Preprocessing in MaxSAT Solving

Jeremias Berg

University of Helsinki
Finland

May 5th, 2021 Simons Institute / Online

Work done with Matti Järvisalo, Tuukka Korhonen, Paul Saikko, Marcus Leivo

Maximum Satisfiability

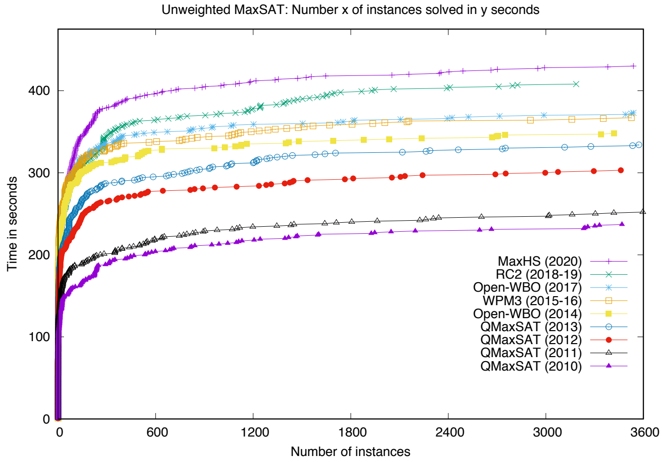
Plot by Ruben Martins

Maximum Satisfiability—MaxSAT

- Builds on the success story of SAT solving
- Great recent improvements in practical solver technology
- Expanding range of real-world applications

Solver Performance

Plot by Ruben Martins



Comparing some of the best solvers from 2010–2020:

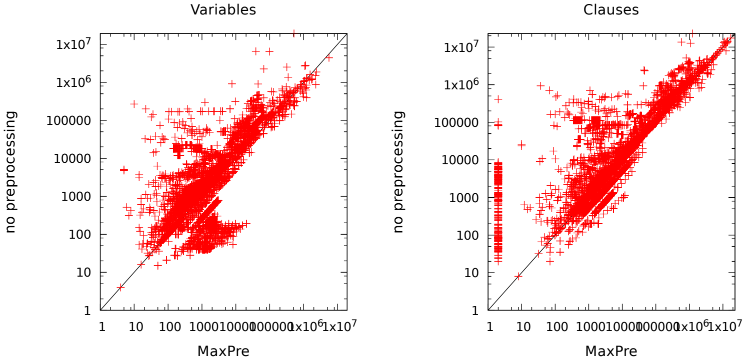
In 2020: 81% more instances solved than in 2010!

Solver Performance

An increasing number of MaxSAT solvers make use of different kinds of preprocessing.

MaxSAT preprocessing

[Korhonen, Berg, Saikko, and Järvisalo, 2017]



5425 MaxSAT instances collected and made available by the 2008–2016 MaxSAT Evaluations.

Outline

1. Basic concepts
2. MaxSAT preprocessing:
   - in practice
   - in theory
3. Conclusions

MaxSAT: Basic Definitions

MaxSAT

INPUT: a set of clauses F. (*a CNF formula*)

TASK: find τ s.t. $cost(\tau) = \sum_{C \in F} \tau(C)$ is maximized.

find a truth assignment that satisfies a maximum number of clauses

This is the standard definition, much studied in Theoretical Computer Science.

- Often inconvenient for modelling practical problems.

Central Generalization of MaxSAT

Partial MaxSAT

INPUT: sets H and S of hard and soft clauses
TASK: find model τ of H s.t:

$$cost(\tau) = \sum_{C \in S} \tau(C) \text{ is maximized}$$

find a truth assignment that satisfies all hard clauses and a maximum sum-of-weights of soft clauses

Central Generalization of MaxSAT

Partial MaxSAT

INPUT: sets H and S of hard and soft clauses
TASK: find model τ of H s.t:

$$cost(\tau) = \sum_{C \in S} \tau(C) \text{ is maximized}$$

find a truth assignment that satisfies all hard clauses and a maximum sum-of-weights of soft clauses

Note: *Can have weights on soft clauses*

Example

$$H = \{(x_1 \vee y), (\neg y \vee z),$$
$$(x_2 \vee y), (\neg y, \neg x_2)\}$$

$$S = \{(\neg x_1), (\neg x_1 \vee \neg x_2), (\neg z)\}$$

Example

$$H = \{(x_1 \vee y), (\neg y \vee z),$$
$$(x_2 \vee y), (\neg y, \neg x_2)\}$$

$$S = \{(\neg x_1), (\neg x_1 \vee \neg x_2), (\neg z)\}$$

$$\tau = \{\neg x_1, \neg x_2, y, z\}$$

$$H = \{(x_1 \vee y), (\neg y \vee z),$$
$$(x_2 \vee y), (\neg y, \neg x_2)\}$$

$$S = \{(\neg x_1), (\neg x_1 \vee \neg x_2), (\neg z)\}$$

$$cost(\tau) = 1$$

Context - MaxSAT solving

Branch & Bound MaxSAT solving

- Can be effective of small-but hard & randomly generated instances

SAT-based MaxSAT algorithms

- **Make extensive use of SAT solvers.**
- **Reduce MaxSAT solving into a sequence of satisfiability queries.**

Context - MaxSAT preprocessing

Preprocessing based on MaxSAT resolution

- Used in some B&B MaxSAT solvers
- [Heras and Bañeres, 2010; Argelich, Li, and Manyà, 2008]

SAT-based preprocessing in MaxSAT

- **Lift preprocessing rules (BVE, SE, BCE, . . .) from SAT solving to MaxSAT solving.**
- **[Belov, Morgado, and Marques-Silva, 2013; Berg and Järvisalo, 2016; Berg, Saikko, and Järvisalo, 2015; Korhonen, Berg, Saikko, and Järvisalo, 2017; Leivo, Berg, and Järvisalo, 2020; Paxian, Raiola, and Becker, 2021]**

The Nuts and Bolts of SAT-based MaxSAT preprocessing

Motivation

- SAT-based MaxSAT solvers build heavily on SAT-solvers.
- Preprocessing is important in SAT solving.

Motivation

- SAT-based MaxSAT solvers build heavily on SAT-solvers.
- Preprocessing is important in SAT solving.

Central Question

Can we leverage the work on SAT preprocessing in MaxSAT?

Preprocessing a SAT instance F

F

Preprocessing a SAT instance F

$$F \xrightarrow{\mathbf{1}} pre(F)$$

1. Preprocess
   - To obtain preprocessed instance $pre(F)$

Preprocessing a SAT instance F

$$F \xrightarrow{\mathbf{1}} pre(F) \xrightarrow{\mathbf{2}} \tau^{p}(pre(F)) = 1$$

1. Preprocess
   - To obtain preprocessed instance $pre(F)$
2. Solve
   - To obtain solution τ^{p} to $pre(F)$

Preprocessing a SAT instance F

$$F \xrightarrow{\mathbf{1}} pre(F) \xrightarrow{\mathbf{2}} \tau^p(pre(F)) = 1 \xrightarrow{\mathbf{3}} \tau(F) = 1$$

1. Preprocess
   - To obtain preprocessed instance $pre(F)$
2. Solve
   - To obtain solution τ^p to $pre(F)$
3. Reconstruct
   - To obtain solution τ to F

Does not work for MaxSAT

At least not directly

$$H = \{(x_1 \vee y), (\neg y \vee z),$$
$$(x_2 \vee y), (\neg y, \neg x_2)\}$$

$$S = \{(\neg x_1), (\neg x_1 \vee \neg x_2), (\neg z)\}$$

Does not work for MaxSAT

At least not directly

$$\xrightarrow{\textbf{BVE}}$$

$$H = \{(x_1 \vee y), (\neg y \vee z), (x_2 \vee y), (\neg y, \neg x_2)\}$$

$$pre(H) = \emptyset$$

$$S = \{(\neg x_1), (\neg x_1 \vee \neg x_2), (\neg z)\}$$

$$pre(S) = \emptyset$$

Does not work for MaxSAT

At least not directly

BVE

$$\longrightarrow$$

$$H = \{(x_1 \lor y), (\neg y \lor z), (x_2 \lor y), (\neg y, \neg x_2)\}$$

$$S = \{(\neg x_1), (\neg x_1 \lor \neg x_2), (\neg z)\}$$

$$\tau^p = \{x_1, x_2, \neg y, \neg z\}$$

$$pre(H) = \emptyset$$

$$pre(S) = \emptyset$$

$$cost(\tau^p) = 0$$

Does not work for MaxSAT

At least not directly

Reconstruct

$\longleftarrow$

$\tau = \{x_1, x_2, \neg y, \neg z\}$	$\tau^p = \{x_1, x_2, \neg y, \neg z\}$
$H = \{(x_1 \vee y), (\neg y \vee z), (x_2 \vee y), (\neg y, \neg x_2)\}$	$pre(H) = \emptyset$
$S = \{(\neg x_1), (\neg x_1 \vee \neg x_2), (\neg z)\}$	$pre(S) = \emptyset$
$cost(\tau) = 2$	$cost(\tau^p) = 0$

Does not work for MaxSAT

At least not directly

Reconstruct

$\longleftarrow$

$\tau^p = \{x_1, x_2, \neg y, \neg z\}$

$H = \{(x_1 \vee y), (\neg y$

$(x_2 \vee y), (\neg y$

Problem:
preprocessing preserves satisfiability
not the cost of models

$S = \{(\neg x_1), (\neg x_1 \vee \neg x_2), (\neg z)\}$

$pre(S) = \emptyset$

$cost(\tau^p) = 0$

Solution - Labels

[Belov, Morgado, and Marques-Silva, 2013]

$F = (H, S)$

Solution - Labels

[Belov, Morgado, and Marques-Silva, 2013]

$$F = (H, S) \xrightarrow{\mathbf{1}} (H, S^E)$$

1. Label
   - $S^E = \{C \vee l_C \mid C \in S\}$

Solution - Labels

[Belov, Morgado, and Marques-Silva, 2013]

$$F = (H, S) \xrightarrow{\mathbf{1}} (H, S^E) \xrightarrow{\mathbf{2}} F^P = (pre(H \cup S^E), S^L)$$

1. Label
   - $S^E = \{C \vee l_C \mid C \in S\}$
2. Preprocess
   - $pre(H \cup S^E)$, clauses remaining after preprocessing*.
   - $S^L = \{(\neg l_C) \mid C \in S\}$

Solution - Labels

[Belov, Morgado, and Marques-Silva, 2013]

$$F = (H, S) \xrightarrow{\mathbf{1}} (H, S^E) \xrightarrow{\mathbf{2}} F^P = (pre(H \cup S^E), S^L)$$

$$\downarrow \mathbf{3}$$

$$cost(\tau^P) = cost(F^P)$$

1. Label
   - $S^E = \{C \vee l_C \mid C \in S\}$
2. Preprocess
   - $pre(H \cup S^E)$, clauses remaining after preprocessing*.
   - $S^L = \{(\neg l_C) \mid C \in S\}$
3. Solve
   - τ^P optimal to F^P

Solution - Labels

[Belov, Morgado, and Marques-Silva, 2013]

$$F = (H, S) \xrightarrow{\mathbf{1}} (H, S^E) \xrightarrow{\mathbf{2}} F^P = (pre(H \cup S^E), S^L)$$

$$\downarrow \mathbf{3}$$

$$cost(\tau) = cost(F) \xleftarrow[\mathbf{4}]{} cost(\tau^P) = cost(F^P)$$

1. Label
   - $S^E = \{C \vee l_C \mid C \in S\}$
2. Preprocess
   - $pre(H \cup S^E)$, clauses remaining after preprocessing*.
   - $S^L = \{(\neg l_C) \mid C \in S\}$
3. Solve
   - τ^P optimal to F^P
4. Reconstruct
   - τ optimal to F

Solution - Labels

[Belov, Morgado, and Marques-Silva, 2013]

$$F = (H, S) \xrightarrow{\mathbf{1}} (H, S^E) \xrightarrow{\mathbf{2}} F^P = (pre(H \cup S^E), S^L)$$

$$\Big\downarrow \mathbf{3}$$

$$cost(\tau) = cost(F) \xleftarrow[\mathbf{4}]{} cost(\tau^P) = cost(F^P)$$

Note:
Labels also called: *relaxation variables*, *reification variables*, *blocking variables*, ...

Example

$$H = \{(x_1 \vee y), (\neg y \vee z),$$
$$(x_2 \vee y), (\neg y, \neg x_2)\}$$

$$S = \{(\neg x_1), (\neg x_1 \vee \neg x_2), (\neg z)\}$$

Example

$$\xrightarrow{\textbf{Label}}$$

$$H = \{(x_1 \vee y), (\neg y \vee z), (x_2 \vee y), (\neg y, \neg x_2)\}$$

$$S = \{(\neg x_1), (\neg x_1 \vee \neg x_2), (\neg z)\}$$

$$H = \{(x_1 \vee y), (\neg y \vee z), (x_2 \vee y), (\neg y, \neg x_2)\}$$

$$S^E = \{(\neg x_1 \vee l_1), (\neg x_1 \vee \neg x_2 \vee l_2), (\neg z \vee l_3)\}$$

1. Label

Example

$$H = \{(x_1 \vee y), (\neg y \vee z),$$
$$(x_2 \vee y), (\neg y, \neg x_2)\}$$

$$S^E = \{(\neg x_1 \vee l_1), (\neg x_1 \vee \neg x_2 \vee l_2),$$
$$(\neg z \vee l_3)\}$$

1. Label

Example

Preprocess

$\longrightarrow$

$$H = \{(x_1 \vee y), (\neg y \vee z), (x_2 \vee y), (\neg y, \neg x_2)\}$$

$$S^E = \{(\neg x_1 \vee l_1), (\neg x_1 \vee \neg x_2 \vee l_2), (\neg z \vee l_3)\}$$

$$pre(H \cup S^E) = \{(l_2 \vee l_3), (l_1 \vee l_3)\}$$

$$S^E = \{(\neg l_1), (\neg l_2), (\neg l_3)\}$$

1. Label
2. Preprocess $H \cup S^E$

Example

$H = \{(x_1 \vee y), (\neg y \vee z),$
$(x_2 \vee y), (\neg y, \neg x_2)\}$

$S^E = \{(\neg x_1 \vee l_1), (\neg x_1 \vee \neg x_2 \vee l_2),$
$(\neg z \vee l_3)\}$

$\tau^P = \{\neg l_1, \neg l_2, l_3\}$

$pre(H \cup S^E) =$
$\{(l_2 \vee l_3), (l_1 \vee l_3)\}$

$S^E = \{(\neg l_1), (\neg l_2), (\neg l_3)\}$

$cost(\tau^P) = 1$

1. Label
2. Preprocess $H \cup S^E$
3. Solve

Example

Reconstruct

$\longleftarrow$

$\tau = \{\neg x_1, \neg x_2, y, z\}$

$H = \{(x_1 \vee y), (\neg y \vee z),$
$(x_2 \vee y), (\neg y, \neg x_2)\}$

$S^E = \{(\neg x_1), (\neg x_1 \vee \neg x_2),$
$(\neg z)\}$

$cost(\tau) = 1$

$\tau^P = \{\neg l_1, \neg l_2, l_3\}$

$pre(H \cup S^E) =$
$\{(l_2 \vee l_3), (l_1 \vee l_3)\}$

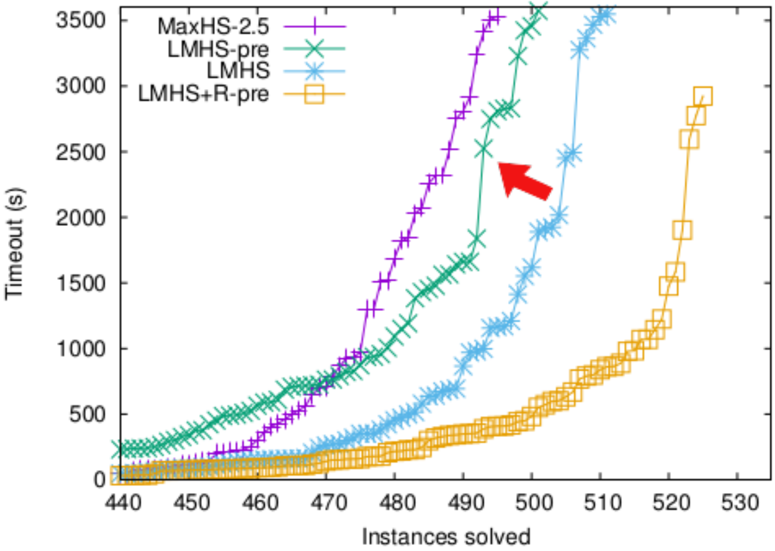
$S^E = \{(\neg l_1), (\neg l_2), (\neg l_3)\}$

$cost(\tau^P) = 1$

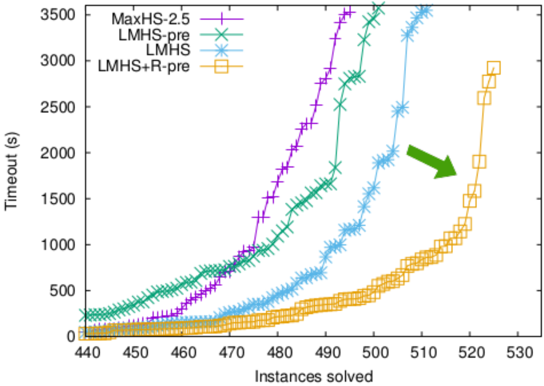
1. Label
2. Preprocess $H \cup S^E$
3. Solve
4. Reconstruct

However

[Berg, Saikko, and Järvisalo, 2015]



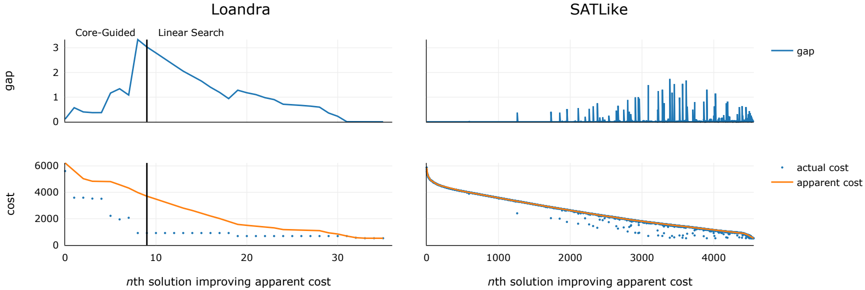
Solution



- Modify the solver to use labels directly as assumptions,
- Today implemented by many available solvers
  - *but not necessarily all*.

We are not quite done....

[Leivo, Berg, and Järvisalo, 2020]



- Intermediate solutions central in many solvers.
- Costs of intermediate solutions of $pre(F)$ are often misrepresented.

Beyond SAT-based preprocessing

Many solvers incorporate various "preprocessing-like" heuristics during search:

Beyond SAT-based preprocessing

Many solvers incorporate various "preprocessing-like" heuristics during search:

- Stratification & Hardening [Ansótegui, Bonet, Gabàs, and Levy, 2012]
  - use bounds on $cost(F)$ in order to harden soft clauses.

Beyond SAT-based preprocessing

Many solvers incorporate various "preprocessing-like" heuristics during search:

- Stratification & Hardening [Ansótegui, Bonet, Gabàs, and Levy, 2012]
  - use bounds on $cost(F)$ in order to harden soft clauses.
- Reduced cost fixing [Bacchus, Hyttinen, Järvisalo, and Saikko, 2017]
  - use reduced cost of soft clauses to fix them to true or false.

Beyond SAT-based preprocessing

Many solvers incorporate various "preprocessing-like" heuristics during search:

- Stratification & Hardening [Ansótegui, Bonet, Gabàs, and Levy, 2012]
  - use bounds on *cost*(F) in order to harden soft clauses.
- Reduced cost fixing [Bacchus, Hyttinen, Järvisalo, and Saikko, 2017]
  - use reduced cost of soft clauses to fix them to true or false.
- and others: [Paxian, Raiola, and Becker, 2021; Ignatiev, Morgado, and Marques-Silva, 2019; Martins, Manquinho, and Lynce, 2013]

Take Home Message

Preprocessing can be effective in MaxSAT

Take Home Message

Preprocessing can be effective in MaxSAT

as long as your solver supports it

The Theory of SAT-based MaxSAT preprocessing

Central Concepts

Cores & MUSes

A set $\kappa \subset S$ of an instance (H, S) is:

Central Concepts

Cores & MUSes

A set $\kappa \subset S$ of an instance (H, S) is:

- A core if $H \wedge \kappa$ is UNSAT

Central Concepts

Cores & MUSes

A set $\kappa \subset S$ of an instance (H, S) is:

- A core if $H \wedge \kappa$ is UNSAT
- An MUS if $H \wedge \kappa_s$ SAT for all $\kappa_s \subset \kappa$.
- $\textsc{mus}(F)$ = set of all MUSes of F.

Central Concepts

Cores & MUSes

A set $\kappa \subset S$ of an instance (H, S) is:

- A core if $H \wedge \kappa$ is UNSAT
- An MUS if $H \wedge \kappa_s$ SAT for all $\kappa_s \subset \kappa$.
- $\textsc{mus}(F) =$ set of all MUSes of F.

Why relevant for MaxSAT?

Every solution to (H, S) falsifies at least one clause from each MUS.

What we know

Theorem

[Belov, Morgado, and Marques-Silva, 2013]

Preprocess F with BVE, BCE, SE or SSR to obtain $pre(F)$.
Then:

$$\textsc{mus}(F) = \textsc{mus}(pre(F))$$

What we know

Theorem

[Belov, Morgado, and Marques-Silva, 2013]

Preprocess F with BVE, BCE, SE or SSR to obtain $pre(F)$.
Then:

$$\textsc{mus}(F) = \textsc{mus}(pre(F))$$

Consequence

[Berg and Järvisalo, 2016]

The best case number of iterations (SAT-solver calls) of many MaxSAT solvers when solving F and $pre(F)$ are equal.

More Generally

[Berg and Järvisalo, 2019; Järvisalo, Heule, and Biere, 2012]

Max-RAT

- Extension of resolvent asymetric tautologies (RAT) to MaxSAT.
- (Informally) C is Max-RAT if it is RAT on a non-label variable.

More Generally

[Berg and Järvisalo, 2019; Järvisalo, Heule, and Biere, 2012]

Max-RAT

- Extension of resolvent asymetric tautologies (RAT) to MaxSAT.
- (Informally) C is Max-RAT if it is RAT on a non-label variable.

Theorem

Preprocess F with any techniques corresponding to the addition and removal of Max-RAT clauses to obtain $pre(F)$.
Then:

$$\textsc{mus}(F) = \textsc{mus}(pre(F))$$

More Generally

[Berg and Järvisalo, 2019; Järvisalo, Heule, and Biere, 2012]

Max-RAT

- Extension of resolvent asymetric tautologies (RAT) to MaxSAT.
- (Informally) C is Max-RAT if it is RAT on a non-label variable.

Theorem

Preprocess F with any techniques corresponding to the addition and removal of Max-RAT clauses to obtain $pre(F)$.
Then:

$$\textsc{mus}(F) = \textsc{mus}(pre(F))$$

so the best-case number of iterations of solvers are equal

Take-Home Message

Significantly altering the execution of MaxSAT solvers via preprocessing requires affecting the MUSes

Take-Home Message

Significantly altering the execution of MaxSAT solvers via preprocessing requires affecting the MUSes

Liftings of commonly used techniques in SAT do not do this

Beyond techniques from SAT

- **hardening, reduced cost fixing, subsumed label elimination**

  [Bacchus, Hyttinen, Järvisalo, and Saikko, 2017; Ansótegui, Bonet, Gabàs, and Levy, 2012; Berg, Saikko, and Järvisalo, 2016]

  - Can fix soft clauses in MUSes.
- **intrinsic at-most-one constraints**

  [Ignatiev, Morgado, and Marques-Silva, 2019]

  - Can alter MUSes.

Take-Home Message

Many effective preprocessing rules for MaxSAT have been proposed as part of solver heuristics
Unifying the theory underlying the existing methods could lead to new insights & more effective solvers

Preprocessing for MaxSAT- Summary

- Preprocessing for MaxSAT can be effective
  - Requires careful integration with the solver.
- The field is divided.
  - Unifying the theory underlying existing techniques central for further improvements.

Preprocessing for MaxSAT- Summary

- Preprocessing for MaxSAT can be effective
  - Requires careful integration with the solver.
- The field is divided.
  - Unifying the theory underlying existing techniques central for further improvements.

Available software

- MaxPRE: `https://github.com/Laakeri/maxpre`

  [Korhonen, Berg, Saikko, and Järvisalo, 2017]
- Coprocessor [Manthey, 2012]

Further Reading and Links

Talks at the Simons Institute

- Fahiem Bacchus on incremental SAT and MaxSAT on April 1st.
- Jeremias and Matti Järvisalo on MaxSAT on April 13th.

Surveys

- "Maximum Satisfiability" by Bacchus, Järvisalo & Martins
  - Chapter in forthcoming vol. 2 of Handbook of Satisfiability
  - Preprint available.
- Somewhat older surveys:
  - Handbook chapter on MaxSAT: [Li and Manyà, 2009]
  - Surveys on MaxSAT algorithms:

    [Ansótegui, Bonet, and Levy, 2013]

    [Morgado, Heras, Liffiton, Planes, and Marques-Silva, 2013]

MaxSAT Evaluations

https://maxsat-evaluations.github.io

Most recent report: [Bacchus, Järvisalo, and Martins, 2019]

Thank you for attending!

Bibliography I

Carlos Ansótegui, Maria Luisa Bonet, Joel Gabàs, and Jordi Levy. Improving sat-based weighted maxsat solvers. In Michela Milano, editor, *Principles and Practice of Constraint Programming - 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*, volume 7514 of *Lecture Notes in Computer Science*, pages 86–101. Springer, 2012. doi: 10.1007/978-3-642-33558-7_9. URL https://doi.org/10.1007/978-3-642-33558-7_9.

Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. SAT-based MaxSAT algorithms. *Artificial Intelligence*, 196: 77–105, 2013. doi: 10.1016/j.artint.2013.01.002. URL http://dx.doi.org/10.1016/j.artint.2013.01.002.

Josep Argelich, Chu Min Li, and Felip Manyà. A preprocessor for Max-SAT solvers. In Hans Kleine Büning and Xishun Zhao, editors, *Theory and Applications of Satisfiability Testing - SAT 2008, 11th International Conference, SAT 2008, Guangzhou, China, May 12-15, 2008. Proceedings*, volume 4996 of *Lecture Notes in Computer Science*, pages 15–20. Springer, 2008.

Fahiem Bacchus, Antti Hyttinen, Matti Järvisalo, and Paul Saikko. Reduced cost fixing in MaxSAT. In J. Christopher Beck, editor, *Principles and Practice of Constraint Programming - 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, volume 10416 of *Lecture Notes in Computer Science*, pages 641–651. Springer, 2017. doi: 10.1007/978-3-319-66158-2_41. URL https://doi.org/10.1007/978-3-319-66158-2_41.

Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. Maxsat evaluation 2018: New developments and detailed results. *J. Satisf. Boolean Model. Comput.*, 11(1):99–131, 2019. doi: 10.3233/SAT190119. URL https://doi.org/10.3233/SAT190119.

Anton Belov, António Morgado, and João Marques-Silva. SAT-based preprocessing for MaxSAT. In Kenneth L. McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings*, volume 8312 of *Lecture Notes in Computer Science*, pages 96–111. Springer, 2013.

Jeremias Berg and Matti Järvisalo. Impact of SAT-based preprocessing on core-guided MaxSAT solving. In Michel Rueher, editor, *Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*, volume 9892 of *Lecture Notes in Computer Science*, pages 66–85. Springer, 2016. doi: 10.1007/978-3-319-44953-1_5. URL https://doi.org/10.1007/978-3-319-44953-1_5.

Bibliography II

Jeremias Berg and Matti Järvisalo. Unifying reasoning and core-guided search for maximum satisfiability. In Francesco Calimeri, Nicola Leone, and Marco Manna, editors, *Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Rende, Italy, May 7-11, 2019, Proceedings*, volume 11468 of *Lecture Notes in Computer Science*, pages 287–303. Springer, 2019. doi: 10.1007/978-3-030-19570-0_19. URL https://doi.org/10.1007/978-3-030-19570-0_19.

Jeremias Berg, Paul Saikko, and Matti Järvisalo. Improving the effectiveness of SAT-based preprocessing for MaxSAT. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 239–245. AAAI Press, 2015. URL http://ijcai.org/papers15/Abstracts/IJCAI15-040.html.

Jeremias Berg, Paul Saikko, and Matti Järvisalo. Subsumed label elimination for maximum satisfiability. In Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, and Frank van Harmelen, editors, *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 630–638. IOS Press, 2016. doi: 10.3233/978-1-61499-672-9-630. URL https://doi.org/10.3233/978-1-61499-672-9-630.

Katalin Fazekas, Armin Biere, and Christoph Scholl. Incremental inprocessing in SAT solving. In *SAT*, volume 11628 of *Lecture Notes in Computer Science*, pages 136–154. Springer, 2019.

Federico Heras and David Bañeres. The impact of max-sat resolution-based preprocessors on local search solvers. *J. Satisf. Boolean Model. Comput.*, 7(2-3):89–126, 2010.

Marijn J. H. Heule, Benjamin Kiesl, Martina Seidl, and Armin Biere. Pruning through satisfaction. In *Haifa Verification Conference*, volume 10629 of *Lecture Notes in Computer Science*, pages 179–194. Springer, 2017.

Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Strong extension-free proof systems. *J. Autom. Reason.*, 64 (3):533–554, 2020.

Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an efficient maxsat solver. *J. Satisf. Boolean Model. Comput.*, 11(1):53–64, 2019.

Matti Järvisalo, Marijn Heule, and Armin Biere. Inprocessing rules. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, volume 7364 of *Lecture Notes in Computer Science*, pages 355–370. Springer, 2012. doi: 10.1007/978-3-642-31365-3_28. URL https://doi.org/10.1007/978-3-642-31365-3_28.

Bibliography III

Benjamin Kiesl, Martina Seidl, Hans Tompits, and Armin Biere. Local redundancy in SAT: generalizations of blocked clauses. *Log. Methods Comput. Sci.*, 14(4), 2018.

Tuukka Korhonen, Jeremias Berg, Paul Saikko, and Matti Järvisalo. MaxPre: An extended MaxSAT preprocessor. In Serge Gaspers and Toby Walsh, editors, *Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, volume 10491 of *Lecture Notes in Computer Science*, pages 449–456. Springer, 2017. doi: 10.1007/978-3-319-66263-3_28. URL https://doi.org/10.1007/978-3-319-66263-3_28.

Marcus Leivo, Jeremias Berg, and Matti Järvisalo. Preprocessing in incomplete maxsat solving. In *ECAI*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 347–354. IOS Press, 2020.

Chu Min Li and Felip Manyà. MaxSAT, hard and soft constraints. In *Handbook of Satisfiability*, pages 613–631. 2009. doi: 10.3233/978-1-58603-929-5-613. URL http://dx.doi.org/10.3233/978-1-58603-929-5-613.

Norbert Manthey. Coprocessor 2.0 - A flexible CNF simplifier - (tool presentation). In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, volume 7317 of *Lecture Notes in Computer Science*, pages 436–441. Springer, 2012. doi: 10.1007/978-3-642-31612-8_34. URL https://doi.org/10.1007/978-3-642-31612-8_34.

Ruben Martins, Vasco M. Manquinho, and Inês Lynce. Community-based partitioning for MaxSAT solving. In Matti Järvisalo and Allen Van Gelder, editors, *Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings*, volume 7962 of *Lecture Notes in Computer Science*, pages 182–191. Springer, 2013. ISBN 978-3-642-39070-8. doi: 10.1007/978-3-642-39071-5. URL http://dx.doi.org/10.1007/978-3-642-39071-5.

António Morgado, Federico Heras, Mark H. Liffiton, Jordi Planes, and João Marques-Silva. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints*, 18(4):478–534, 2013. doi: 10.1007/s10601-013-9146-2. URL http://dx.doi.org/10.1007/s10601-013-9146-2.

Tobias Paxian, Pascal Raiola, and Bernd Becker. On preprocessing for weighted maxsat. In *VMCAI*, volume 12597 of *Lecture Notes in Computer Science*, pages 556–577. Springer, 2021.