Reduced Cost Fixing for Maximum Satisfiability*

Fahiem Bacchus¹, Antti Hyttinen², Matti Järvisalo², Paul Saikko²

¹ Department of Computer Science, University of Toronto, Canada
² HIIT, Department of Computer Science, University of Helsinki, Finland

fbacchus@cs.toronto.edu, matti.jarvisalo@helsinki.fi, antti.hyttinen@helsinki.fi, psaikko@cs.helsinki.fi

Abstract

Maximum satisfiability (MaxSAT) offers a competitive approach to solving NP-hard real-world optimization problems. While state-of-the-art MaxSAT solvers rely heavily on Boolean satisfiability (SAT) solvers, a recent trend, brought on by MaxSAT solvers implementing the so-called implicit hitting set (IHS) approach, is to integrate techniques from the realm of integer programming (IP) into the solving process. This allows for making use of additional IP solving techniques to further speed up MaxSAT solving. In this line of work, we investigate the integration of the technique of reduced cost fixing from the IP realm into IHS solvers, and empirically show that reduced cost fixing considerable speeds up a state-of-the-art MaxSAT solver implementing the IHS approach.

1 Introduction

Beyond its importance as a classical NP-hard optimization problem, maximum satisfiability (MaxSAT) is today a thriving constraint optimization paradigm, with successful applications in a range of real-world domains. The underlying declarative language of MaxSAT consists of conjunctive normal form (CNF) propositional formulas extended with weights assigned to individual clauses. The aim is to find an optimal truth assignment, i.e., one that maximizes the weight of satisfied clauses. Algorithmically, however, MaxSAT is typically treated as the equivalent problem of minimizing the weight (or cost) of falsified clauses.

The currently most successful exact MaxSAT solvers make iteratively use of Boolean satisfiability (SAT) solvers, either to extract unsatisfiable cores (unsatisfiable sets of soft clauses) or to extract truth assignments with ever-improving cost (e.g., [Koshimura et al., 2012]). Pure SAT-based solvers that extract cores (e.g., [Martins et al., 2014; Ansótegui et al., 2015; Alviano et al., 2015; Morgado et al., 2014]) use these cores to modify the formula until eventually a formula is obtained that is satisfiable and whose satisfying truth assignments are optimal truth assignments for the original input formula. Implicit hitting set (IHS) solvers [Davies and Bacchus, 2011; Davies, 2013; Saikko et al., 2016a] also extract cores, but then use an integer programming (IP) solver to compute a minimum-cost hitting set (MCHS) of the accumulated set of cores. If removing such a hitting set (which is a set of soft clauses) from the input formula makes the formula satisfiable, then the satisfying truth assignment must be an optimal truth assignment for the original input formula.

The hybrid approach implemented in IHS solvers for MaxSAT opens up opportunities for making use of various proven IP solving techniques to further speed up MaxSAT solving. In this work we investigate utilizing the known IP technique of reduced cost fixing for this purpose.

In the context of MaxSAT, truth assignments found during solving provide an upper bound on the cost of an optimal truth assignment. Every MCHS of the cores on the other hand provides a lower bound. The linear programming (LP) relaxation of the MCHS problem thus also provides a lower bound. This LP can also be used to obtain a reduced cost for every soft clause $c_i$. The reduced cost of $c_i$ specifies a minimum increase in the cost of the LP that would arise from falsifying (satisfying) $c_i$ which was satisfied (falsified) in the LP solution. If this cost increase makes the LP MCHS lower bound greater than the best known upper bound, we can conclude that $c_i$ must be fixed to its status in the LP solution; that is, $c_i$ must be falsified (satisfied) by any optimal truth assignment. This reasoning can be extended to cover the case where the increase in cost makes the lower bound equal to the upper bound. In this case, we know that fixing the status of $c_i$ preserves at least one optimal solution. Fixing the status of various soft clauses makes solving MaxSAT instances easier, as our empirical results demonstrate.

To facilitate the integration of reduced cost fixing, we present IHS search in terms of improving upper and lower bounds. To study the impact of reduced cost fixing for MaxSAT, we present results from an extensive empirical evaluation on the effect of integrating reduced cost fixing into the MaxHS IHS solver, showing that reduced cost fixing considerable speeds up this state-of-the-art MaxSAT solver.

In terms of related work, different techniques of using lower and upper bounds for speeding up MaxSAT solvers

---

*This is an abridged version of “Reduced Cost Fixing in MaxSAT” that won the CP 2017 Distinguished Paper Award. This work was financially supported by Academy of Finland (grants 251170, 276412, 295673, 312662) and DoCS Doctoral Programme in Computer Science, University of Helsinki.
have been studied in varying contexts, including branch and bound for MaxSAT [Li et al., 2006; Lin and Su, 2007; Lin et al., 2008; Li et al., 2008], use of bounds for MaxSAT solvers in general [Heras et al., 2012], and hardening based on the costs of residual formulas in pure SAT-based core-guided MaxSAT solving [Ansótegui et al., 2013; Morgado et al., 2012]. To the best of our knowledge, linear programming relaxation based reduced cost fixing has not been previously proposed in the context of MaxSAT. However, besides being a standard technique in IP solving [Wolsey, 1998; Danzig et al., 1954; Crowder et al., 1983; Nemhauser and Wolsey, 1999], there have been a number of recent works on exploiting reduced cost fixing in constraint programming, IP/constraint logic programming, and IP/constraint-based solving, e.g., [Thorsteinsson and Ottoson, 2002; Focacci et al., 1999; Yunes et al., 2010].

2 Maximum Satisfiability and Hitting Sets

An instance of (weighted partial) MaxSAT \((F_h, F_s, wt)\) consists of a set of hard clauses \(F_h\), a set of soft clauses \(F_s\), and weight function \(wt: F_s \rightarrow \mathbb{R}^+\). A MaxSAT solution \(\tau\) is a truth assignment for the variables in \(F_h \cup F_s\) that satisfies all clauses in \(F_h\). The cost of a solution, \(cost(\tau)\), is the sum of the weights of soft clauses not satisfied by \(\tau\). A MaxSAT solution \(\tau'\) is optimal if it has minimum cost over all solutions, \(\tau' = \arg\min_{\tau} cost(\tau)\). A subset of soft clauses \(\kappa \subseteq F_s\) is an unsatisfiable core if the clauses in \(\kappa\) cannot be simultaneously satisfied by any solution.

A hitting set \(HS\) of a set of cores \(K\) is a set which contains at least one element of each \(k \in K\), i.e., \(HS \cap k \neq \emptyset\) for all \(k \in K\). A minimum-cost hitting set (MCHS) of \(K\) is a hitting set which minimizes \(\sum_{c_i \in HS} wt(c_i)\) over all hitting sets \(HS\) of \(K\). The optimal cost of a MaxSAT instance is equal to the cost of the minimum-cost hitting sets of the set of all cores of the MaxSAT instance.

Example 2.1. Consider the MaxSAT instance \((F_h, F_s, wt)\) with the hard clauses

\[
F_h = \{ x_1 \lor x_2, x_2 \lor x_3, x_1 \lor x_4, \neg x_4 \lor \neg x_5, \neg x_1 \lor \neg x_6, \neg x_2 \lor \neg x_7, \neg x_3 \lor \neg x_8 \},
\]

and soft clauses

\[
F_s = \{ x_5, x_6, x_7, x_8 \}
\]

with \(wt(c_1) = wt(c_2) = 2, wt(c_3) = 4, wt(c_4) = 1\).

The cores of this instance form the set \(K = \{\{c_1, c_2\}, \{c_2, c_3\}, \{c_3, c_4\}\}\). For example, \(\{c_1, c_2\}\) is a core since \(x_5 = 1\) \((c_1)\) implies \(x_4 = 0\) and \(x_1 = 1\), and \(x_6 = 1\) \((c_2)\) implies \(x_1 = 0\); thus both cannot be satisfied by the same solution. An optimal solution is \(\tau: x_1 = x_3 = x_5 = x_7 = 1, x_2 = x_4 = x_6 = x_8 = 0\) with \(cost(\tau) = 3\). One MCHS of \(K\) is \(\{c_1, c_3\}\) with cost 3.

3 The Implicit Hitting Set Approach

We focus on IHS MaxSAT solvers [Davies and Bacchus, 2011; 2013a; 2013b; Davies, 2013; Saikko et al., 2016a] which instantiate the implicit hitting set paradigm [Karp, 2010; Moreno-Centeno and Karp, 2013; Saikko et al., 2016b] as natural candidates for integrating reduced cost fixing. IHS solvers iteratively alternate between computing a set of unsatisfiable cores \(K\) of the MaxSAT instance and minimum-cost hitting sets over the monotonically growing \(K\). To obtain a MCHS \(HS\), an IP solver is employed on the MCHS IP formulation

\[
\begin{align*}
\text{minimize} & \sum_{c_i \in F_s} wt(c_i) \cdot b_i \\
\text{subject to} & \sum_{c_i \in k} b_i \geq 1 & & \forall k \in K, \\
& b_i \in \{0, 1\} & & \forall c_i \in F_s,
\end{align*}
\]

associating a new binary variable \(b_i\) with each soft clause \(c_i\). At each iteration the satisfiability of \(F_h \cup (F_s \setminus HS)\) is checked with a SAT solver. If satisfiable, the set \(HS\) implicitly hits all cores of the instance and, as first shown in [Davies and Bacchus, 2011], the optimality of \(HS\) implies the optimality of the MaxSAT solution, terminating the MaxSAT search.

To make use of reduced cost fixing, upper bounds on the cost of optimal MaxSAT solutions are needed. As originally described in [Davies and Bacchus, 2011], IHS does not directly provide upper bounds during search. However, upper bounds are obtained in the IHS solvers MaxHS and LMHS when non-optimal hitting sets are used as described in [Davies and Bacchus, 2013b]. In particular, for any hitting set \(HS\), if the SAT solver finds a satisfying assignment \(\tau\) for \(F_h \cup (F_s \setminus HS)\), then \(cost(\tau)\) is an upper bound \(cost(\tau)\) is equal to the optimal cost if \(HS\) is a MCHS.

Here we provide a bounds-based view on the iterations and termination of the IHS algorithm for MaxSAT. Illustrated in Figure 1, the bounds-based IHS algorithm for MaxSAT starts by initializing the SAT solver and IP solver with their respective inputs. The SAT solver is initialized with the CNF formula \(F_h \cup F_s\), while the IP solver is initialized with the soft clause weights \(wt\). The hitting set \(HS\) and set of cores \(K\) are both initially empty. The upper bound \(UB\) is initialized to \(\infty\) and \(LB\) to 0. We assume for simplicity that \(F_h\) is satisfiable. The first SAT solver invocation tests the entire set of clauses \(F_h \cup F_s\). If the formula is satisfiable, we obtain an upper

![Figure 1: A bounds-based view on IHS for MaxSAT](image-url)
bound of 0 and terminate as $LB = UB$. Otherwise we obtain an unsatisfiable core $\kappa$ and add it to $K$.

In order to obtain the upper bounds required for reduced cost fixing, we deviate from the original IHS algorithm. Instead of immediately computing a new optimal hitting set for $K$ we add the clauses of the new core to $HS$, forming a non-optimal hitting set of $K$, and check the satisfiability of $F_h \cup (F_s \setminus HS)$ again. (A non-optimal hitting set of $K \cup \{\kappa\}$ could be also obtained in other ways [Davies and Bacchus, 2013b]). This is repeated, adding cores $\kappa$ to $K$ and the clauses of the new core to $HS$, until $F_h \cup (F_s \setminus HS)$ becomes satisfiable (this is the “disjoint” strategy of [Saikko, 2015]). $F_3$ is satisfiable, so this condition will eventually be met when $HS$ grows sufficiently large. Once $F_h \cup (F_s \setminus HS)$ is satisfiable, a MaxSAT solution is obtained. Its cost gives an upper bound value and $UB$ is updated. At this point the IP solver is invoked to compute a minimum-cost hitting set $HS$ for $K$. The hitting set $HS$ is optimal, so its cost gives a lower bound for the cost of the optimal MaxSAT solution and $LB$ is updated.

The algorithm terminates when $LB = UB$ and returns the MaxSAT solution which yielded $UB$. By the same argument as presented in [Davies and Bacchus, 2011] the algorithm must eventually terminate. A detailed proof of correctness is provided in [Bacchus et al., 2017].

**Example 3.1.** Consider again the MaxSAT instance $(F_h, F_s, wt)$ from Example 2.1. Following Figure 1, a SAT solver is called on $F_h \cup F_s$. Assume that it returns UNSAT with the core $\kappa = \{c_2, c_3\}$. The SAT solver is called again on $F_h \cup (F_s \setminus \kappa)$, now returning the solution $\tau : x_1 = x_3 = x_5 = x_7 = 1, x_2 = x_4 = x_6 = x_8 = 0$. This solution has cost($\tau$) = 3, which gives an upper bound $UB = 3$. The IP solver then called to compute a minimum-cost hitting set over $K = \{\{c_2, c_3\}\}$. Clearly, $HS = \{c_2\}$. The SAT solver is called on $(F_h \cup F_s) \setminus \{c_2\}$, it returns UNSAT with core $\kappa = \{c_3, c_4\}$. The next SAT solver call over $(F_h \cup F_s) \setminus \{c_2, c_3, c_4\}$ returns SAT but cannot improve $UB$. The IP solver is called for a minimum-cost hitting set over $K = \{\{c_2, c_3, c_4\}\}$. This gives $HS = \{c_2, c_4\}$ and $LB = 3$. Now $UB = LB$ and the search terminates.

4 Reduced Cost Fixing for IHS

Reduced cost fixing is a standard technique in IP solving [Wolsey, 1998; Danzig et al., 1954; Crowder et al., 1983; Nemhauser and Wolsey, 1999]. It uses an upper bound and reduced costs obtained from an LP relaxation to fix variables in an IP. Given a minimization IP $P$ containing Boolean (0/1) variables, we can solve $P$ as an LP by allowing the Boolean variables to take on intermediate values between 0 and 1. The cost of the LP solution will be a lower bound on the optimal cost of $P$. The LP solver also provides a reduced cost for the non-basic

1 variables set at 0 or 1 in the LP solution. These reduced costs specify a minimum increase in the cost of the LP that would arise from changing a non-basic variable at 0 (1) to 1 (0). Suppose we know a feasible IP solution to $P$ with cost $z$. If changing a non-basic variable causes the LP solution to increase in cost beyond $z$, then we can fix that variable to the value it has in the LP solution. Since the LP solution is a lower bound, putting such variables at their opposite values would cause the cost of the IP to increase beyond the cost of an already known feasible solution. Here we explain how this technique can be used within IHS MaxSAT solvers.

**Theorem 4.1.** For a MaxSAT problem $F = (F_h, F_s, wt)$, suppose we have (a) $B = \{b_1, \ldots, b_n\}$ a set of Boolean variables where each $b_i = 0 (b_i = 1)$ represents the satisfaction (falsification) of a soft clause $c_i \in F_s$, (b) $IP_{HS}$ an IP over the $b_i$ representing the minimum-cost hitting set problem over the current set of cores, (c) $LP_{HS}$ the LP relaxation of $IP_{HS}$, (d) best $\tau$ a feasible solution to $F$, (e) an optimal solution to $LP_{HS}$ with cost $z_{opt}^{LP_{HS}}$, and (f) LP reduced costs $rc(b_i)$ at the optimal basis.

Then the following simplifications can be performed without changing opt $cost(F)$. (1) For every non-basic variable $b_i$ set to 0 in the optimal $LP_{HS}$ solution we can make the soft clause $c_i$ hard in $F$ if $z_{opt}^{LP_{HS}} + rc(b_i) > \text{cost}(best, \tau)$ or if $z_{opt}^{LP_{HS}} + rc(b_i) = \text{cost}(best, \tau)$ and $c_i$ is satisfied in best $\tau$. (2) For every non-basic variable $b_i$ set to 1 in the optimal $LP_{HS}$ we can make soft clause $c_i$ false in $F$ if $z_{opt}^{LP_{HS}} - rc(b_i) > \text{cost}(best, \tau)$ or if $z_{opt}^{LP_{HS}} - rc(b_i) = \text{cost}(best, \tau)$ and $c_i$ is falsified in best $\tau$.

For more intuition on how reduced cost fixing allows for hardening and falsifying soft clauses during IHS search for an optimal MaxSAT solution, we consider the following example.

**Example 4.2.** Consider again the MaxSAT instance $(F_h, F_s, wt)$ from Example 2.1 and the first iteration of the execution of IHS as described in Example 3.1. After obtaining from the SAT solver the first solution best $\tau$ with $\text{cost}(best, \tau) = 3$, the IP solver is called for a hitting set over $K = \{\{c_2, c_3\}\}$. The LP-relaxation of the hitting set problem $LP_{HS}$ can be formulated as

$$\min 2b_1 + 2b_2 + 4b_3 + 1b_4$$

subject to $b_4 - b_1 - s = 1,$

$$0 \leq s, 0 \leq b_1, b_2, b_3, b_4 \leq 1,$$

where $s$ is a surplus variable. This LP can be solved using different algorithms, clearly the optimal LP solution is $b_2 = 1, b_1 = b_3 = b_4 = s = 0$, which has cost $z_{opt}^{LP_{HS}} = 2$. This induces a lower bound $LB = 2$. In this case, the optimal LP solution happens to be the optimal solution to the IP HS problem, i.e. $HS = \{c_2\}$.

A feasible solution to an LP can be represented using a basis, where non-basic variables are at their upper or lower bounds. Consider the basis consisting of $b_3$. The basic variables can be written as a function of the non-basic ones, in this case as $b_3 = 1 - b_2 + s$. This allows the objective to be rewritten solely as a function of the non-basic variables as

$$2b_1 + 2b_2 + 4(1 - b_2 + s) + b_4 = 4 + 2b_1 - 2b_2 + b_4 + 4s$$

The coefficients in front of the non-basic variables are the reduced costs and (obviously) specify the change of the objective to modifications of the non-basic variables. Now, since
$b_1$ is set to 0 in the optimal LP solution and $z_{opt}^{LP_{dis}} + r(c(b_1)) = 2 + 2 > 3 = \text{cost(best, } \tau) = v_1$, we can make $c_1$ hard. Intuitively, solutions under which $c_1$ is satisfied ($b_1 = 1$) would have cost at least 4 and hence such solutions cannot be optimal for the MaxSAT instance. Since $b_2$ is set to 1 in the LP solution and $z_{opt}^{LP_{dis}} - r(c(b_2)) = 2 - (-2) > 3 = \text{cost(best, } \tau) = v_2$, we can make $c_2$ false. Intuitively, solutions under which $c_2$ is satisfied ($b_2 = 0$) would have cost at least 4 and hence such solutions cannot be optimal, either.

5 Empirical Results

We implemented reduced cost fixing in MaxHS v2.9.8 which entered the 2016 MaxSAT Evaluation. This version of MaxHS includes a number of other features shown to improve the solver, described in [Davies and Bacchus, 2013b; Saikko et al., 2016a]. We compare the performance of MaxHS with and without reduced cost fixing, with all other features unchanged. We utilized IBM CPLEX v12.7 as the ILP solver, and ran our experiments on computing nodes with Xeon 2.8-GHz cores and 256-GB RAM. We limited MaxHS to 1800 seconds and 3.5 GB on each instance. We also report on longer 5-hour (18,000 s), 5-GB runs on Xeon 2.0-GHz cores and 256-GB RAM.

We experimented with all non-random instances that have been collected and made available by the MaxSAT Evaluation during the years 2008–2016. These include extra submitted benchmarks never used in the evaluation. After pruning duplicate instances this yielded 6290 MaxSAT instances (4361 unweighted, 1929 weighted). For the 5-hour runs, however, we omitted 507 unweighted instances with no hard clauses (MS instances) most of which encode MaxCut on random graphs. Core-based solvers, including IHS solvers, perform poorly on such instances, and we did not expect any of these instances to complete in 5 hours with or without reduced cost fixing. This left 5783 instances for these longer experiments (4361 unweighted, 1422 weighted).

Figure 2 left shows a histogram of the instances grouped by the number of soft clauses that become fixed during solving. In 5024 of the 6290 instances no reduced cost fixing ever occurs (3953 unweighted, 1071 weighted), but in the remaining 1266 instances fixing can be quite common—in 791 of these instances 100 or more fixings occurred. In extreme cases over a million soft clauses were fixed by the technique.

There were 26 instances where fixing took more than 100 seconds. However, 25 of these were not solvable without additional fixings (22 were MaxCut on random graphs). On one solved instance fixing required 214 seconds out of a total solve time of 835 seconds (this instance was solved in 416 seconds without fixing). Of the remaining 6264 instances, on 1782 instances fixing took zero seconds (LP solving was never invoked since the gap between UB and LB was never small enough), on 3746 instances fixing took less than 1 second, on 298 instances fixing took between 1 and 10 seconds, and on 438 instances fixing took more than between 10 and 100 seconds. Figure 2 middle shows, however, that on these 438 instances fixing is well worth the time it takes. The scatter plot shows that fixing provides a significant speedup for most of these instances, especially on the harder instances.

Figure 2 right shows the performance improvement obtained from reduced cost fixing under a 5-hour per-instance time limit. Here we computed the speedup ratio for each instance, i.e., the CPU time taken without reduced cost fixing divided by the CPU time taken when reduced cost fixing is used. As this ratio will be between 0 and 1 for instances that are slowed down by fixing, we took $\log_2$ of this ratio which produces a symmetry between speedups and slowdowns. The plot is in the form of a histogram showing how many instances experience various ranges of the log speedup, and it clearly shows the value of our approach to employing reduced cost fixing in IHS based MaxSAT solvers.

References


