Proofs and Certificates for Max-SAT (Extended Abstract) *

Matthieu Py[1,2] , **Mohamed Sami Cherif**[1] , **Djamal Habet**[1]

[1]Aix-Marseille Université, Université de Toulon, CNRS, LIS, Marseille, France
[2]Université Clermont Auvergne, Mines Saint-Etienne, CNRS, LIMOS, F-63000 Clermont-Ferrand, France

matthieu.py@uca.fr, {mohamed-sami.cherif, djamal.habet}@univ-amu.fr

Abstract

In this paper, we present a tool, called MS-Builder, which generates certificates for the Max-SAT problem in the particular form of a sequence of equivalence-preserving transformations. To generate a certificate, MS-Builder iteratively calls a SAT oracle to get a SAT resolution refutation which is handled and adapted into a sound refutation for Max-SAT. In particular, the size of the computed Max-SAT refutation is linear with respect to the size of the initial refutation if it is semi-read-once, tree-like regular, tree-like or semi-tree-like. Additionally, we propose an extendable tool, called MS-Checker, able to verify the validity of any Max-SAT certificate using Max-SAT inference rules.

1 Introduction

Given a Boolean formula in Conjunctive Normal Form (CNF), the Maximum Satisfiability (Max-SAT) problem consists in determining the maximum number of clauses that it is possible to satisfy by an assignment of the variables, while the Satisfiability (SAT) problem simply ascertains whether there exists an assignment which satisfies all the clauses. Max-SAT is an optimization extension of the satisfiability problem and is a natural formalism enabling to model many real-world and crafted problems [Muise *et al.*, 2016; Zhang and Bacchus, 2012; Demirovic and Musliu, 2017; Manyà *et al.*, 2020; Achá and Nieuwenhuis, 2014; Bofill *et al.*, 2015; Xu *et al.*, 2003; Guerra and Lynce, 2012; D'Almeida and Grégoire, 2012]. Different complete solving paradigms for Max-SAT have seen the day including Branch and Bound algorithms [Li *et al.*, 2007; Küegel, 2012; Abramé and Habet, 2014; Li *et al.*, 2022] and SAT-based algorithms [Fu and Malik, 2006; Manquinho *et al.*, 2009; Ansótegui *et al.*, 2009; Davies and Bacchus, 2011; Ansótegui *et al.*, 2013; Martins *et al.*, 2014; Ignatiev *et al.*, 2019].

Inference plays an important role in the context of Max-SAT solving [Li *et al.*, 2007; Narodytska and Bacchus, 2014; Abramé and Habet, 2014] and this has led to an increasing interest in studying proof systems for Max-SAT in the literature [Larrosa and Heras, 2005; Bonet *et al.*, 2006; Bonet *et al.*, 2007; Larrosa and Rollon, 2020a; Larrosa and Rollon, 2020b; Bonet and Levy, 2020; Filmus *et al.*, 2020; Cherif *et al.*, 2022]. In particular, Max-SAT resolution [Larrosa and Heras, 2005; Bonet *et al.*, 2006; Bonet *et al.*, 2007] is one of the first known complete systems for Max-SAT and is a natural extension of the resolution rule [Robinson, 1965] used in the context of SAT. Max-SAT resolution proofs are more constrained than their SAT counterparts as the premise clauses are replaced by the conclusions when applying Max-SAT resolution. Consequently, switching from a resolution proof to a Max-SAT resolution proof is possible and well-known for the particular case of read-once resolution [Bonet *et al.*, 2007; Heras and Marques-Silva, 2011], where clauses can be used at most once in the proof. However, the adaptation of any resolution proof to a Max-SAT resolution proof is an established problem. Bonet et al. state that *"it seems difficult to adapt a classical resolution proof to get a Max-SAT resolution proof, and it is an open question if this is possible without increasing substantially the size of the proof"* [Bonet *et al.*, 2006].

In this paper, we first contribute to the open problem of adapting resolution refutations for Max-SAT. To this end, we augment Max-SAT resolution with the split rule which allows to generate two clauses subsumed by the original clause. We prove that it is always possible to adapt a resolution refutation into a max-refutation, i.e., a refutation using Max-SAT inference rules, whose size is linear with respect to the initial refutation for the following cases: semi-read-once resolution, tree-like regular resolution, tree-like resolution and semi-tree-like resolution. Furthermore, we propose a complete adaptation for any resolution refutation into a max-refutation, although with a worst-case exponential blow-up in the size of the proofs.

Secondly, we propose an independent tool, called MS-Builder, able to build certificates for the Max-SAT problem. To build such certificates, MS-Builder iteratively calls a SAT oracle to get a resolution refutation, adapts it for Max-SAT and applies it on the current formula. Moreover, we implemented an associated tool, called MS-Checker to check the validity of the certificates. Both tools are experimentally evaluated on the unweighted and weighted benchmarks of the 2020 Max-SAT Evaluation [Bacchus *et al.*, 2020].

*This paper is an extended abstract our work [Py et al., 2022], published in the *Journal of Artificial Intelligence Research (JAIR)*, and is an extension of our previous work [Py et al., 2019; Py et al., 2021a].

2 Preliminaries

2.1 Definitions and Notations

Let X be the set of propositional variables. A literal l is a variable $x \in X$ or its negation $\overline{x}$. A clause $c = (l_1 \vee l_2 \vee ... \vee l_k)$ is a disjunction of literals. A unit clause is composed of only one literal. A formula in Conjunctive Normal Form (CNF) $\phi = c_1 \wedge c_2 \wedge ... \wedge c_m$ is a conjunction of clauses. An assignment $I : X \longrightarrow \{0, 1\}$ maps each variable to a Boolean value. A literal l is satisfied (resp. falsified) by an assignment I if $l \in I$ (resp. $\overline{l} \in I$). A clause c is satisfied by an assignment I if at least one of its literals is satisfied by I, otherwise it is falsified. The empty clause $\square$ contains zero literals and is always falsified. A CNF formula ϕ is satisfied by an assignment I, that we call model of ϕ, if each clause $c \in \phi$ is satisfied by I, otherwise it is falsified. Solving the Satisfiability problem (SAT) consists in determining whether there exists an assignment I that satisfies a given CNF formula ϕ. In the case where such an assignment exists, we say that ϕ is satisfiable, otherwise we say that ϕ is unsatisfiable or inconsistent. Solving the Maximum Satisfiability problem (Max-SAT) consists in determining the maximum number of clauses that can be satisfied by an assignment of a CNF formula ϕ, or equivalently the minimum number of clauses that each assignment must falsify. In the weighted partial Max-SAT problem, a finite or infinite weight is associated to each clause, representing the penalty of falsifying it.

2.2 SAT Resolution

To certify that a CNF formula is satisfiable, it is sufficient to exhibit a model of the formula. On the other hand, to prove that a CNF formula is unsatisfiable, we need to refute the existence of a model. A well-known SAT refutation system is based on an inference rule for SAT called resolution [Robinson, 1965]. The resolution rule deduces a clause called resolvent which can be added to the formula. Note that this rule is sound for SAT as it maintains SAT equivalence (models are the same before and after the transformation) and it is extensively used in the context of SAT solving and particularly the CDCL framework [Silva and Sakallah, 1996].

Definition 1 (Resolution [Robinson, 1965]).

$$\frac{c_1 = (x \vee A) \quad c_2 = (\overline{x} \vee B)}{c_3 = (A \vee B)}$$

It is possible to prove that a formula is unsatisfiable using a resolution refutation, which is a sequence of resolutions leading to an empty clause. Many restricted classes of resolution refutations have been studied in the literature namely linear [Loveland, 1970], unit [Hertel and Urquhart, 2009], input [Hertel and Urquhart, 2009], regular [Urquhart, 2011], read-once [Iwama and Miyano, 1995] and tree-like resolution refutations [Ben-Sasson *et al.*, 2004] among others. In particular, a resolution refutation is tree-like if every intermediate clause is used at most once in the proof. Similarly, a resolution refutation is read-once if each clause is used at most once in the proof. Finally, a resolution refutation is regular if each branch, i.e., path from a leaf to the empty clause, contains at most one resolution per variable.

Example 1. *We consider the CNF formula $\phi = (\overline{x_1} \vee x_3) \wedge (x_1) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee \overline{x_3})$. The resolution refutation of ϕ, represented in Figure 1, is tree-like (and) regular, but not read-once because of clause (x_1).*

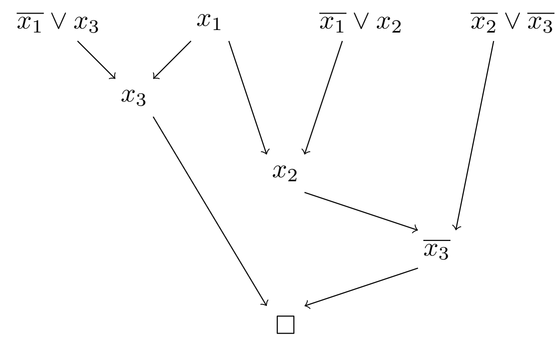

Figure 1: A resolution refutation

2.3 Max-SAT Resolution

One of the first and most studied proof systems for Max-SAT is based on an inference rule called Max-SAT resolution, which is an extension of the resolution rule. The aim of complete Max-SAT systems is to compute the Max-SAT optimum of a given CNF formula, i.e., the maximum number of falsified clauses. The formula is thus refuted as many times as its optimum through equivalence-preserving transformations in the sense of Max-SAT (each assignment falsfies the same amount of clauses before and after the transformation). Other than the resolvent clause, Max-SAT resolution introduces new clauses referred to as compensation clauses and which are essential to preserve Max-SAT equivalence.

Definition 2 (Max-SAT resolution [Larrosa and Heras, 2005; Bonet *et al.*, 2006; Bonet *et al.*, 2007]).

$$\frac{c_1 = x \vee A \qquad\qquad c_2 = \overline{x} \vee B}{\begin{array}{c} c_r = A \vee B \\ cc_1 = x \vee A \vee \overline{b_1} \\ cc_2 = x \vee A \vee b_1 \vee \overline{b_2} \\ \vdots \\ cc_t = x \vee A \vee b_1 \vee ... \vee b_{t-1} \vee \overline{b_t} \\ cc_{t+1} = \overline{x} \vee B \vee \overline{a_1} \\ cc_{t+2} = \overline{x} \vee B \vee a_1 \vee \overline{a_2} \\ \vdots \\ cc_{t+s} = \overline{x} \vee B \vee a_1 \vee ... \vee a_{s-1} \vee \overline{a_s} \end{array}}$$

As a sound and complete rule for Max-SAT [Bonet *et al.*, 2006], Max-SAT resolution plays an important role in the context of Max-SAT theory and solving. In particular, it is extensively used in the context of Branch and Bound algorithms [Li *et al.*, 2007; Küegel, 2012; Abramé and Habet, 2014; Cherif *et al.*, 2020] and more marginally in the context of SAT-based algorithms [Heras and Marques-Silva, 2011; Narodytska and Bacchus, 2014]. For a given CNF formula, it is always possible to generate a Max-SAT resolution proof of its optimum by applying the saturation algorithm [Bonet *et al.*, 2006] to deduce empty clauses. A Max-SAT refutation, or simply max-refutation, is a sequence of Max-SAT inference steps deducing the empty clause. Its size is the number of its inference steps.

Example 2. *We consider the CNF formula from Example 1. A hand-made max-refutation of ϕ was proposed in [Bonet* et al.*, 2006] and is represented in Figure 2.*

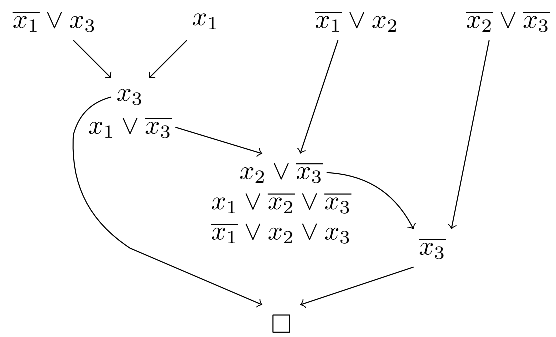

Figure 2: A max-refutation

In recent work, Max-SAT resolution was augmented with other rules such as the split rule [Larrosa and Rollon, 2020b; Bonet and Levy, 2020; Py *et al.*, 2021b; Py *et al.*, 2021c] or the extension rule [Larrosa and Rollon, 2020a]. The addition of such rules to Max-SAT resolution can improve its efficiency in generating shorter proofs [Larrosa and Rollon, 2020b; Larrosa and Rollon, 2020a; Py *et al.*, 2021c] or in simulating other proof systems [Filmus *et al.*, 2020; Bonet and Levy, 2020]. To be exhaustive, we must also mention that other Max-SAT proof systems were introduced and studied in the literature [Li *et al.*, 2016; Atserias and Lauria, 2019; Larrosa and Rollon, 2020a; Filmus *et al.*, 2020].

Definition 3 (Split rule)**.** *Given a clause $c_1 = (A)$ where A is a disjunction of literals and x a variable, the split rule replaces the premise c_1 by two new clauses as follows:*

$$\frac{c_1 = (A)}{c_2 = (x \vee A) \qquad c_3 = (\overline{x} \vee A)}$$

If these proof systems have been extensively studied in theory, generating proofs remains an unexplored topic in practice. Hence, this work aims to contribute to this topic by proposing tools to build and check certificates for the Max-SAT problem. To this aim, we first propose adaptations from resolution refutations to max-refutations. These adaptations are used in a tool enabling to build certificates for the Max-SAT problem. For the sake of simplicity, we will exhibit examples with unweighted unpartial formulas to introduce MS-Builder. However, MS-Builder is also able to generate certificates for weighted (partial) Max-SAT formulas, using the fold and the unfold rules, and the weighted version of Max-SAT resolution and split [Bonet *et al.*, 2007; Larrosa *et al.*, 2008; Larrosa and Rollon, 2020b].

Definition 4 (Fold & Unfold)**.** *Given a weighted clause c and two positive weights w_1 and w_2, the fold and unfold rules are respectively defined as follows:*

$$\frac{(c, w_1) \quad (c, w_2)}{(c, w_1 + w_2)} \qquad\qquad \frac{(c, w_1 + w_2)}{(c, w_1) \quad (c, w_2)}$$

3 From Resolution Refutations to Max-Refutations

In the state of the art, the adaptation of any resolution refutation to get a max-refutation is known possible in the *read-once* case, and the size of the computed max-refutation is linear with respect to the size of the initial resolution refutation. In our work [Py *et al.*, 2022; Py *et al.*, 2020], we prove extend this result in the case of semi-read-once, tree-like regular, tree-like or semi-tree-like resolution. We also prove that the adaptation is always possible in the unrestricted case, but with a worst-case exponential blow-up in the size of the proofs. The theoretical results are resumed in Table 1.

Resolution Refutation	Size of the Max-Refutation
Read-Once	Linear [Bonet *et al.*, 2007]
Semi-Read-once	Linear
Tree-like regular	Linear
Tree-like	Linear
Semi-tree-like	Linear
Unrestricted	Exponential

Table 1: Adaptation of resolution refutations for Max-SAT

3.1 From Semi-Read-Once Resolution Refutations to Max-Refutations

SAT algorithms are based on unit propagation, which means that when a unit clause is deduced, the value of its only literal is propagated in the whole formula, because satisfying this literal is necessary to satisfy the formula. Applying unit propagation can be seen as the use of a particular unit clause in several resolution steps. As such, transforming resolution refutations to fix non-read-once unit clauses can therefore be a useful preprocessing technique to our proof builder which relies on iterative calls to a SAT oracle, as will be shown in Section 4. To fix a non-read-once unit clause, we remove the resolution steps in which it is involved and we add a new resolution step at the end of the refutation. Such a strategy works when the refutation is *based on unit propagation*, i.e., every time a resolution step is applied on a unit clause, the variable contained in the unit clause no longer appears in the rest of the refutation. As SAT algorithms make a strong application of the unit propagation technique, we made the hypothesis, confirmed by experiments, that the computed resolution refutation will be often based on unit propagation. The proposed transformation can be seen as a preprocessing technique for any non-read-once resolution refutation. In particular, some non-read-once resolution refutations can be non-read-once only because of unit clauses and we say that such refutations are *semi-read-once*.

Definition 5 (Semi-read-once)**.** *A resolution refutation is semi-read-once if it is based on unit propagation and if each non-read-once clause is also a unit clause.*

Theorem 1. *Given an unsatisfiable formula ϕ and a semi-read-once resolution refutation P of ϕ, there exists a max-refutation of ϕ containing $O(|P|)$ inference steps.*

3.2 From Tree-Like Regular Resolution Refutations to Max-Refutations

To adapt a tree-like regular resolution refutation for Max-SAT, the idea is to use the split rule to fix non-read-once clauses. More precisely, if a clause c is used k times ($k > 1$) as a premise of a resolution step, we use the split rule on clause c with respect to a particular variable x which is carefully chosen to duplicate c into two distinct clauses $c \vee x$ and $c \vee \overline{x}$. We then use $c \vee x$ and $c \vee \overline{x}$ to replace c as a premise of its resolution steps. If necessary, we repeat the same process on clauses $c \vee x$ and/or $c \vee \overline{x}$.

Theorem 2. *Given an unsatisfiable formula ϕ and a regular tree-like resolution refutation P of ϕ, there exists a max-refutation of ϕ containing $O(|P|)$ inference steps.*

3.3 From (Semi-)Tree-Like Resolution Refutations to Max-Refutations

To extend the linear case to tree-like resolution refutations, we simply use a known transformation from any tree-like resolution refutation to a regular tree-like resolution refutation without increasing its size (proved in [Urquhart, 1995]).

Theorem 3. *Given an unsatisfiable formula ϕ and a tree-like resolution refutation P of ϕ, there exists a max-refutation of ϕ containing $O(|P|)$ inference steps.*

To extend our result to semi-tree-like resolution refutations, defined below, we propose an adaptation which relies on the fact that such refutations can be partitioned into two parts where the first part is a read-once sequence of resolutions and the second part is a tree-like resolution refutation.

Definition 6 (semi-tree-like resolution refutation)**.** *A resolution refutation is semi-tree-like if, for any branch of the refutation, at most one clause is non-read-once.*

Theorem 4. *Given an unsatisfiable formula ϕ and a semi-tree-like resolution refutation P of ϕ, there exists a max-refutation of ϕ containing $O(|P|)$ inference steps.*

3.4 From Unrestricted Resolution Refutations to Max-Refutations

To adapt any resolution refutation to a max-refutation, we add a prior transformation to make the initial resolution refutation (semi-)tree-like. To achieve this prior transformation, we iteratively search the proof for the first non-read-once intermediate clause c, and we duplicate the the part of the proof leading to c. Repeating this treatment on intermediary non-read-once clauses forces the resolution refutation to become (semi-)tree-like, even if we accept an exponential blow-up of the size of the formula.

Theorem 5. *Given an unsatisfiable formula ϕ and an unrestricted resolution refutation P of ϕ, there exists a max-refutation of ϕ with $O(2^{\mu(P)} \times |P|)$ inference steps.*

4 MS-Builder & MS-Checker

MS-Builder [Py *et al.*, 2022; Py *et al.*, 2021a] generates certificates for the Max-SAT Problem in the particular form of a Max-SAT-equivalence-preserving transformation from the initial formula into a formula composed of a set of empty clauses and a satisfiable sub-formula. Given an initial formula, MS-Builder iteratively calls a SAT oracle [Biere, 2010] to get a resolution refutation, adapts it for Max-SAT and applies it to the current formula. When the SAT oracle returns that the current formula is now satisfiable (with a model), the algorithm terminates. The complete sequence of transformations generating k empty clauses is a proof that the Max-SAT optimum is at least k while the model is a proof that it is possible to falsify exactly k clauses and therefore that the Max-SAT optimum is k. MS-Builder also works on weighted (partial) formulas. MS-Builder receives a file containing a formula in the standard WCNF format and it returns a certificate. The builder is also coupled with a tool called MS-Checker which verifies the validity of the computed certificates.

Algorithm 1 MS-Builder

Require: Formula ϕ
Ensure: Max-SAT certificate c for ϕ

```
1: (T, opt) ← (∅, 0)
2: while φ is inconsistent do
3:     RP ← compute_RES_refutation(φ)
4:     MRP ← adapt__RES_refutation_for_Max-SAT(RP)
5:     φ ← apply_max-refutation(φ, MRP)
6:     (φ, opt) ← remove_empty_clauses(φ, opt)
7:     T ← concatenate(T, MRP)
8: end while
9: I ← compute_model(φ)
10: return (T, opt, I)
```

On the complete track benchmarks of the 2020 Max-SAT Evaluation [Bacchus *et al.*, 2020], MS-Builder has succeeded to construct full proofs for 163 instances over 576 unweighted (partial) instances and for 144 instances over 600 weighted (partial) instances. In the experiments, a slot of only 1 hour and at most 16 GB of memory was allocated to each instance. More interestingly, MS-Builder has also succeeded to build at least half of the proofs (with respect to the optimum value) of 302 instances over 463 unweighted instances and of 326 instances over 489 weighted instances for which the optimum cost is known. Finally, we report in Table 2 the encountered refutation types during proof building. We observed different behaviours for unweighted and weighted instances. Indeed, while the percentage of read-once and semi-read-once resolution refutations is 83.7 % in the unweighted benchmark, it is only 35.60 % in the weighted benchmark. Such a difference can explain why weighted instances are harder to proove than unweighted ones.

	Unweighted instances		Weighted instances	
	Number	Percentage	Number	Percentage
read-once	169,239	83.7 %	135,594	35.60 %
semi-read-once	24,556	12.1 %	87,748	23.04 %
tree-like regular	2,879	1.4 %	23,612	6.20 %
tree-like	1,795	0.9 %	87,529	22.99 %
unrestricted	3,799	1.9 %	46,337	12.17 %

Table 2: Encountered types of resolution refutations

References

[Abramé and Habet, 2014] André Abramé and Djamal Habet. Ahmaxsat: Description and Evaluation of a Branch and Bound Max-SAT Solver. *J. Satisf. Boolean Model. Comput.*, 9(1):89–128, 2014.

[Achá and Nieuwenhuis, 2014] Roberto Javier Asín Achá and Robert Nieuwenhuis. Curriculum-based course timetabling with SAT and MaxSAT. *Ann. Oper. Res.*, 218(1):71–91, 2014.

[Ansótegui *et al.*, 2009] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. Solving (Weighted) Partial MaxSAT through Satisfiability Testing. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, pages 427–440. Springer, 2009.

[Ansótegui *et al.*, 2013] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. SAT-based MaxSAT algorithms. *Artif. Intell.*, 196:77–105, 2013.

[Atserias and Lauria, 2019] Albert Atserias and Massimo Lauria. Circular (yet sound) proofs. In Mikolás Janota and Inês Lynce, editors, *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, volume 11628 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2019.

[Bacchus *et al.*, 2020] Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. MaxSAT Evaluation 2020. https://maxsat-evaluations.github.io/2020/, 2020. Accessed: 2021-07-11.

[Ben-Sasson *et al.*, 2004] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near Optimal Separation Of Tree-Like And General Resolution. *Comb.*, 24(4):585–603, 2004.

[Biere, 2010] Armin Biere. Booleforce. http://fmv.jku.at/booleforce/, 2010. Accessed: 2020-01-01.

[Bofill *et al.*, 2015] Miquel Bofill, Marc Garcia, Josep Suy, and Mateu Villaret. MaxSAT-Based Scheduling of B2B Meetings. In Laurent Michel, editor, *Integration of AI and OR Techniques in Constraint Programming - 12th International Conference, CPAIOR 2015, Barcelona, Spain, May 18-22, 2015, Proceedings*, volume 9075 of *Lecture Notes in Computer Science*, pages 65–73. Springer, 2015.

[Bonet and Levy, 2020] Maria Luisa Bonet and Jordi Levy. Equivalence Between Systems Stronger Than Resolution. In Luca Pulina and Martina Seidl, editors, *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, volume 12178 of *Lecture Notes in Computer Science*, pages 166–181. Springer, 2020.

[Bonet *et al.*, 2006] Maria Luisa Bonet, Jordi Levy, and Felip Manyà. A Complete Calculus for Max-SAT. In Armin Biere and Carla P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, volume 4121 of *Lecture Notes in Computer Science*, pages 240–251. Springer, 2006.

[Bonet *et al.*, 2007] Maria Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution for Max-SAT. *Artif. Intell.*, 171(8-9):606–618, 2007.

[Cherif *et al.*, 2020] Mohamed Sami Cherif, Djamal Habet, and André Abramé. Understanding the power of Max-SAT resolution through UP-resilience. *Artif. Intell.*, 289:103397, 2020.

[Cherif *et al.*, 2022] Mohamed Sami Cherif, Djamal Habet, and Matthieu Py. From Crossing-Free Resolution to Max-SAT Resolution. In Christine Solnon, editor, *28th International Conference on Principles and Practice of Constraint Programming, CP 2022, July 31 to August 8, 2022, Haifa, Israel*, volume 235 of *LIPIcs*, pages 12:1–12:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[D'Almeida and Grégoire, 2012] Dominique D'Almeida and Éric Grégoire. Model-based diagnosis with default information implemented through MAX-SAT technology. In Chengcui Zhang, James Joshi, Elisa Bertino, and Bhavani Thuraisingham, editors, *IEEE 13th International Conference on Information Reuse & Integration, IRI 2012, Las Vegas, NV, USA, August 8-10, 2012*, pages 33–36. IEEE, 2012.

[Davies and Bacchus, 2011] Jessica Davies and Fahiem Bacchus. Solving MAXSAT by Solving a Sequence of Simpler SAT Instances. In Jimmy Ho-Man Lee, editor, *Principles and Practice of Constraint Programming - CP 2011 - 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011. Proceedings*, volume 6876 of *Lecture Notes in Computer Science*, pages 225–239. Springer, 2011.

[Demirovic and Musliu, 2017] Emir Demirovic and Nysret Musliu. MaxSAT-based large neighborhood search for high school timetabling. *Comput. Oper. Res.*, 78:172–180, 2017.

[Filmus *et al.*, 2020] Yuval Filmus, Meena Mahajan, Gaurav Sood, and Marc Vinyals. MaxSAT Resolution and Subcube Sums. In Luca Pulina and Martina Seidl, editors, *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, volume 12178 of *Lecture Notes in Computer Science*, pages 295–311. Springer, 2020.

[Fu and Malik, 2006] Zhaohui Fu and Sharad Malik. On Solving the Partial MAX-SAT Problem. In Armin Biere and Carla P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, volume 4121 of *Lecture Notes in Computer Science*, pages 252–265. Springer, 2006.

[Guerra and Lynce, 2012] João Guerra and Inês Lynce. Reasoning over Biological Networks Using Maximum Satisfiability. 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 941–956. Springer, 2012.

[Heras and Marques-Silva, 2011] Federico Heras and João Marques-Silva. Read-Once Resolution for Unsatisfiability-Based Max-SAT Algorithms. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 572–577. IJCAI/AAAI, 2011.

[Hertel and Urquhart, 2009] Alexander Hertel and Alasdair Urquhart. Algorithms and Complexity Results for Input and Unit Resolution. *J. Satisf. Boolean Model. Comput.*, 6(1-3):141–164, 2009.

[Ignatiev *et al.*, 2019] Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an Efficient MaxSAT Solver. *J. Satisf. Boolean Model. Comput.*, 11(1):53–64, 2019.

[Iwama and Miyano, 1995] Kazuo Iwama and Eiji Miyano. Intractability of Read-Once Resolution. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*, pages 29–36. IEEE Computer Society, 1995.

[Küegel, 2012] Adrian Küegel. Improved Exact Solver for the Weighted MAX-SAT Problem. In *POS-10. Pragmatics of SAT*, volume 8 of *EPiC Series in Computing*, pages 15–27. EasyChair, 2012.

[Larrosa and Heras, 2005] Javier Larrosa and Federico Heras. Resolution in Max-SAT and its relation to local consistency in weighted CSPs. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 193–198. Professional Book Center, 2005.

[Larrosa and Rollon, 2020a] Javier Larrosa and Emma Rollon. Augmenting the Power of (Partial) MaxSat Resolution with Extension. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 1561–1568. AAAI Press, 2020.

[Larrosa and Rollon, 2020b] Javier Larrosa and Emma Rollon. Towards a Better Understanding of (Partial Weighted) MaxSAT Proof Systems. In Luca Pulina and Martina Seidl, editors, *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, volume 12178 of *Lecture Notes in Computer Science*, pages 218–232. Springer, 2020.

[Larrosa *et al.*, 2008] Javier Larrosa, Federico Heras, and Simon de Givry. A logical approach to efficient Max-SAT solving. *Artificial Intelligence*, 172(2):204–233, 2008.

[Li *et al.*, 2007] Chu Min Li, Felip Manyà, and Jordi Planes. New Inference Rules for Max-SAT. *J. Artif. Intell. Res.*, 30:321–359, 2007.

[Li *et al.*, 2016] Chu Min Li, Felip Manyà, and Joan Ramon Soler. A Clause Tableau Calculus for MaxSAT. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 766–772. IJCAI/AAAI Press, 2016.

[Li *et al.*, 2022] Chu-Min Li, Zhenxing Xu, Jordi Coll, Felip Manyà, Djamal Habet, and Kun He. Boosting branch-and-bound MaxSAT solvers with clause learning. *AI Commun.*, 35(2):131–151, 2022.

[Loveland, 1970] D.W. Loveland. A linear format for resolution. In *Symposium on Automatic Demonstration*, pages 147–162, 1970.

[Manquinho *et al.*, 2009] Vasco M. Manquinho, João P. Marques Silva, and Jordi Planes. Algorithms for Weighted Boolean Optimization. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, pages 495–508. Springer, 2009.

[Manyà *et al.*, 2020] Felip Manyà, Santiago Negrete, Carme Roig, and Joan Ramon Soler. Solving the Team Composition Problem in a Classroom. *Fundam. Informaticae*, 174(1):83–101, 2020.

[Martins *et al.*, 2014] Ruben Martins, Vasco M. Manquinho, and Inês Lynce. Open-WBO: A Modular MaxSAT Solver. In Carsten Sinz and Uwe Egly, editors, *Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8561 of *Lecture Notes in Computer Science*, pages 438–445. Springer, 2014.

[Muise *et al.*, 2016] Christian J. Muise, J. Christopher Beck, and Sheila A. McIlraith. Optimal Partial-Order Plan Relaxation via MaxSAT. *J. Artif. Intell. Res.*, 57:113–149, 2016.

[Narodytska and Bacchus, 2014] Nina Narodytska and Fahiem Bacchus. Maximum Satisfiability Using Core-Guided MaxSAT Resolution. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 2717–2723. AAAI Press, 2014.

[Py *et al.*, 2020] Matthieu Py, Mohamed Sami Cherif, and Djamal Habet. Towards Bridging the Gap Between SAT and Max-SAT Refutations. In *32nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2020, Baltimore, MD, USA, November 9-11, 2020*, pages 137–144. IEEE, 2020.

[Py *et al.*, 2021a] Matthieu Py, Mohamed Sami Cherif, and Djamal Habet. A Proof Builder for Max-SAT. In Chu-Min Li and Felip Manyà, editors, *Theory and Applications of Satisfiability Testing - SAT 2021 - 24th International Conference, Barcelona, Spain, July 5-9, 2021, Proceedings*, volume 12831 of *Lecture Notes in Computer Science*, pages 488–498. Springer, 2021.

[Py *et al.*, 2021b] Matthieu Py, Mohamed Sami Cherif, and Djamal Habet. Computing Max-SAT Refutations using SAT Oracles. In *33rd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2021, Washington, DC, USA, November 1-3, 2021*, pages 404–411. IEEE, 2021.

[Py *et al.*, 2021c] Matthieu Py, Mohamed Sami Cherif, and Djamal Habet. Inferring Clauses and Formulas in Max-SAT. In *33rd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2021, Washington, DC, USA, November 1-3, 2021*, pages 632–639. IEEE, 2021.

[Py *et al.*, 2022] Matthieu Py, Mohamed Sami Cherif, and Djamal Habet. Proofs and certificates for max-sat. *Journal of Artificial Intelligence Research*, 75:1373–1400, 2022.

[Robinson, 1965] John Alan Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *J. ACM*, 12(1):23–41, 1965.

[Silva and Sakallah, 1996] João P. Marques Silva and Karem A. Sakallah. GRASP - a new search algorithm for satisfiability. In Rob A. Rutenbar and Ralph H. J. M. Otten, editors, *Proceedings of the 1996 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 1996, San Jose, CA, USA, November 10-14, 1996*, pages 220–227. IEEE Computer Society / ACM, 1996.

[Urquhart, 1995] Alasdair Urquhart. The complexity of propositional proofs. *Bull. Symb. Log.*, 1(4):425–467, 1995.

[Urquhart, 2011] Alasdair Urquhart. A Near-Optimal Separation of Regular and General Resolution. *SIAM J. Comput.*, 40(1):107–121, 2011.

[Xu *et al.*, 2003] Hui Xu, Rob A. Rutenbar, and Karem A. Sakallah. Sub-SAT: a formulation for relaxed Boolean satisfiability with applications in routing. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 22(6):814–820, 2003.

[Zhang and Bacchus, 2012] Lei Zhang and Fahiem Bacchus. MAXSAT Heuristics for Cost Optimal Planning. In Jörg Hoffmann and Bart Selman, editors, *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. AAAI Press, 2012.