Heuristics for SAT/MaxSAT

Shaowei Cai

University of Chinese Academy of Sciences (UCAS)

2017

The SAT Problem

- A set of boolean variables: $X = \{x_1, x_2, ..., x_n\}$.
- Literals: $x_1, \neg x_1, x_2, \neg x_2, ..., x_n, \neg x_n$
- A set of Clauses: $x_1 \vee \neg x_2, x_2 \vee x_3, x_2 \vee \neg x_4, \neg x_1 \vee \neg x_3 \vee x_4, ...$
- A Conjunctive Normal Form (CNF) formula:

$$\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge (x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4)$$

- The satisfiability problem (SAT): test whether there exists an assignment of truth values to the variables in F under which F evaluates to true.

Importance of SAT

- Theoretical importance
  - the first NP-Complete problem [1971]
- Application everywhere
  - Verification of Win 7
  - Design of Intel Core CPU
  - ...
- Big progress

Solving SAT

Methods for solving SAT

- Complete methods for SAT:
  DPLL $\longrightarrow$ Conflict Driven Clause Learning (CDCL)
- Incomplete methods for SAT: Local search

Local Search for SAT

- Local search starts at some point in the solution space, and moves to adjacent points.
- two assignments are neighbors iff they only differ in one bit.

Algorithm 1: Local Search Framework for SAT

begin
  $s \leftarrow$ a randomly generated truth assignment;
  while *not reach terminal condition* **do**
    if s *satisfies* F **then** return s ;
    pick a variable x;
    $s := s$ with x flipped;
  return "Solution not found";
end

Scoring in Local Search for SAT

To pick the flipping variable, we need scoring functions.

- *cost*(α): number of unsatisfied clauses under assignment α.
- $score(x) = cost(F, \alpha) - cost(F, \alpha')$

Scoring in Local Search for SAT

To pick the flipping variable, we need scoring functions.

- *cost*(α): number of unsatisfied clauses under assignment α.
- $score(x) = cost(F, \alpha) - cost(F, \alpha')$
- *make*(x): the number of unsatisfied clauses that would become satisfied by flipping x.
- *break*(x): the number of satisfied clauses that would become unsatisfied by flipping x.
- $score(x) = make(x) - break(x)$.

Distinguishing Satisfied Clause

Scoring functions for SAT care about whether a clause is satisfied.

Distinguishing Satisfied Clause

Scoring functions for SAT care about whether a clause is satisfied.

But satisfied clauses are at the same safety level.

Define

A clause is said to be δ-satisfied *if and only if the clause contains* δ *true literals.*

δ-Satisfied Clause

Example:

- Given an assignment $s = \{x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1, x_5 = 1\}$
- $c_1 = x_1 \vee x_2 \vee \neg x_3 \vee x_4 \vee \neg x_5$
- $c_2 = x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4 \vee \neg x_5$

- c_1 is a 4-satisfied clause, while c_2 has 1-satisfied.
- If variable x_1 is flipped (from 1 to 0), then c_1 is still satisfied, while c_2 is unsatisfied.

δ-Satisfied Clause

Example:

- Given an assignment
  $s = \{x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1, x_5 = 1\}$
- $c_1 = x_1 \vee x_2 \vee \neg x_3 \vee x_4 \vee \neg x_5$
- $c_2 = x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4 \vee \neg x_5$

- c_1 is a 4-satisfied clause, while c_2 has 1-satisfied.
- If variable x_1 is flipped (from 1 to 0), then c_1 is still satisfied, while c_2 is unsatisfied.

Remark:
1-satisfied clauses are the most dangerous satisfied clauses.

Second Level Scoring

- $make_2(x)$ is the number of 1-satisfied clauses that would become 2-satisfied by flipping x.
- $break_2(x)$ is the number of 2-satisfied clauses that would become 1-satisfied by flipping x.
- $score_2(x) = make_2(x)$ - $break_2(x)$.

Use Second Level Score

Our first try:

- Break ties by $score_2$
  CCASat won Random SAT Track in SAT Challenge 2012
  (main ideas: Configuration Checking and $score_2$)

Use Second Level Score

Our first try:

- Break ties by $score_2$
  CCASat won Random SAT Track in SAT Challenge 2012 (main ideas: Configuration Checking and $score_2$)

Main Track: Random SAT

Rank	RiG	Solver	#solved	%solved	cum. run-time	median run-time
–	–	Virtual Best Solver (VBS)	558	93.0	72841	39.2
1	1	CCASat	423	70.5	76206	218.8
2	1	SATzilla2012 RAND	321	53.5	80796	714.4
3	2	SATzilla2012 ALL	306	51.0	83273	845.6
–	–	Sparrow2011 (SAT Competition 2011 Gold) (REFERENCE)	303	50.5	76396	876.1
–	–	EagleUP (SAT Competition 2011 Bronze) (REFERENCE)	283	47.2	83787	900.0

Use Second Level Score

More results:

- Combine $score_2$ and $score$, allowing to overwrite priority:
  $cscore(x) = score(x) + score_2(x)/d$
- Hybrid function:
  $hscore(x) = score(x) + score_2(x)/d + age(x)/\beta$

Use Second Level Score

More results:

- Combine $score_2$ and $score$, allowing to overwrite priority:
  $cscore(x) = score(x) + score_2(x)/d$
- Hybrid function:
  $hscore(x) = score(x) + score_2(x)/d + age(x)/\beta$

CSCCSat won 2nd Place Award of Random SAT Track in SAT Competition 2016, and is the best on solving k-SAT with $k > 3$.

Not suitable for short clauses formulas

Theorem

For a random 3-SAT formula $F(n, m)$, under any solution s to the formula, the number of 1-satisfied clauses is more than $m/2$.

In order to satisfy the formula, most clauses should be dangerous!
$\rightarrow$ Should not encourage them to become 2-satisfied.

Not suitable for short clauses formulas

Theorem

For a random 3-SAT formula $F(n, m)$, under any solution s to the formula, the number of 1-satisfied clauses is more than $m/2$.

In order to satisfy the formula, most clauses should be dangerous!
$\rightarrow$ Should not encourage them to become 2-satisfied.

Reference:
Shaowei Cai, Kaile Su: Local Search for Boolean Satisfiability with Configuration Checking and Subscore, Artificial Intelligence 2013.

MaxSAT Problems

Variants of MaxSAT

- MaxSAT: find an assignment that satisfies the most clauses.
- Partial MaxSAT: clauses are divided into hard and soft clauses, the goal is to satisfy all hard clauses, and as many soft clauses as possible.
- Weighted Partial MaxSAT

Solving MaxSAT

Popular methods for MaxSAT

- Complete Methods:
  - Branch and Bound
  - SAT-based solvers
- Incomplete Methods:
  - Local Search

Solving MaxSAT

Popular methods for MaxSAT

- Complete Methods:
  - Branch and Bound
  - SAT-based solvers
- Incomplete Methods:
  - Local Search

Over the past decade, perhaps the most successful approach to solving industrial instances of MaxSAT is the SAT-based approach, which relies on iteratively calling a SAT solver.

Methods for MaxSAT

- For many large industrial instances, because of the NP-hardness and the time constraint, we do not expect to solve them exactly.
- Goal: find a good-quality solution in a reasonable amount of time.
- To this end, many heuristic algorithms mainly local search have been developed.

Methods for MaxSAT

- For many large industrial instances, because of the NP-hardness and the time constraint, we do not expect to solve them exactly.
- Goal: find a good-quality solution in a reasonable amount of time.
- To this end, many heuristic algorithms mainly local search have been developed.

Local search has shown their great success in random and crafted instances.

Methods for MaxSAT

- For many large industrial instances, because of the NP-hardness and the time constraint, we do not expect to solve them exactly.
- Goal: find a good-quality solution in a reasonable amount of time.
- To this end, many heuristic algorithms mainly local search have been developed.

Local search has shown their great success in random and crafted instances.
However, local search has poor performance on industrial instances. (Maybe just unsuitable...)

Think Twice...

Think Twice...

- Power of SAT-based MaxSAT solvers on industrial problems

Think Twice...

- Power of SAT-based MaxSAT solvers on industrial problems
  $\leftarrow$ CDCL-based SAT solvers

Think Twice...

- Power of SAT-based MaxSAT solvers on industrial problems
  $\leftarrow$ CDCL-based SAT solvers
  $\leftarrow$ Reasoning Techniques: Unit Propagation, Clause learning.

Think Twice...

- Power of SAT-based MaxSAT solvers on industrial problems
  $\leftarrow$ CDCL-based SAT solvers
  $\leftarrow$ Reasoning Techniques: Unit Propagation, Clause learning.
- It seems real-world problems like reasoning.

Think Twice...

- Power of SAT-based MaxSAT solvers on industrial problems
  $\leftarrow$ CDCL-based SAT solvers
  $\leftarrow$ Reasoning Techniques: Unit Propagation, Clause learning.
- It seems real-world problems like reasoning.
  $\leftarrow$ How we human being solver real-world problems? Make some decisions, and figure out some middle results, then make some more decisions, and go on...

Think Twice...

- Power of SAT-based MaxSAT solvers on industrial problems
  $\leftarrow$ CDCL-based SAT solvers
  $\leftarrow$ Reasoning Techniques: Unit Propagation, Clause learning.
- It seems real-world problems like reasoning.
  $\leftarrow$ How we human being solver real-world problems? Make some decisions, and figure out some middle results, then make some more decisions, and go on...
  $\leftarrow$ The solutions are **consistent with itself** in many parts.

Think Twice...

- Power of SAT-based MaxSAT solvers on industrial problems
  $\leftarrow$ CDCL-based SAT solvers
  $\leftarrow$ Reasoning Techniques: Unit Propagation, Clause learning.
- It seems real-world problems like reasoning.
  $\leftarrow$ How we human being solver real-world problems? Make some decisions, and figure out some middle results, then make some more decisions, and go on...
  $\leftarrow$ The solutions are **consistent with itself** in many parts.
- What is wrong with local search?

Think Twice...

- Power of SAT-based MaxSAT solvers on industrial problems
  $\leftarrow$ CDCL-based SAT solvers
  $\leftarrow$ Reasoning Techniques: Unit Propagation, Clause learning.
- It seems real-world problems like reasoning.
  $\leftarrow$ How we human being solver real-world problems? Make some decisions, and figure out some middle results, then make some more decisions, and go on...
  $\leftarrow$ The solutions are **consistent with itself** in many parts.
- What is wrong with local search?
  $\leftarrow$ Not consider the relationship among variables.

Apply Unit propagation to MaxSAT

- A simple way to get solutions with more consistency: Unit propagation

Apply Unit propagation to MaxSAT

- A simple way to get solutions with more consistency: Unit propagation
- Worry about UP on MaxSAT $\rightarrow$ could be wrong.

Apply Unit propagation to MaxSAT

- A simple way to get solutions with more consistency: Unit propagation
- Worry about UP on MaxSAT $\rightarrow$ could be wrong.
  - What if we have many different reasoning chains? (Not many decision variables)

Apply Unit propagation to MaxSAT

- A simple way to get solutions with more consistency: Unit propagation
- Worry about UP on MaxSAT $\rightarrow$ could be wrong.
  - What if we have many different reasoning chains? (Not many decision variables)
  - What if we have chance to correct it?

From UP decimation to local search and back

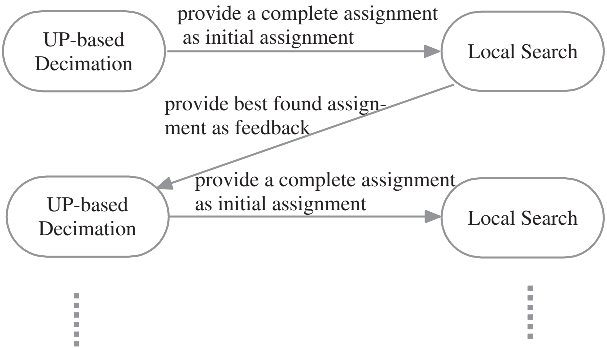
A novel approach to combine reasoning and local search:

- the UP decimation algorithm: give solutions according to different reasoning chains
- local search: search for better solutions nearby

From UP decimation to local search and back

A novel approach to combine reasoning and local search:

- the UP decimation algorithm: give solutions according to different reasoning chains
- local search: search for better solutions nearby



Results

Unweighted Max-SAT - Industrial

Solver	#Ins.	CnC-LS	dsat-wpm3-in-ms	WPM3-2015-in
ial/circuit-debugging-problems	3	2.76(2)	10.50(3)	3.68(3)
sean-safarpour	52	51.05 (45)	43.30(37)	24.18(35)
Total	55	47	40	38

Table: Results on UNSAT benchmarks from SAT Competition 2016.

Benchmark	#inst.	*DeciLS* #win.	*CCLS* #win.	*WPM3-2015-in* #win.	*MiniWalk* #win.	*dsat-wpm3-in-ms* #win.
App_Unknown	103	**80**	8	1	39	1
App_Unsat	109	**99**	22	14	34	12

Extension to (W)PMS

Table: Experimental Results on the MSE2016 PMS benchmarks.

Benchmark	#inst.	*DeciDist*		*Dist*	
		#win.	time	#win.	time
MSE2016_PMS_Industrial	601	**398**	84.91	225	57.31

Table: Experimental Results on MSE2016 WPMS benchmarks.

Benchmark	#inst.	*DeciCCEHC*		*CCEHC*	
		#win.	time	#win.	time
MSE2016_WPMS_Industrial	630	**319**	117.67	140	103.74

Extension to (W)PMS

Table: Experimental Results on the MSE2016 PMS benchmarks.

Benchmark	#inst.	*DeciDist*		*Dist*	
		#win.	time	#win.	time
MSE2016_PMS_Industrial	601	**398**	84.91	225	57.31

Table: Experimental Results on MSE2016 WPMS benchmarks.

Benchmark	#inst.	*DeciCCEHC*		*CCEHC*	
		#win.	time	#win.	time
MSE2016_WPMS_Industrial	630	**319**	117.67	140	103.74

Still much worse than SAT-based solvers on (W)PMS industrial instances.
DeciDist vs. WPM3-2015-in: 221 vs 533.

Conclusions

Conclusions:

- Consider the satisfactory degree in clauses, propose second level score for local search.
- Combine unit propagation based decimation algorithm and local search for MaxSAT.

Conclusions

Conclusions:

- Consider the satisfactory degree in clauses, propose second level score for local search.
- Combine unit propagation based decimation algorithm and local search for MaxSAT.

Future works:

- Apply second level score for other combinatorial problems with long clauses.
- Exploit more advanced reasoning in the new approach for Partial MaxSAT.