Beyond Core-Guided MaxSAT

Ilario Bonacina
UPC Universitat Politècnica de Catalunya, Spain

Jordi Levy
IIIA, CSIC, Spain

Ion Mikel Liberal
IIIA, CSIC, Spain

Abstract

Several proof systems for MaxSAT have been proposed in the literature, including MaxSAT resolution and, more recently, systems based on polynomial calculus and tableaux. Although these systems are sound and complete and have varying strengths, they fail to capture the specific inferential strategies used by practical MaxSAT solvers, particularly those used in core-guided approaches. As a result, a formula that is hard to prove in these proof systems may not be hard for a solver, and vice versa.

In this paper, we describe a new proof system for MaxSAT, the *Comparator Calculus* (CC), which models the inferential strategies used in core-guided MaxSAT solvers. We adapt it for partial MaxSAT and prove soundness and completeness in both settings. Based on this formalism, we introduce CSat, a novel MaxSAT solver prototype that, while not core-guided, employs SAT solver calls to remove unsatisfiable soft clauses. Experiments on random 2-CNF instances demonstrate that this scheme avoids the phase-transition slowdown observed in core-guided solvers near optimality certification.

2012 ACM Subject Classification Theory of computation → Proof complexity; Mathematics of computing → Solvers

Keywords and phrases MaxSAT, Proof Systems, Solvers, Optimization

1 Introduction

MaxSAT is the optimization version of the SAT problem. Unlike SAT, which asks whether a CNF formula is satisfiable or not, in MaxSAT we are interested in finding the maximum number of clauses that are satisfiable. Roughly speaking, there are three families of MaxSAT solvers in the literature: *branch-and-bound* [11, 12], *core-guided* [6, 2, 14, 13] , and *implicit hitting set* [5].Whereas branch-and-bound MaxSAT solvers represent the state-of-the-art for random formulas, core-guided and implicit hitting set solvers are preferred for industrial instances. Recently, Ihalainen et al. [8] showed how core-guided and implicit hitting set solvers can be seen as particular instances of a more general scheme. In both cases, these solvers call a SAT solver (in some cases with additional assumptions) to check the satisfiability of a CNF formula or obtain unsatisfiable subsets of clauses (cores) to guide the search toward an optimal solution. Recent advances have focused on certifying the correctness of core-guided MaxSAT solvers, ensuring their reliability in practical applications [3] and the integrity of preprocessing steps in MaxSAT solvers [9].

Proof systems for MaxSAT include MaxSAT resolution [4], and versions of the polynomial calculus and Tableaux. However, they all fail to model the inferential strategies practical MaxSAT solvers use. In this work, we show how a simple proof system, the *Comparator Calculus* (CC), can model the type of calls core-guided solvers make to a SAT solver. A relevant feature of the Comparator Calculus is that it can also model uses of SAT solvers that differ significantly from core extraction. We exemplify this by showing an algorithm (CSat) for MaxSAT, whose behavior can be modeled by the Comparator Calculus. Despite relying on SAT solver calls, it does not follow the core-guided approach.

The Comparator Calculus consists essentially of two substitution rules: the *Comparator* (COMP) rule and the *Contradiction* (CONTR) rule. Without going into technical details, the COMP rule substitutes two formulas by their conjunction and disjunction, while the contradiction rule substitutes an unsatisfiable formula with an immediate contradiction. In this case, to ensure the rule is polynomially checkable, the witness of the unsatisfiability is given by a refutation in some propositional proof system. This rule aims to capture the use of SAT solvers for solving MaxSAT.

The COMP rule is intended to capture the nature of the networks that are usually used to encode the cardinality constraints that come up in runs of core-guided solvers. Indeed, CC polynomially simulates the behavior of core-guided MaxSAT solvers such as Fu&Malik and OLL when the cardinality constraints are encoded using sorting networks, as is typically the case (see Section 4).

Core-guided MaxSAT solvers were first introduced by Fu and Malik in [6], where the authors proposed a solver for partial MaxSAT that can be informally described as follows. The algorithm Fu&Malik calls a SAT solver with all the clauses (hard and soft) and takes advantage of the unsatisfiable set of clauses returned by the SAT solver by *weakening* the clauses in the core with new variables and imposing the condition that only one of those new variables must be falsified (cardinality constraint). Then, it repeats the same procedure until it gets a satisfiable formula. This algorithm is guided by the unsatisfiable sets of clauses returned by the SAT solvers. After this, several modifications and refinements of Fu&Malik's scheme were proposed and, roughly speaking, the main differences lie in how these different approaches encoded and used cardinality constraints [2, 13, 5]. Nowadays, the OLL algorithm [13, 7] is a good representative of the state-of-the-art MaxSAT solvers.

In this work, we present an algorithm for MaxSAT (CSat), which uses calls to a SAT solver but cannot be classified as *core-guided*. CSat works with a partial MaxSAT instance with unitary soft clauses. Initially, it obtains several satisfying assignments (models) of the hard part of the problem. Then, instead of searching for cores in the soft part, it tries to *infer* them. Taking advantage of the models, it constructs binary comparator circuits between soft clauses. When a soft clause is found that is falsified in all current models (a *candidate*), the SAT solver is called to certify its unsatisfiability. Depending on the result, the candidate is either removed, or the model returned by the SAT solver is added to the set of models. The algorithm terminates when a model satisfying all remaining soft clauses is obtained.

On the theoretical side, the definition of a not-too-strong proof system that simulates the behavior of practical MaxSAT solvers opens up the possibility of applying tools from proof complexity. For instance, to derive concrete performance limits for specific classes of formulas or encodings, such as cardinality constraints or random CNF instances.

Structure of the article

In Section 2 we recall all the necessary preliminaries for this work. Section 3 contains the definition of the comparator calculus, its soundness and completeness. Section 4 contains some remarks on the connection between the comparator calculus and core-based MaxSAT solvers. Section 5 contains the high-level description and the pseudo-code of CSat. Finally, in Section 6, we make some concluding remarks.

2 Preliminaries

In this section, we recall standard notions and notations used throughout the paper. Let X be a set of Boolean variables. A *literal* ℓ is a variable x from X or its negation $\neg x$.

Propositional formulas are constructed recursively from literals using conjunctions ($\wedge$) and disjunctions ($\vee$). In particular, a *clause* is a disjunction of literals, and a formula in CNF is a conjunction of clauses. We denote the empty clause with $\perp$.

A (total) *assignment* is a mapping $\alpha : X \to \{0,1\}$. We extend assignments to propositional formulas in the usual way: setting $\alpha(\neg x) = 1 - \alpha(x)$, $\alpha(A \vee B) = \max\{\alpha(A), \alpha(B)\}$, and $\alpha(A \wedge B) = \min\{\alpha(A), \alpha(B)\}$, together with all the properties of classical logic ($1 \vee C = 1$, $0 \vee C = C$, etc). An assignment α *satisfies* a multi-set of formulas Σ (noted $\alpha \models \Sigma$) if, for every formula $F \in \Sigma$, $\alpha(F) = 1$. The multi-set Σ is *satisfiable* if there exists an assignment α such that $\alpha \models \Sigma$. We say that α is a *model* of Σ. Otherwise, Σ is *unsatisfiable*. Any subset of Σ which is unsatisfiable is an (unsatisfiable) *core*. Given a multi-set of formulas Σ and an assignment α, the *cost* of Σ under α is

$$\mathrm{cost}_\alpha(\Sigma) = \sum_{F \in \Sigma} (1 - \alpha(F)) \ ,$$

i.e. $\mathrm{cost}_\alpha(\Sigma)$ is the number of formulas in Σ falsified by α.

We consider (partial) MaxSAT instances of the form $\mathcal{F} = \mathcal{H} \cup \mathcal{S}$ where $\mathcal{H}$ is a set of *hard* formulas and $\mathcal{S}$ is a multi-set of soft formulas. The *cost* of $\mathcal{F}$ is

$$\mathrm{cost}(\mathcal{F}) = \mathrm{cost}_{\mathcal{H}}(\mathcal{S}) = \min_{\alpha \,:\, \alpha \models \mathcal{H}} \mathrm{cost}_\alpha(\mathcal{S}),$$

i.e., $\mathrm{cost}(\mathcal{F})$ is the minimum number of falsified formulas in $\mathcal{S}$ by any assignment satisfying all the hard formulas in $\mathcal{H}$. W.l.o.g. we can assume the soft formulas to be literals $\neg b_1, \dots, \neg b_m$: indeed, any soft formula $F_i \in \mathcal{S}$ can be equivalently represented by a hard formula $F_i \vee b_i$ and a soft literal $\neg b_i$ (where b_i is a fresh new variable). This is the *blocking literals encoding* and it is how usually MaxSAT instances are encoded in practice.

Notice that, in SAT solving, formulas are assumed to be sets or conjunctions of clauses, i.e., in CNF. Similarly, in MaxSAT solving, formulas are assumed to be multi-sets of clauses. Here, we deal with arbitrary formulas, and MaxSAT problems are multi-sets of arbitrary formulas. Moreover, we also deal with sets of assignments $\mathcal{A}$.

3 The Comparator Calculus

In this section, we introduce the *Comparator Calculus* (CC) for MaxSAT. For simplicity, we only consider unweighted formulas, although the calculus can be easily extended to the weighted case by adding *fold* and *unfold* rules to it (see [4]). First, we describe the calculus on multi-sets of soft formulas and then we show the minor adaptations for the calculus to deal with hard and soft formulas.

The *Comparator Calculus* manipulates multi-sets of propositional formulas with two substitution rules: the *comparator* (COMP) rule and *contradiction* (CONTR) rule. That is, the following inference rules are applied by replacing the premises with the conclusions:

$$\frac{A \quad B}{A \wedge B \quad A \vee B}\ (\textsc{comp}) \qquad \frac{A}{\perp}\ \{A \text{ unsatisfiable}\}\ (\textsc{contr}) \ . \tag{1}$$

In the CONTR rule, there is a *side condition* that requires the unsatisfiability of the premise A in order to apply the rule. To certify each application of this rule we need to provide for each of them a refutation of the premise A (or a suitable encoding of A) in some propositional proof system P. For instance, we could consider as P the propositional proof system Frege (see for instance [10]), or we could provide the proof of unsatisfiability of a suitable encoding enc(A) produced by a SAT solver. Since most SAT solvers we could use for certifying the unsatisfiability of A only work on CNF formulas, in this case, the use of an encoding becomes necessary.

Notice that both the COMP rule and the CONTR rule preserve the cost: for every assignment α we have

$$\mathrm{cost}_\alpha(\{A, B\}) = \mathrm{cost}_\alpha(\{A \wedge B, A \vee B\}) \quad \text{and} \quad \mathrm{cost}_\alpha(A') = \mathrm{cost}_\alpha(\bot) \tag{2}$$

when A' is unsatisfiable.

▶ **Example 3.1.** Given the (multi-)set of formulas $\{x_1 \wedge x_2,\ \neg x_1 \wedge x_3,\ \neg x_2 \wedge \neg x_3\}$, we can perform the following substitutions using the COMP and CONTR rules:

$$\begin{array}{cl}
x_1 \wedge x_2, \quad \neg x_1 \wedge x_3, \quad \neg x_2 \wedge \neg x_3 & \\
\hline
x_1 \wedge x_2 \wedge \neg x_1 \wedge x_3, \quad (x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3), \quad \neg x_2 \wedge \neg x_3 & \text{COMP} \\
\hline
\bot, \quad (x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3), \quad \neg x_2 \wedge \neg x_3 & \text{CONTR} \\
\hline
\bot, \quad ((x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3)) \wedge \neg x_2 \wedge \neg x_3, \quad (x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3) & \text{COMP} \\
\hline
\bot, \quad \bot, \quad (x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3) & \text{CONTR}
\end{array}$$

▶ **Definition 3.2** (Comparator Calculus). *Let $\mathcal{F}$ and Γ be multi-sets of (soft) formulas and P a propositional proof system. A* derivation *of Γ from $\mathcal{F}$ in the* Comparator Calculus (CC_P) *is a sequence of multi-sets $\pi = \langle M_0, \dots, M_s\rangle$ such that $M_0 = \mathcal{F}$, $\Gamma \subseteq M_s$, and for each i one of the following two cases occurs:*

1. *there are formulas $A, B \in M_i$ such that*

$$M_{i+1} = (M_i \setminus \{A, B\}) \cup \{A \wedge B, A \vee B\}\ ,$$

2. *there is an unsatisfiable formula $A \in M_i$ and*

$$M_{i+1} = (M_i \setminus \{A\}) \cup \{\bot\}\ .$$

In this case, there is also attached a P-proof of the unsatisfiability of A.

In other words, for every $i = 0, \dots, s-1$, the multi-set M_{i+1} is obtained from M_i applying either the COMP rule or the CONTR rule from eq. (1) *to formulas in M_i. The size of the derivation is the total number of bits needed to write down the derivation, including the P-proofs certifying the validity of the CONTR steps.*

For a lighter notation we usually omit the proof system P in the notation for CC. We show first that CC is sound and complete.

▶ **Theorem 3.3** (Soundness). *Given $\mathcal{F}$ and a CC derivation $\pi = \langle M_1, \dots, M_s\rangle$ with $M_1 = \mathcal{F}$ and $\{\bot, \overset{k}{\dots}, \bot\} \subseteq M_s$, it holds that* $\mathrm{cost}(\mathcal{F}) \geq k$.

▶ **Theorem 3.4** (Completeness). *For every multi-set of formulas $\mathcal{F}$ with $\mathrm{cost}(\mathcal{F}) = k$ there exists a CC derivation $\pi = \langle M_1, \dots, M_s\rangle$ with $M_1 = \mathcal{F}$ and $\{\bot, \overset{k}{\dots}, \bot\} \subseteq M_s$.*

For partial MaxSAT, this ideas can be easily adapted if we consider sequences of pairs of multisets.

4 Connection with Core-Based MaxSAT Solvers

In this section, we show how CC simulates the OLL and a version of the Fu&Malik algorithm. In particular, a version of Fu&Malik with symmetry breaking is considered, which better adapts to CC. As a consequence, size lower-bounds for CC imply time lower-bounds for those particular core-guided MaxSAT solvers.

4.1 The Proof System Behind the OLL Algorithm

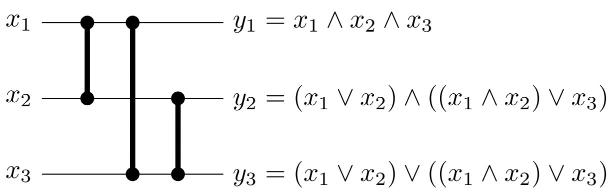
Given a MaxSAT instance $\mathcal{F}$, the OLL algorithm replaces the clauses in a core $\mathcal{C}$ by *soft* cardinality constraints. If we focus on the unweighted case and use formulas instead of clauses, this is equivalent to applying the rule:

$$\frac{A_1 \quad \cdots \quad A_r}{\bot \quad \mathrm{enc}(\sum_{i=1}^{r} A_i \geq r-1) \quad \cdots \quad \mathrm{enc}(\sum_{i=1}^{r} A_i \geq 1)} \{A_1 \wedge \cdots \wedge A_r \text{ unsatisfiable}\} \text{ (OLL)} \tag{3}$$

where the unsatisfiability of $A_1 \wedge \cdots \wedge A_r$ is certified in a propositional proof system P and $\mathrm{enc}(\sum_{i=1}^{r} A_i \geq j)$ is a propositional formula equivalent to the cardinality constraint $\sum_{i=1}^{r} A_i \geq j$. Notice that if $A_1 \wedge \cdots \wedge A_r$ is unsatisfiable then there is no assignment α satisfying $\mathrm{enc}(\sum_{i=1}^{r} \alpha(A_i) \geq r)$. The OLL rule in eq. (3) is cost-preserving. To emphasize the underlying propositional proof system P, we use the notation *OLL_P calculus* to denote the calculus that use the rule above.

Among the various methods to encode cardinality constraints, using sorting networks is one of the most commonly used in current solvers. In Example 4.1, we show how sorting networks can be used to obtain a particular instance of the OLL rule for $r = 3$.

▶ **Example 4.1.** In the case $r = 3$, an optimal sorting network is:



In particular y_1 is equivalent to $x_1 + x_2 + x_3 \geq 3$, y_2 is equivalent to $x_1 + x_2 + x_3 \geq 2$ and y_3 is equivalent to $x_1 + x_2 + x_3 \geq 1$. Therefore, the OLL rule for $r = 3$ can be written as:

$$\frac{A_1 \quad A_2 \quad A_3}{\bot \quad (A_1 \vee A_2) \wedge ((A_1 \wedge A_2) \vee A_3) \quad (A_1 \vee A_2) \vee ((A_1 \wedge A_2) \vee A_3)} \{A_1 \wedge A_2 \wedge A_3 \text{ unsatisfiable}\}$$

The main idea behind this method is as follows. Let $y_1, \dots, y_r = \mathsf{SN}(x_1, \dots, x_r)$ be a sorting network with inputs $x_1, \dots, x_r$ and outputs $y_1, \dots, y_r$, the assignment α satisfies the constraint $x_1 + \cdots + x_r \geq i$ if, and only if, the output y_{r-i+1} in $\mathsf{SN}(\alpha(x_1), \dots, \alpha(x_r))$ is one.

The Partial MaxSAT version of the OLL rule, using sorting networks, is:

$$\frac{x_1 \quad \cdots \quad x_r}{\bot \quad y_2 \cdots y_r \quad \mathsf{CNF}(y_1, \dots, y_r = \mathsf{SN}(x_1, \dots, x_r))_\infty} \{\mathcal{H} \cup \{x_1, \dots, x_r\} \text{ unsatisfiable}\} \tag{4}$$

This calculus is the one modeling the basic OLL algorithm, which uses calls to a SAT solver to find the core. To do this, the formulas describing the sorting network in eq. (4) need to be CNFs (for example via a Tseitin encoding),and are added as hard clauses.

Sorting networks with n inputs can be defined using $\mathcal{O}(n \log n)$ comparators and with depth $\mathcal{O}(\log n)$ [1], where a comparator is a circuit that takes input x, y and has output $x \wedge y, x \vee y$, like our COMP rule. Defining these networks with comparator circuits fits very well in the context of CC. In particular, the rule for $r = 3$ from above can be simulated in CC with 3 applications of the COMP rule (see Example 4.1) because there exists a sorting network with $n = 3$ that only uses 3 comparators. In general, we have the following result.

▶ **Theorem 4.2.** CC_P *polynomially simulates the* OLL_P *calculus using sorting networks to encode soft cardinality constraints.*

4.2 The Proof System Behind Fu&Malik Algorithm

In the Fu&Malik algorithm, every time a core $\{A_1, \ldots, A_r\}$ is found (by calling a SAT solver on the soft clauses), all the soft clauses are relaxed adding a fresh variable x_i for each $i \leq r$ and a hard constraint $x_1 + \cdots + x_r \leq 1$. In addition, an empty clause is added to ensure that only one of the original soft clauses is *repaired*. Therefore, in terms of formulas, the derivation can be modeled as:

$$\frac{A_1 \cdots A_r}{\bot \quad A_1 \vee x_1 \ \cdots \ A_r \vee x_r \quad \mathrm{enc}\left(\sum_{i=1}^{r} x_i \leq 1\right)_\infty} \{A_1 \wedge \cdots \wedge A_r \text{ unsatisfiable}\} \ (\textsc{Fu\&Malik}) \tag{5}$$

As for the OLL calculus we call the calculus using the rule above the *Fu&Malik calculus* and this calculus clearly models the Fu&Malik algorithm.

In this case, the introduction of hard constraints avoids the possibility of defining a calculus where all formulas are soft. Moreover, this encoding introduces a lot of *symmetries*. If an assignment falsifies several A_i's, we can decide to set any of their x_i's to true, getting several optimal assignments.

In other words, we set x_i to true only if all previous formulas $A_1, \ldots, A_{i-1}$ are already satisfied. Therefore, the soft clause $A_i \vee x_i$ is equivalent to the formula $A_i \vee (A_1 \wedge \cdots \wedge A_{i-1})$. With this restriction, the derivation results in:

$$\frac{A_1 \cdots A_r}{\bot \quad A_2 \vee A_1 \quad A_3 \vee (A_1 \wedge A_2) \ \cdots \ A_r \vee (A_1 \wedge \cdots \wedge A_{r-1})} \begin{array}{l}\{A_1 \wedge \cdots \wedge A_r \text{ unsatisfiable}\} \\ (\textsc{Fu\&Malik Sym. Break})\end{array} \tag{6}$$

We call the calculus using the rule above *Fu&Malik calculus with symmetry breaking*. As usual, we use a subscript P to denote the propositional proof system used to certify unsatisfiability.

Notice that with this restriction on which soft clause we repair, we no longer need the cardinality restriction ensuring that only one is repaired. The first soft formula $A_1 \vee x_1$ is always repaired. Therefore, it is a tautology and can be removed from the conclusions.

▶ **Example 4.3.** For the case $r = 3$, this derivation can be simulated with 2 applications of COMP and one of CONTR:

$$\dfrac{\dfrac{\dfrac{A_1 \quad A_2 \quad A_3}{A_1 \wedge A_2 \quad A_1 \vee A_2 \quad A_3}\ \textsc{comp}}{A_1 \wedge A_2 \wedge A_3 \quad (A_1 \wedge A_2) \vee A_3 \quad A_1 \vee A_2}\ \textsc{comp}}{\bot \quad (A_1 \wedge A_2) \vee A_3 \quad A_1 \vee A_2}\ \textsc{contr} \quad \{A_1 \wedge A_2 \wedge A_3 \text{ unsatisfiable}\}$$

Algorithm 1 The CSat algorithm

```
Input: A set of hard clauses H and a multi-set of unary soft clauses S
Output: cost_H(S) and an optimal assignment α s.t. α ⊨ H and cost_α(S) = cost_H(S)
  A ← {α | sat, α ← SAT(H)}                ▷ Set of assignments s.t. ∀α ∈ A.α ⊨ H
  lb = 0                                    ▷ Lower bound for cost_H(S)
  rub = min_{α∈A} {cost_α(S)}               ▷ Remaining upper bound for cost_H(S)
  while rub > 0 do
     if ∃c ∈ S ∀α ∈ A, α(c) = 0 then        ▷ Exists a candidate c of empty clause
        sat, α ← SAT(H ∪ {c})
        if sat then                         ▷ Introduce new model in A
           A ← A ∪ {α}
           rub ← min {rub, cost_α(S)}
        else                                ▷ Apply CONTR rule
           lb ← lb + 1
           rub ← rub − 1
           S ← S \ {c}
        end if
     else                                   ▷ Apply COMP rule
        b1, b2 ← heuristic(H, S, A)
        H ← H ∪ CNF(x ↔ b1 ∧ b2, y ↔ b1 ∨ b2)   ▷ x and y are fresh variables
        S ← S \ {b1, b2} ∪ {x, y}
     end if
  end while
  return lb and α ∈ A s.t. cost_α(S) = 0
```

The idea behind the simulation in the previous example is generalized in the following theorem.

▶ **Theorem 4.4.** CC_P *linearly simulates the* $Fu\&Malik_P$ *calculus with symmetry breaking.*

5 A New SAT-based Algorithm for MaxSAT

This section presents a new algorithm, called CSat, for solving (weighted) Partial MaxSAT. For the sake of clarity, we only cover the unweighted case, but a minor adaptation of the discussion would work for Weighted Partial MaxSAT as well. CSat, as other recent MaxSAT solvers, takes as input a partial MaxSAT instance encoded with blocking variables, that is, a multi-set $\mathcal{H}$ of hard clauses and a multi-set $\mathcal{S}$ of soft unary clauses.

For the description of CSat we refer to the pseudo-code in Algorithm 1.

▶ **Theorem 5.1** (Correctness). *The algorithm CSat on input* $\mathcal{F} = \mathcal{H} \cup \mathcal{S}$*, if it terminates, returns* $\mathrm{cost}_{\mathcal{H}}(\mathcal{S})$*.*

In general, it is not true that CSat always terminates. It depends on the heuristic subroutine we use to select the two soft clauses to apply COMP. Suppose, for instance, that heuristic were selecting always the last two soft clauses. Since the application of COMP to the formulas $A \wedge B$ and $A \vee B$ results in $(A \wedge B) \wedge (A \vee B) = A \wedge B$ and $(A \wedge B) \vee (A \vee B) = A \vee B$, we would enter an infinite loop.

Heuristic. The pseudo-code of the heuristic used in the implementation of CSat is described in Algorithm 2. This heuristic ensures termination of the CSat algorithm.

For a set of assignments $\mathcal{A}$ and a formula F, let $\text{count}_{\mathcal{A}}(F)$ be the number of assignments in $\mathcal{A}$ falsifying F, in other words

$$\text{count}_{\mathcal{A}}(F) = |\{\alpha \in \mathcal{A} : \alpha(F) = 0\}| = \sum_{\alpha \in \mathcal{A}} \text{cost}_{\alpha}(F) \ .$$

Algorithm 2 The heuristic function

Input: $\mathcal{S}$, $\mathcal{A}$
Output: $x, y \in \mathcal{S}$

$B_1 = \arg\max_{x \in \mathcal{S}} \ \text{count}_{\mathcal{A}}(x)$

$B_2 = \arg\max_{x \in B_1, y \in \mathcal{S} \setminus \{x\}} \ \text{count}_{\mathcal{A}}(x \wedge y)$

$B_3 = \arg\max_{(x,y) \in B_2} \text{count}_{\mathcal{A}}(x \vee y)$

return any $(x, y) \in B_3$

▶ **Theorem 5.2.** *The algorithm CSat with the heuristic in Algorithm 2 always terminates in* $\mathcal{O}(|\mathcal{S}|^2)$ *iterations.*

It is worthy of mention that we tested CSat on random Max2SAT instances and compared it to (our implementations of) other competitive solvers. In general, CSat seems to exhibit a better asymptotic behavior, but it is still not at the level of the others.

6 Conclusions and Further Work

This paper introduced the Comparator Calculus (CC), a sound and complete proof system for MaxSAT designed to reflect the inferential behavior of core-guided MaxSAT solvers and which is built from two simple cost-preserving substitution rules. We show how CC can simulate key solving strategies used by algorithms such as Fu&Malik and OLL. As a consequence, size lower-bounds of proofs in CC imply time lower-bounds in the mentioned algorithms. In contrast to prior approaches rooted in resolution or algebraic methods, CC offers a lightweight and practically motivated model that can serve both as a tool for theoretical analysis and as inspiration for new solver design.

We also proposed a novel SAT-based MaxSAT solver prototype, CSat, that departs from the core-guided paradigm. Rather than relying on unsatisfiable core extraction, CSat incrementally constructs candidate formulas and tests their consistency. This approach helps to mitigate the known limitations of core-guided solvers near the optimality certification.

Future work includes exploring proof complexity lower bounds within the calculus and improving the efficiency of CSat's heuristics (aiming for quasi-linear COMP rule complexity, similar to optimal sorting networks). It also remains open to study how different CNF encodings of formulas in the CONTR rule affect derivation size and tractability. More broadly, this line of work aims to bridge the gap between theoretical proof systems and the behavior of modern MaxSAT solvers, offering new avenues for both understanding and improving SAT-based optimization.

References

1 Miklós Ajtai, János Komlós, and Endre Szemerédi. An $O(n \log n)$ sorting network. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–9, 1983. doi:10.1145/800061.808726.

2 Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. SAT-based MaxSAT algorithms. *Artif. Intell.*, 196:77–105, 2013. doi:10.1016/J.ARTINT.2013.01.002.

3 Jeremias Berg, Bart Bogaerts, Jakob Nordström, Andy Oertel, and Dieter Vandesande. Certified core-guided MaxSAT solving. In *Proceedings of the 29th International Conference on Automated Deduction (CADE)*, pages 1–22, 2023. doi:10.1007/978-3-031-38499-8_1.

4 Maria Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution for Max-SAT. *Artif. Intell.*, 171(8-9):606–618, 2007. URL: https://doi.org/10.1016/j.artint.2007.03.001, doi:10.1016/J.ARTINT.2007.03.001.

5 Jessica Davies and Fahiem Bacchus. Solving MAXSAT by solving a sequence of simpler SAT instances. In *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP)*, pages 225–239, 2011. doi:10.1007/978-3-642-23786-7_19.

6 Zhaohui Fu and Sharad Malik. On solving the partial MAX-SAT problem. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 252–265, 2006. doi:10.1007/11814948_25.

7 Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an efficient MaxSAT solver. *J. Satisf. Boolean Model. Comput.*, 11(1):53–64, 2019. doi:10.3233/SAT190116.

8 Hannes Ihalainen, Jeremias Berg, and Matti Järvisalo. Unifying core-guided and implicit hitting set based optimization. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1935–1943, 2023. doi:10.24963/IJCAI.2023/215.

9 Hannes Ihalainen, Andy Oertel, Yong Kiam Tan, Jeremias Berg, Matti Järvisalo, Magnus O. Myreen, and Jakob Nordström. Certified MaxSAT preprocessing. In *Proceedings of the 12th International Joint Conference on Automated Reasoning (IJCAR)*, pages 396–418, 2024. doi:10.1007/978-3-031-63498-7_24.

10 Jan Krajíček. *Proof Complexity.* Cambridge University Press, 2019.

11 Chu Min Li, Felip Manyà, Nouredine Ould Mohamedou, and Jordi Planes. Resolution-based lower bounds in MaxSAT. *Constraints An Int. J.*, 15(4):456–484, 2010. doi:10.1007/S10601-010-9097-9.

12 Chu-Min Li, Zhenxing Xu, Jordi Coll, Felip Manyà, Djamal Habet, and Kun He. Combining clause learning and branch and bound for MaxSAT. In *27th International Conference on Principles and Practice of Constraint Programming, CP 2021*, volume 210 of *LIPIcs*, pages 38:1–38:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.CP.2021.38.

13 António Morgado, Carmine Dodaro, and João Marques-Silva. Core-guided MaxSAT with soft cardinality constraints. In *Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming (CP)*, pages 564–573, 2014. doi:10.1007/978-3-319-10428-7_41.

14 António Morgado, Federico Heras, Mark H. Liffiton, Jordi Planes, and João Marques-Silva. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints An Int. J.*, 18(4):478–534, 2013. doi:10.1007/S10601-013-9146-2.

A Appendix

A.1 The comparator Calculus

▶ **Theorem A.1** (Soundness). *Given $\mathcal{F}$ and a* CC *derivation $\pi = \langle M_1, \dots, M_s \rangle$ with $M_1 = \mathcal{F}$ and $\{\bot, \overset{k}{\dots}, \bot\} \subseteq M_s$, it holds that $\mathrm{cost}(\mathcal{F}) \geq k$.*

Proof. Since clearly $\mathrm{cost}(M_s) \geq k$, it is enough to prove by induction on i that $\mathrm{cost}(\mathcal{F}) = \mathrm{cost}(M_i)$.

The base case $i = 1$ is trivial. For the induction step, we have two cases: M_{i+1} is obtained from M_i using the COMP rule or the CONTR rule.

If M_{i+1} is obtained from M_i using the COMP rule, then there are some formulas A and B in M_i such that $M_{i+1} = (M_i \setminus \{A, B\}) \cup \{A \wedge B, A \vee B\}$. Eq. (2) implies that for every assignment α, $\mathrm{cost}_\alpha(M_i) = \mathrm{cost}_\alpha(M_{i+1})$, hence $\mathrm{cost}(F) = \mathrm{cost}(M_i) = \mathrm{cost}(M_{i+1})$.

If M_{i+1} is obtained from M_i using the CONTR rule, let A be the formula for which we apply the rule and denote with M'_i the multi-set $M_i \setminus \{A\}$. Since A is unsatisfiable, for every assignment α we have $\mathrm{cost}_\alpha(M_i) = 1 + \mathrm{cost}_\alpha(M'_i)$. But, $M_{i+1} = M'_i \cup \{\bot\}$ and $\mathrm{cost}_\alpha(M_{i+1}) = \mathrm{cost}_\alpha(M'_i) + 1$. Hence, again, $\mathrm{cost}(\mathcal{F}) = \mathrm{cost}(M_i) = \mathrm{cost}(M_{i+1})$. ◀

▶ **Theorem A.2** (Completeness). *For every multi-set of formulas $\mathcal{F}$ with $\mathrm{cost}(\mathcal{F}) = k$ there exists a* CC *derivation $\pi = \langle M_1, \dots, M_s \rangle$ with $M_1 = \mathcal{F}$ and $\{\bot, \overset{k}{\dots}, \bot\} \subseteq M_s$.*

Proof. Let $\mathcal{F} = \{F_1, \dots, F_m\}$. We show the result by induction on k. If $k = 0$, i.e. $\mathcal{F}$ is satisfiable there is nothing to do. Otherwise, suppose that $\mathrm{cost}(\mathcal{F}) = k$. We construct a sequence of multi-sets $\langle M_1, \dots, M_{m+1} \rangle$, where $M_1 = \mathcal{F}$ and

$$M_s = \left\{ \bigwedge_{i=1}^{s} F_i \right\} \cup \left\{ F_i \vee \bigwedge_{j=1}^{i-1} F_j \;:\; i \in \{2, \dots, s\} \right\} \cup \{F_j \;:\; j \in \{s+1, \dots, m\}\}$$

for every $s \in \{2, \dots, m\}$. We can obtain M_{s+1} from M_s applying the COMP rule to F_{s+1} and $\bigwedge_{i=1}^{s} F_i$. In other words, the sequence $\langle M_1, \dots, M_m \rangle$ is a CC derivation of $F_1 \wedge F_2 \wedge \cdots \wedge F_m$ together with a multi-set of $m-1$ other formulas Γ. Since $\mathcal{F}$ is unsatisfiable we can apply the CONTR rule to $F_1 \wedge F_2 \wedge \cdots \wedge F_m$ in M_m and obtain the multi-set $M_{m+1} = \{\bot\} \cup \Gamma$. By the soundness of CC (Theorem A.1), we know that $\mathrm{cost}(\Gamma) = k - 1$; hence, by induction hypothesis, there is a CC derivation $\langle N_1, \dots, N_t \rangle$ of $\bot, \overset{k-1}{\dots}, \bot$ from Γ. Putting everything together we have that $\langle M_1, \dots, M_m, M_{m+1}, N_1 \cup \{\bot\}, \dots, N_t \cup \{\bot\} \rangle$ is a CC derivation of $\bot, \overset{k}{\dots}, \bot$ from $\mathcal{F}$. ◀

A.2 The Comparator Calculus on Partial MaxSAT

We consider partial MaxSAT instances of the form $\mathcal{F} = \mathcal{H} \cup \mathcal{S}$, where $\mathcal{H}$ is a set of hard formulas and $\mathcal{S}$ is a multi-set of soft formulas and w.l.o.g. we assume $\mathcal{S}$ only contains literals (or $\bot$s) and we denote hard clauses using a suffix ∞, i.e. to denote that a clause C is hard we write it as C_∞. In adapting the rules of CC to this context, the COMP rule becomes

$$\frac{\ell_1 \quad \ell_2}{y_1 \quad y_2 \quad \mathsf{CNF}(y_1 \leftrightarrow \ell_1 \wedge \ell_2,\; y_2 \leftrightarrow \ell_1 \vee \ell_2)_\infty} \text{(COMP')}$$

where ℓ_1, ℓ_2 are soft literals, y_1, y_2 are fresh new variables (also soft literals) and

$$\mathsf{CNF}(y_1 \leftrightarrow \ell_1 \wedge \ell_2,\; y_2 \leftrightarrow \ell_1 \vee \ell_2)_\infty$$

denotes the natural CNF encoding of $y_1 \leftrightarrow \ell_1 \wedge \ell_2$ and $y_2 \leftrightarrow \ell_1 \vee \ell_2$ as hard clauses.

▶ **Definition A.3** (CC in partial MaxSAT). *A* CC *derivation from $\mathcal{F} = \mathcal{H} \cup \mathcal{S}$ is a sequence of pairs $\pi = \langle(\mathcal{H}_1; \mathcal{S}_1), \dots, (\mathcal{H}_s; \mathcal{S}_s)\rangle$ where $\mathcal{H}_1 = \mathcal{H}$, $\mathcal{S}_1 = \mathcal{S}$ and for each i one of the following two cases occurs:*

1. *If $\mathcal{S}_i = \mathcal{S}'_i \cup \{\ell_1, \ell_2\}$ then introduce two fresh new variables y_1, y_2 and let*

$$\mathcal{H}_{i+1} = \mathcal{H}_i \cup \{\mathsf{CNF}(y_1 \leftrightarrow \ell_1 \wedge \ell_2,\ y_2 \leftrightarrow \ell_1 \vee \ell_2)\} \quad \text{and} \quad \mathcal{S}_{i+1} = \mathcal{S}'_i \cup \{y_1, y_2\}\ .$$

2. *If $\mathcal{S}_i = \mathcal{S}'_i \cup \{\ell\}$ and $\mathcal{H}_i \cup \{\ell\}$ is unsatisfiable (which is certified by a refutation in a proof system P), then*

$$\mathcal{H}_{i+1} = \mathcal{H}_i \quad \text{and} \quad \mathcal{S}_{i+1} = \{\bot\} \cup \mathcal{S}'_i\ .$$

The option in Item 1 corresponds to the COMP *rule, while the option in Item 2 corresponds to an application of the* CONTR *rule. The* size *of a proof is the total number of bits needed to write the derivation, including the refutations in P appearing in Item 2. Notice that we count towards the size also the hard formulas.*

The soundness and completeness proofs of CC seen previously for the context of soft clauses adapt to hard and soft clauses without major modifications.

▶ **Theorem A.4** (Soundness). *Given $\mathcal{F} = \mathcal{H} \cup \mathcal{S}$ and a* CC *derivation $\pi = \langle(\mathcal{H}_1; \mathcal{S}_1), \dots, (\mathcal{H}_s; \mathcal{S}_s)\rangle$ with $\mathcal{H}_1 = \mathcal{H}$, $\mathcal{S}_1 = \mathcal{S}$ and $\{\bot, \overset{k}{\dots}, \bot\} \subseteq \mathcal{S}_s$, it holds that $\mathrm{cost}(\mathcal{F}) = \mathrm{cost}_{\mathcal{H}}(\mathcal{S}) \geq k$.*

Proof. We prove by induction on i that $\mathrm{cost}(\mathcal{F}) = \mathrm{cost}_{\mathcal{H}_i}(\mathcal{S}_i)$. For $i = 1$ is trivial. Suppose that $\mathrm{cost}(\mathcal{F}) = \mathrm{cost}_{\mathcal{H}_i}(\mathcal{S}_i)$. We show that $\mathrm{cost}_{\mathcal{H}_i}(\mathcal{S}_i) = \mathrm{cost}_{\mathcal{H}_{i+1}}(\mathcal{S}_{i+1})$.

If $(\mathcal{H}_{i+1}, \mathcal{S}_{i+1})$ is obtained from $(\mathcal{H}_i, \mathcal{S}_i)$ by an application of the COMP rule, there are some $y_1, y_2 \in \mathcal{S}_{i+1}$ such that $\mathcal{H}_{i+1} = \mathcal{H}_i \cup \{CNF(y_1 \leftrightarrow b_1 \wedge b_2, y_2 \leftrightarrow b_1 \vee b_2\}$ and $\mathcal{S}_{i+1} = (\mathcal{S}_i \setminus \{b_1, b_2\}) \cup \{y_1, y_2\}$. Showing that for every assignment $\alpha \models \mathcal{H}_{i+1}$ it holds $\mathrm{cost}_\alpha(\mathcal{S}_{i+1}) = \mathrm{cost}_\alpha(\mathcal{S}_i)$ implies that $\mathrm{cost}_{\mathcal{H}_i}(\mathcal{S}_i) \leq \mathrm{cost}_{\mathcal{H}_{i+1}}(\mathcal{S}_{i+1})$. Recall that, since $\alpha \models \mathcal{H}_{i+1}$ we have $\alpha(y_1) = \alpha(b_1 \wedge b_2)$ and $\alpha(y_2) = \alpha(b_1 \vee b_2)$. Therefore,

$$\begin{aligned}
\mathrm{cost}_\alpha(\mathcal{S}_{i+1}) &= \sum_{b \in \mathcal{S}_{i+1} \setminus \{y_1, y_2\}} \mathrm{cost}_\alpha(b) + \mathrm{cost}_\alpha(y_1) + \mathrm{cost}_\alpha(y_2) \\
&= \sum_{b \in \mathcal{S}_{i+1} \setminus \{y_1, y_2\}} \mathrm{cost}_\alpha(b) + \mathrm{cost}_\alpha(b_1 \wedge b_2) + \mathrm{cost}_\alpha(b_1 \vee b_2) \\
&= \sum_{b \in \mathcal{S}_{i+1} \setminus \{y_1, y_2\}} \mathrm{cost}_\alpha(b) + \mathrm{cost}_\alpha(\{b_1 \wedge b_2, b_1 \vee b_2\}) \\
&= \sum_{b \in \mathcal{S}_{i+1} \setminus \{y_1, y_2\}} \mathrm{cost}_\alpha(b) + \mathrm{cost}_\alpha(\{b_1, b_2\}) \\
&= \mathrm{cost}_\alpha(\mathcal{S}_i)
\end{aligned}$$

The equality follows from the fact that any assignment $\alpha \models \mathcal{H}_i$ is uniquely extended to a model of $\mathcal{H}_{i+1}$.

If $\mathcal{S}_i = \{\ell\} \cup \mathcal{S}'_i$ where $\mathcal{H}_i \cup \{\ell\}$ is unsatisfiable and $(\mathcal{H}_{i+1}, \mathcal{S}_{i+1}) = (\mathcal{H}_i, \{\bot\} \cup \mathcal{S}'_i)$ then for every model $\alpha \models \mathcal{H}_i$ we have

$$\begin{aligned}
\mathrm{cost}_\alpha(\mathcal{S}_{i+1}) &= \mathrm{cost}_\alpha(\{\bot\}) + \mathrm{cost}_\alpha(\mathcal{S}'_i) \\
&= \mathrm{cost}_\alpha(\{\ell\}) + \mathrm{cost}_\alpha(\mathcal{S}'_i) \\
&= \mathrm{cost}_\alpha(\mathcal{S}_i),
\end{aligned}$$

concluding that $\mathrm{cost}_{\mathcal{H}_{i+1}}(\mathcal{S}_{i+1}) = \mathrm{cost}_{\mathcal{H}_i}(\mathcal{S}_i)$. ◀

▶ **Theorem A.5** (Completeness). *For every multiset of formulas* $\mathcal{F} = \mathcal{H} \cup \mathcal{S}$ *with* $\text{cost}(\mathcal{F}) = \text{cost}_{\mathcal{H}}(\mathcal{S}) = k$ *there exists a* CC *derivation* $\pi = \langle(\mathcal{H}_1; \mathcal{S}_1), \ldots, (\mathcal{H}_s; \mathcal{S}_s)\rangle$ *with* $\mathcal{H}_1 = \mathcal{H}$, $\mathcal{S}_1 = \mathcal{S}$ *and* $\{\bot, \overset{k}{\ldots}, \bot\} \subseteq \mathcal{S}_s$.

Proof. Let $\mathcal{S} = \{b_1, \ldots, b_m\}$ and assume without loss of generality that the multi-set $\mathcal{S}$ is well ordered. We proceed by induction on $\text{cost}(\mathcal{F})$. For $k = 0$ it is immediate. Suppose $\text{cost}(\mathcal{F}) = k$. We construct a sequence $\pi = \langle(\mathcal{H}_1, \mathcal{S}_1), \ldots, (\mathcal{H}_{m+1}, \mathcal{S}_{m+1})\rangle$ as follows:

- $\mathcal{H}_1 = \mathcal{H}$ and $\mathcal{S}_1 = \mathcal{S}$.
- For every $2 \leq t \leq m$ we define

  $$\mathcal{H}_t := \mathcal{H}_{t-1} \cup \{CNF(y_1 \leftrightarrow x_1 \wedge x_t, y_2 \leftrightarrow x_1 \vee x_t)\}$$

  and

  $$\mathcal{S}_t := (\mathcal{S}_{t-1} \setminus \{x_1, x_t\}) \cup \{y_1, y_2\},$$

  where $\mathcal{S}_{t-1} = \{x_1, \ldots, x_m\}$ and y_1, y_2 are fresh variables. That is, we obtain each $(\mathcal{H}_t, \mathcal{S}_t)$ from $(\mathcal{H}_{t-1}, \mathcal{S}_{t-1})$ applying the COMP rule once to x_1 and x_t.

Since $\mathcal{S}$ is well-ordered, each $\mathcal{S}_t$ is the multi-set $\{y_1, x_2, \ldots, x_{t-1}, y_2, x_{t+1}, \ldots, x_m\}$ where $\mathcal{S}_{t-1} = \{x_1, \ldots, x_m\}$: hence, if $\mathcal{S}_m = \{\ell_1, \ldots, \ell_m\}$ then $\mathcal{H}_m \cup \{\ell\}$ is logically equivalent to $\mathcal{F}$.

Since $\text{cost}(\mathcal{F}) > 0$, $\mathcal{F}$ is unsatisfiable and therefore $\mathcal{H}_m \cup \{\ell\} \vdash \bot$. We apply then the CONTR rule to $(\mathcal{H}_m, \mathcal{S}_m)$, obtaining the pair

$$(\mathcal{H}_{m+1}, \mathcal{S}_{m+1}) = (\mathcal{H}_m, \{\bot\} \cup \mathcal{S}'_m)$$

where $\mathcal{S}'_m = \mathcal{S}_m \setminus \{\ell\}$.

The sequence $\langle(\mathcal{H}_i, \mathcal{S}_i) : i \in \{1, \ldots, m+1\}\rangle$ is a valid CC proof of $cost(\mathcal{F}) \geq 1$. Moreover, notice that $cost_{\mathcal{H}_m}(\mathcal{S}'_m) = k - 1$, so by induction hypothesis there is a CC proof $\langle(\mathcal{H}_{m+1}, \mathcal{S}'_{m+1}), \ldots, (\mathcal{H}_t, \{\bot, \overset{k-1}{\ldots}, \bot\} \cup \mathcal{S}_t)\rangle$ and where $\mathcal{H}_t \cup \mathcal{S}_t$ is satisfiable. Putting both proofs together, we obtain a valid CC proof of $\text{cost}(\mathcal{F}) = k$.

◀

A.3 Simulations

▶ **Theorem A.6.** CC$_P$ *polynomially simulates the* OLL_P *calculus using sorting networks to encode soft cardinality constraints.*

Proof. One application of the OLL rule (in eq. (3)) can be simulated by $\mathcal{O}(n \log n)$ applications of the COMP rule using the construction from [1], plus one application of CONTR rule to replace $A_1 + \cdots + A_r \geq r$ —which is $A_1 \wedge \cdots \wedge A_r$ in any sorting network— by $\bot$. We use the same proof system P to certify the correctness of the CONTR rule and the correctness of the OLL rule. In both cases, we have to certify in P the unsatisfiability of $A_1 \wedge \cdots \wedge A_r$. ◀

▶ **Theorem A.7.** CC$_P$ *linearly simulates the* $Fu\&Malik_P$ *calculus with symmetry breaking.*

Proof. One application of the FU&MALIK SYM. BREAK rule can be simulated by $r - 1$ applications of the COMP rule plus one application of CONTR to derive $\bot$ from $A_1 \wedge \cdots \wedge A_r$. Like in the OLL case, we assume we are using the same proof system in both calculi to certify the unsatisfiability of $A_1 \wedge \cdots \wedge A_r$. ◀

A.4 CSat

▶ **Theorem A.8** (Correctness). *The algorithm CSat on input* $\mathcal{F} = \mathcal{H} \cup \mathcal{S}$*, if it terminates, returns* $\mathrm{cost}_{\mathcal{H}}(\mathcal{S})$*.*

Proof. Let $\mathcal{H}_0$ and $\mathcal{S}_0$ be the input of the algorithm. The following quantities

$$rub = \min_{\alpha \in \mathcal{A}} \{\mathrm{cost}_\alpha(\mathcal{S})\} \quad \text{and}$$

$$\mathrm{cost}_{\mathcal{H}_0}(\mathcal{S}_0) = \mathrm{cost}_{\mathcal{H}}(\mathcal{S}) + lb$$

are invariants of the main loop.

For the first invariant, notice that rub is initialized to $\min_{\alpha \in \mathcal{A}}\{\mathrm{cost}_\alpha(\mathcal{S}_0)\}$. Every time a new assignment α is added to $\mathcal{A}$, we compute the minimum among the old value and $\mathrm{cost}_\alpha(\mathcal{S})$, hence maintaining the invariant. When we remove an unsatisfiable soft clause from $\mathcal{S}$, i.e. when we apply CONTR, we decrease rub, again maintaining the first invariant.

For the second invariant, notice that runs of CSat simulate the application of CONTR and COMP on $\mathcal{H}$ and $\mathcal{S}$, except that, in the application of CONTR, instead of replacing $\mathcal{S}$ by $(\mathcal{S} \setminus \{c\}) \cup \{\bot\}$, we remove the clause c and increase lb by 1. By induction, if we are in the iteration i and we have already obtained lb_i, if we apply the COMP rule to some $(\mathcal{H}_i, \mathcal{S}_i)$, we do not increase lb_i and therefore the soundness of the rule implies the equality

$$\begin{aligned}\mathrm{cost}_{\mathcal{H}_0}(\mathcal{S}_0) &= \mathrm{cost}_{\mathcal{H}_i}(\mathcal{S}_i) + lb_i \\ &= \mathrm{cost}_{\mathcal{H}_{i+1}}(\mathcal{S}_{i+1}) + lb_{i+1}.\end{aligned}$$

If we remove an unsatisfiable clause c due to the CONTR rule, we increase lb_i by 1, i.e., $lb_{i+1} = lb_i + 1$. Moreover, it is clear that

$$\begin{aligned}\mathrm{cost}_{\mathcal{H}_{i+1}}(\mathcal{S}_{i+1}) &= \mathrm{cost}_{\mathcal{H}_i}(\mathcal{S}_i \setminus \{c\}) \\ &= \mathrm{cost}_{\mathcal{H}_i}(\mathcal{S}_i) - 1.\end{aligned}$$

Therefore,

$$\begin{aligned}\mathrm{cost}_{\mathcal{H}_{i+1}}(\mathcal{S}_{i+1}) + lb_{i+1} &= \mathrm{cost}_{\mathcal{H}_i}(\mathcal{S}_i) - 1 + lb_i + 1 \\ &= \mathrm{cost}_{\mathcal{H}_i}(\mathcal{S}_{i+1}) + lb_i \\ &= \mathrm{cost}_{\mathcal{H}_0}(\mathcal{S}_0).\end{aligned}$$

When we leave the loop, we have $rub = 0$, therefore there exists an assignment α in $\mathcal{A}$ that satisfies all soft clauses in $\mathcal{S}$, $\mathrm{cost}_\alpha(\mathcal{S}) = 0$, hence $\mathrm{cost}_{\mathcal{H}}(\mathcal{S}) = 0$ and $\mathrm{cost}_{\mathcal{H}_0}(\mathcal{S}_0) = lb$. ◀

▶ **Theorem A.9.** *The algorithm CSat with the heuristic in Algorithm 2 always terminates in* $\mathcal{O}(|\mathcal{S}|^2)$ *iterations.*

Proof. The heuristic function in Algorithm 2, when called in CSat, always returns a pair (b_1, b_2) of soft clauses such that $\mathrm{count}_{\mathcal{A}}(b_1 \wedge b_2) > \mathrm{count}_{\mathcal{A}}(z)$, for any soft clause $z \in \mathcal{S}$.

Indeed, when the function heuristic is called

1. the condition $\exists c \in \mathcal{S}\, \forall \alpha \in \mathcal{A}\ \alpha(c) = 0$ is false, hence for every $b \in \mathcal{S}$ exists an assignment $\alpha \in \mathcal{A}$ s.t. $\alpha(b) = 1$, and
2. $rub > 0$, hence for every assignment $\alpha \in \mathcal{A}$ exists a $b \in \mathcal{S}$ with $\alpha(b) = 0$.

In other words, any row in the Boolean matrix M has a 1, and any column has a 0.

For any $x \in B_1$, i.e. maximizing $\mathrm{count}_{\mathcal{A}}(x)$, there exists an $\alpha \in \mathcal{A}$ such that $\alpha(x) = 1$. Hence $\mathrm{count}_{\mathcal{A}}(x)$ is strictly smaller than $|\mathcal{A}|$. For this α, there exists an y such that $\alpha(y) = 0$.

Clearly, $x \neq y$. We conclude that, for any $(x, y) \in B_2$, $\text{count}_{\mathcal{A}}(x \wedge y) > \text{count}_{\mathcal{A}}(x)$ and, since $\text{count}_{\mathcal{A}}(x)$ was maximal, $\text{count}_{\mathcal{A}}(x \wedge y) > \text{count}_{\mathcal{A}}(z)$, for any $z \in \mathcal{S}$. Since B_3 is a subset of B_2, we can conclude this property for any pair returned by heuristic.

Now, every time heuristic is called, it returns a pair of soft clauses whose conjunction was not in $\mathcal{S}$. Therefore, it cannot be called more than $|\mathcal{S}| - 1$ many times. At some point we get a conjunction of soft clauses that is unsatisfiable, or we get a satisfying assignment and we leave the loop. In the worst case, the CONTR rule will be applied $|\mathcal{S}|$ many times, and between every two applications of CONTR, the COMP rule will be applied $|\mathcal{S}| - 1$ many times. ◀