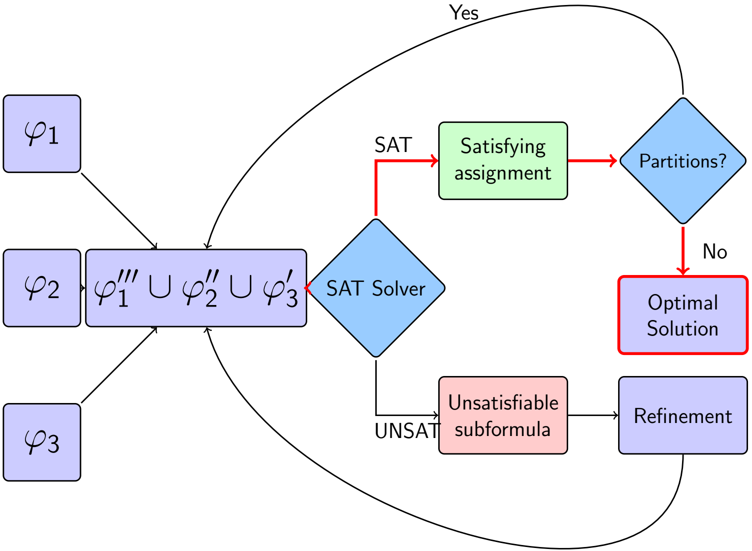
Framework for Partitioning-based MaxSAT Algorithms



Framework for Partitioning-based MaxSAT Algorithms

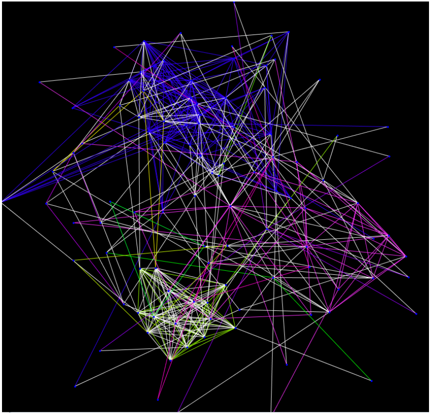


Framework for Partitioning-based MaxSAT Algorithms



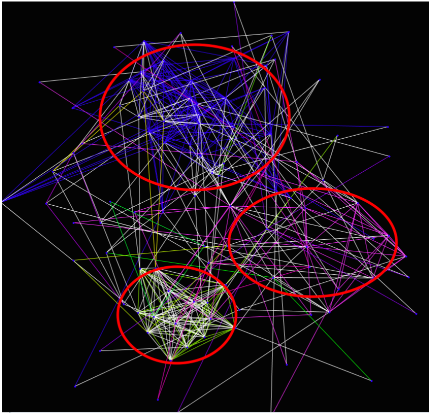
How to Partition Soft Clauses?

- **Graph representation** of the MaxSAT formula:
  - Vertices: Variables
  - Edges: Between variables that appear in the same clause



How to Partition Soft Clauses?

- **Graph representation** of the MaxSAT formula:
  - Vertices: Variables
  - Edges: Between variables that appear in the same clause



Graph representations for MaxSAT

- There are many ways to represent MaxSAT as a graph:
  - Clause-Variable Incidence Graph (CVIG)
  - Variable Incidence Graph (VIG)
  - Hypergraph
  - Resolution Graph
  - . . .

Graph representations for MaxSAT

- There are many ways to represent MaxSAT as a graph:
  - Clause-Variable Incidence Graph (CVIG)
  - Variable Incidence Graph (VIG)
  - Hypergraph
  - **Resolution Graph**
  - ...

MaxSAT Formulas as Resolution-based Graphs

- MaxSAT solvers rely on the identification of **unsatisfiable cores**
- How can we capture sets of clauses that are closely related and are likely to result in unsatisfiable cores?
  - Represent MaxSAT formulas as **resolution graphs**!
  - Resolution graphs are based on the resolution rule

MaxSAT Formulas as Resolution-based Graphs

- MaxSAT solvers rely on the identification of **unsatisfiable cores**
- How can we capture sets of clauses that are closely related and are likely to result in unsatisfiable cores?
  - Represent MaxSAT formulas as **resolution graphs**!
  - Resolution graphs are based on the resolution rule
- Example of the resolution rule:

$$\frac{(x_1 \vee x_2) \quad (\neg x_2 \vee x_3)}{}$$

MaxSAT Formulas as Resolution-based Graphs

- MaxSAT solvers rely on the identification of **unsatisfiable cores**
- How can we capture sets of clauses that are closely related and are likely to result in unsatisfiable cores?
  - Represent MaxSAT formulas as **resolution graphs**!
  - Resolution graphs are based on the resolution rule
- Example of the resolution rule:

$$\frac{(x_1 \vee x_2) \quad (\neg x_2 \vee x_3)}{(x_1 \vee x_3)}$$

MaxSAT Formulas as Resolution-based Graphs

- Vertices: Represent each clause in the graph
- Edges: There is an edge between two vertices if you can apply the **resolution rule** between the corresponding clauses

MaxSAT Formulas as Resolution-based Graphs

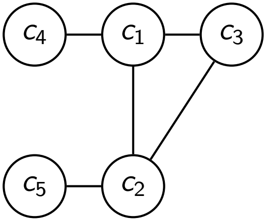
- Vertices: Represent each clause in the graph
- Edges: There is an edge between two vertices if you can apply the **resolution rule** between the corresponding clauses

Hard clauses:	Soft clauses:
$c_1 = x_1 \lor x_2$	$c_4 = \neg x_1$
$c_2 = \neg x_2 \lor x_3$	$c_5 = \neg x_3$
$c_3 = \neg x_1 \lor \neg x_3$	

MaxSAT Formulas as Resolution-based Graphs

- Vertices: Represent each clause in the graph
- Edges: There is an edge between two vertices if you can apply the **resolution rule** between the corresponding clauses

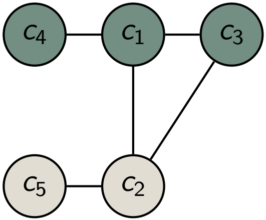
Hard clauses:	Soft clauses:
$c_1 = x_1 \vee x_2$	$c_4 = \neg x_1$
$c_2 = \neg x_2 \vee x_3$	$c_5 = \neg x_3$
$c_3 = \neg x_1 \vee \neg x_3$	



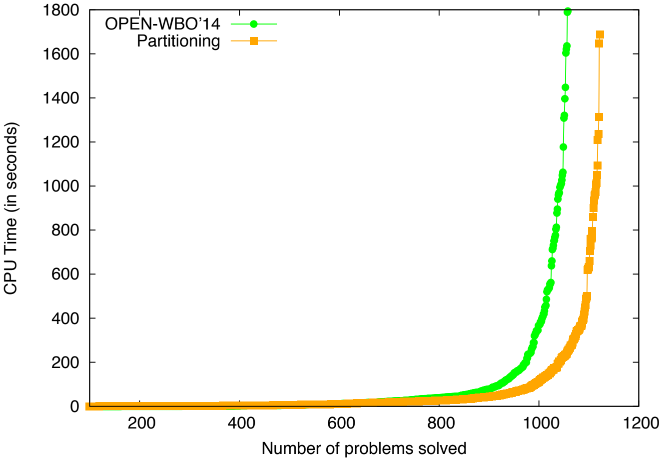
MaxSAT Formulas as Resolution-based Graphs

- Vertices: Represent each clause in the graph
- Edges: There is an edge between two vertices if you can apply the **resolution rule** between the corresponding clauses

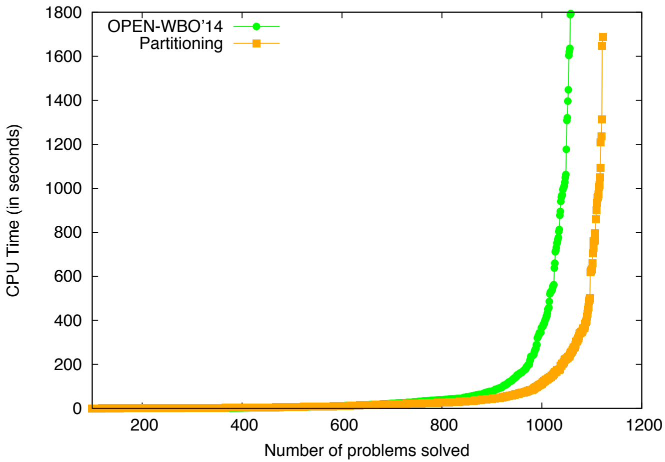
Hard clauses:	Soft clauses:
$c_1 = x_1 \vee x_2$	$c_4 = \neg x_1$
$c_2 = \neg x_2 \vee x_3$	$c_5 = \neg x_3$
$c_3 = \neg x_1 \vee \neg x_3$	



Impact of Partitioning in the MaxSAT Solving



Impact of Partitioning in the MaxSAT Solving



- The techniques in Open-WBO have been **adopted** by other state-of-the-art MaxSAT solvers

Want to try MaxSAT solving?

- Java:
  - **SAT4J**
  - `http://www.sat4j.org/`
- Python:
  - **RC2**
  - Best solver in 2018 and 2019!
  - SAT solvers written in C++
  - `https://pysathq.github.io`

Want to try MaxSAT solving?

- C++:
  - **MaxHS**
  - One of the best solvers for weighted problems!
  - Combines SAT and MIP solvers
  - `http://www.maxhs.org/`
  - **Open-WBO**
  - Winner of multiples tracks in the MaxSAT Competition 2014, 2015 and 2016!
  - `https://github.com/sat-group/open-wbo`
- Annual competition:
  - `http://maxsat-evaluations.github.io`
  - Modify a solver today and enter this year competition!

Standard Solver Input Format: DIMACS WCNF

- Variables indexed from 1 to n
- Negation: -
  - -3 stands for $\neg x_3$
- 0: special end-of-line character
- One special header "p"-line:
  `p wcnf #vars #clauses top`
  - #vars: number of variables
  - #clauses: number of clauses
  - top: "weight" of hard clauses
- Clauses represented as lists of integers
  - Weight is the first number
  - $(\neg x_3 \vee x_1 \vee \neg x_{45})$, weight 2:
    `2 -3 1 -45 0`
- Clause is hard if weight is equal to top

Standard Solver Input Format: DIMACS WCNF

Example: pointer analysis domain (pa-2.wcnf):

```
p wcnf 17997976 23364255 9223372036854775807
142 -11393180 12091478 0
200 -12496389 -1068725 13170751 0
209 -8854604 -8854942 -8854943 -8253894 9864153 0
174 -9406753 -8105076 11844088 0
200 -10403325 -8104972 12524177 0
142 -11987544 12096893 0
37 -10981341 -10980973 10838652 0
209 -9578314 -9579250 -9579251 -8254733 9578317 0
209 -8868994 -8870298 -8870299 -8254157 8868997 0
209 -9387012 -9387508 -9387509 -8253943 9387015 0
174 -9834074 -8106628 12074710 0
200 -10726788 -8105074 12909526 0
...
9223372036854775807 -13181184 0
9223372036854775807 -13181215 0

... truncated 763 MB
```

Push-Button Solver Technology

Example: $ open-wbo pa-2.wcnf

Push-Button Solver Technology

Example: $ open-wbo pa-2.wcnf

```
c Open-WBO: a Modular MaxSAT Solver
c Version:  MaxSAT Evaluation 2016
c Authors:  Ruben Martins, Vasco Manquinho, Ines Lynce
c Contributors:  Miguel Neves, Saurabh Joshi, Mikolas Janota
...
c Problem Type:  Weighted
c Number of variables:  17,997,976
c Number of hard clauses:  8,237,870
c Number of soft clauses:  15,126,385
c Parse time:  5.60 s
...
o 4699
o 4609
o 143
s OPTIMUM FOUND
c Total time:  361.26 s v 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15...

...17997976
```

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

M. Neves, R. Martins, M. Janota, I. Lynce, V. Manquinho Exploiting Resolution-Based Representations for MaxSAT Solving. SAT 2015: 272-286

Jessica Davies, Fahiem Bacchus: Postponing Optimization to Speed Up MAXSAT Solving. CP 2013: 247-262

Alexey Ignatiev, Antnio Morgado, Joao Marques-Silva: PySAT: A Python Toolkit for Prototyping with SAT Oracles. SAT 2018: 428-437

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

T. Philipp, P. Steinke. PBLib - A Library for Encoding Pseudo-Boolean Constraints into CNF. SAT 2015: 9-16 `http://tools.computational-logic.org/content/pblib.php`

Community Structure:

C. Ansótegui, J. Giráldez-Cru, Jordi Levy. The Community Structure of SAT Formulas. SAT 2012: 410-423

Web pages of interest:

MaxSAT Evaluation: `http://www.maxsat.udl.cat/`
Open-WBO: `http://sat.inesc-id.pt/open-wbo/`
SAT4J: `http://www.sat4j.org/`
RC2: `https://pysathq.github.io`
MaxHS: `http://www.maxhs.org/`
SATGraf: `https://bitbucket.org/znewsham/satgraf`