Cores in Core Based MaxSat Algorithms: An Analysis

Fahiem Bacchus and Nina Narodytska

Department of Computer Science, University of Toronto,
Toronto, Ontario, Canada, M5S 3H5
{fbacchus,ninan}@cs.toronto.edu

Abstract. A number of MAXSAT algorithms are based on the idea of generating unsatisfiable cores. A common approach is to use these cores to construct cardinality (or pseudo-boolean) constraints that are then added to the formula. Each iteration extracts a core of the modified formula that now contains cardinality constraints. Hence, the cores generated are not just cores of the original formula, they are cores of more complicated formulas. The effectiveness of core based algorithms for MAXSAT is strongly affected by the structure of the cores of the original formula. Hence it is natural to ask the question: how are the cores found by these algorithms related to the cores of the original formula? In this paper we provide a formal characterization of this relationship. Our characterization allows us to identify a possible inefficiency in these algorithms. Hence, finding ways to address it may lead to performance improvements in these state-of-the-art MAXSAT algorithms.

1 Introduction

MAXSAT is an optimization version of SAT in which the problem is to find a truth assignment that satisfies a maximum weight of clauses. In its most general form, a MAXSAT problem is expressed as a CNF formula partitioned into *hard* and *soft* clauses. Associated with each soft clause c_i is a numeric weight, w_i, and with each set of soft clauses S a cost, $cost(S)$, equal to the sum of the weights of the clauses in S. Various restricted versions of MAXSAT have also been studied [12]. Like SAT many practical problems can be encoded as MAXSAT formulas making the development of efficient MAXSAT solvers an important research problem.

There are a variety of algorithmic approaches to solving MAXSAT including solvers based on branch and bound, e.g., [10,13], solvers based on conversion to integer linear programs [5], solvers based on hybrid SAT and MIPs approaches [8], and core based solvers that use cardinality constraints, e.g., [15,4].

Core based solvers solve MAXSAT by solving a sequence of SAT problems using the cores returned by these SAT solving episodes to construct the next SAT problem. The performance of such solvers seems to depend on the structure of the cores of the original MAXSAT formula. For example, these solvers are quite successful when there are a large number of hard clauses which tends to reduce the size of the cores that must be dealt with. Similarly, these solvers do not work well on random problems and it is known that some types of random problems contain large cores [7]. However, achieving a clearer understanding of this relationship remains an open research problem.

C. Sinz and U. Egly (Eds.): SAT 2014, LNCS 8561, pp. 7–15, 2014.

In this paper we point out that core based solvers using cardinality constraints generate cores of a more complicated formula than the original MAXSAT formula. The cores they generate are cores of the MAXSAT formula augmented by cardinality constraints. We show that there is a precise relationship between the cores they generate and cores of the original MAXSAT formula. Our results could potentially help in obtaining a deeper understanding of how the structure of the cores of the MAXSAT instance affects the performance of this class of MAXSAT algorithms. More concretely, however, our results allow us to identify a possible source of inefficiency in such solvers. Developing techniques for removing this inefficiency thus becomes one way of potentially improving these solvers.

2 Background

A MAXSAT instance $\mathcal{F}$ is expressed as a CNF formula that is partitioned into two subsets of clauses $hard(\mathcal{F})$ and $soft(\mathcal{F})$. Note that in this paper we do not consider $\mathcal{F}$ to be a multi-set of clauses: multiple copies of a hard clause can be discarded, and multiple copies of a soft clause replaced with one copy with weight equal to the sum of the weights of the copies.

Definition 1 (Cost). *Each clause c_i in $soft(\mathcal{F})$ has an associated weight w_i. For any set of soft clauses $A \subseteq soft(\mathcal{F})$ we say that $cost(A) = \sum_{c_i \in A} w_i$, i.e., the cost of A is the sum of the weights of its soft clauses.*

Definition 2 (Solutions of $\mathcal{F}$). *A solution of $\mathcal{F}$ is a truth assignment π to the variables of $\mathcal{F}$ such that $\pi \models hard(\mathcal{F})$. The cost of a solution $cost(\pi)$ is the sum of the weights of the soft clauses it falsifies: $cost(\pi) = \sum_{\pi \not\models c_i} w_i$. The MAXSAT problem is to find a solution of $\mathcal{F}$ of minimum (optimal) cost.*

In this paper we assume that $hard(\mathcal{F})$ is satisfiable, i.e., solutions of $\mathcal{F}$ exist.

Definition 3 (Cores of $\mathcal{F}$). *A **core** κ of $\mathcal{F}$ is a subset of $soft(\mathcal{F})$ such that $\kappa \wedge hard(\mathcal{F})$ is unsatisfiable. Let $Cores(\mathcal{F})$ be the set of all cores of $\mathcal{F}$.*

We observe that for any core κ of $\mathcal{F}$ and solution π of $\mathcal{F}$, π must falsify at least one clause of κ. Furthermore, if κ is a core of $\mathcal{F}$ then any set of soft clauses A, that is a superset of κ, $A \supseteq \kappa$, is also a core.

3 The Fu and Malik Algorithm

To illustrate the type of MAXSAT algorithms under consideration we first describe one of the original core based MAXSAT algorithms due to Fu and Malik [9].

Fu & Malik works on restricted MAXSAT problems in which every soft clause has unit weight. These are called partial MAXSAT problems in the MAXSAT literature.

The algorithm executes a series of iterations, with the i-th iteration operating on the CNF formula $\mathcal{F}^i$, and the first iteration operating on the input MAXSAT formula, i.e., $\mathcal{F}^0 = \mathcal{F}$. Each iteration performs the following steps:

1. A SAT solver is called on $\mathcal{F}^i$. Note that the SAT solver ignores clause weights, regarding both $hard(\mathcal{F}^i)$ and $soft(\mathcal{F}^i)$ as ordinary clauses.

2. If $\mathcal{F}^i$ is satisfiable, then the satisfying truth assignment, restricted to the variables of $\mathcal{F}$ is an optimal MAXSAT solution.
3. Else $\mathcal{F}^i$ is unsatisfiable and we obtain a core κ from the SAT solver. Now the algorithm constructs the next formula $\mathcal{F}^{i+1}$ in two steps:
   (a) For every soft clause $c \in \mathcal{F}^i$ such that $c \in \kappa$ we add to c a literal b which is the positive literal of a brand new blocking variable (b-variable). Thus in $\mathcal{F}^{i+1}$ the clause c becomes the new clause $(c \vee b)$. This new clause $(c \vee b)$ is a soft clause of $\mathcal{F}^{i+1}$.
   (b) We add to $\mathcal{F}^i$ a new set of hard clauses encoding the cardinality constraint that the sum of the above newly added b-variables is equal to one. These new clauses are hard clauses of $\mathcal{F}^{i+1}$.

The added cardinality constraint allows one and only one soft clause of the discovered core to be relaxed by setting its b-variable to *true*. Each iteration installs an additional cardinality constraint which permits one more clause to be relaxed. Eventually $\mathcal{F}^i$, for some i, permits the relaxation of a sufficient number of clauses to achieve satisfaction.

One important point to notice is that if the SAT solver finds $\mathcal{F}^i$ to be unsatisfiable, then the core it returns is not a core of the original MAXSAT formula $\mathcal{F}$, it is a core of the relaxed formula $\mathcal{F}^i$. Since we want to increase our understanding of how algorithms like Fu & Malik are affected by the core structure of the MAXSAT formula, it becomes important to understand how the cores of $\mathcal{F}^i$, generated by the algorithm, are related to the cores of the original MAXSAT formula $\mathcal{F}$.

4 Cardinality Constraints

Now we present a general formulation of the problem we are addressing. This formulation is applicable not only to the Fu & Malik algorithm, but also to other core guided algorithms exploiting cardinality constraints like WPM1 [2] and WPM2 [3]. Our results are also applicable to the lower bounding phase of core guided algorithms that exploit binary search [15]. However, our results do not directly apply to iterative MAXSAT solvers, e.g., [6,11].

In general core guided algorithms impose linear inequalities or equalities over the blocking variables. These linear constraints are usually encoded into CNF and added to the formula. In some cases these constraints are handled directly without conversion to CNF, e.g., [1,14]. But even in these cases the constraints serve to restrict the satisfying models of the formula, so they are in effect "added" to the formula.

For convenience, we will call all such constraints *cardinality constraints*, although some of them are actually pseudo-boolean contraints. One important restriction of the analysis provided in this paper is that the cardinality contraints can only mention the b-variables and perhaps some other auxiliary variables. In particular, the cardinality constraints cannot mention any of the variables of $\mathcal{F}$. To the best of our knowledge, this restriction is satisfied by all existing core guided algorithms.

The cardinality constraints allow various sets of soft clauses to be "turned off" or blocked by allowing various combinations of the b-variables to be set to true. Since the b-variables appear only positively in the soft clauses of $\mathcal{F}^i$ each true b-variable satisfies

some soft clause making it impossible for that clause to contribute to unsatisfiability (the cardinality constraints do not affect $hard(\mathcal{F})$).

As seen in the previous section, every time a cardinality constraint is added the soft clauses of the current formula $\mathcal{F}^i$ are modified. Current algorithms use two types of modifications to the clauses in $soft(\mathcal{F}^i)$:

Adding a b-variable: This involves replacing $c \in soft(\mathcal{F}^i)$ by $c \vee b$ where b is the positive literal of a new b-variable. Since $c \in \mathcal{F}^i$ it might already contain some b-variables added in previous iterations.[1]

Cloning: This involves adding a duplicate c' of a clause $c \in soft(\mathcal{F}^i)$ where c' contains a new b-variable: $c' = (c \vee b)$.[2] The clone c' is given a weight w and w is subtracted from c's weight. Since $c \in F^i$, it might be that c is itself a clone added in a previous iteration. Thus an original soft clause of $\mathcal{F}$ might be split into multiple clones, each with its own sequence of b-variables. The total sum of the weights of all these clones is always equal to the weight of the original soft clause.

Let $\mathbf{card}^i$ be the set of cardinality constraints that have been added up to iteration i of a MAXSAT algorithm, $soft(\mathcal{F}^i)$ be the corresponding modified set of soft clauses, and $\mathcal{B}^i$ be the set of all b-variables in $\mathcal{F}^i$.

Definition 4 (Solutions of $\mathbf{card}^i$). *A truth assignment β to all of the variables of $\mathcal{B}^i$ that satisfies the cardinality constraints in $\mathbf{card}^i$ is called a **solution** of $\mathbf{card}^i$. The set of all solutions of $\mathbf{card}^i$ is denoted by $\mathbf{soln}(\mathbf{card}^i)$.*

4.1 Residues and Reductions

A solution of $\mathbf{card}^i$, β, relaxes various clauses of $soft(\mathcal{F}^i)$. Each clauses $c^i \in soft(F^i)$ is an original soft clause $c \in soft(\mathcal{F})$ disjoined with some b-variables. If β makes any of these b-variables true then c^i is in effect removed from the formula: β relaxes c^i. If c^i is not relaxed by β then it is reduced to the original soft clause c by β: β values every b-variable so if none of the b-variables in c^i are made true, then they must all be made false, in effect removing them from c^i. These two stages of reduction by β—the removal or relaxation of soft clauses and the reduction of the remaining soft clauses to original soft clauses—are important in characterizing the relationship between the cores of $\mathcal{F}^i$ and those of $\mathcal{F}$. These two stages or reduction are formalized in our definitions of *Residues* and *Reductions*.

Definition 5 (Residues). *Let β be a solution of $\mathbf{card}^i$ ($\beta \in \mathbf{soln}(\mathbf{card}^i)$) and let A^i be a subset of $soft(\mathcal{F}^i)$. The **residue** of A^i induced by β, denoted by $A^i \Downarrow \beta$, is the subset of A^i formed by removing all clauses satisfied by β:*

$$A^i \Downarrow \beta = A^i - \{c^i | c^i \in A^i \text{ and } \beta \models c^i\}.$$

Note that in a residue, $A^i \Downarrow \beta$, the clauses of A^i not satisfied by β are unchanged. Thus $A^i \Downarrow \beta$ is a subset of A^i which in turn is a subset of $soft(\mathcal{F}^i)$. Thus a residue is subset of F^i.

[1] Some algorithms like WPM2 add at most one b-variable to a soft clause, others like Fu & Malik can add multiple b-variables to a clause.

[2] This type of modification is used in the WPM1 algorithm to deal with weighted MAXSAT.

The next stage of reduction is achieved with the standard notion of the **reduction** of a set of clauses by a truth assignment.

Definition 6 (Reduction). *Let $\beta \in \mathbf{soln}(\mathbf{card}^i)$ and $A^i \subseteq soft(\mathcal{F}^i)$. The **reduction** of A^i induced by β, denoted $A^i|_\beta$ is the new set of clauses formed by (a) removing all clauses satisfied by β from A^i, (b) removing all literals falsified by β from the remaining clauses, and (c) removing all duplicate clauses and setting the weight of the remaining clauses to their original weights in $\mathcal{F}$.*

In the three steps to compute a reduction we see that step (a) is the same as forming the residue—removing all satisfied clauses. Step (b) reduces the soft clauses to original soft clauses of $\mathcal{F}$—as noted above, all remaining b-variables in the clauses after step (a) must be falsified by β and thus will be removed by step (b). However, step (b) does not quite produce soft clauses of $\mathcal{F}$ as the weights of these clauses might differ from the weights they had in $\mathcal{F}$ (due to cloning). This is fixed by step (c) which removes all duplicate clauses (due to cloning) and resets the weights back to the original weights. Thus it can be observed that the reduction of A^i is a subset of the original MAXSAT formula $\mathcal{F}$.

We make a few observations about residues and reductions. Let A and B be any subsets of $soft(\mathcal{F}^i)$ and let β and β' be any two solutions of $\mathbf{card}^i$.

1. $A \subseteq B$ implies $(A \Downarrow \beta) \subseteq (B \Downarrow \beta)$ and $(A|_\beta) \subseteq (B|_\beta)$.
2. $A \Downarrow \beta \subseteq soft(\mathcal{F}^i)$ while $A|_\beta \subseteq soft(\mathcal{F})$.
3. $A \Downarrow \beta = A \Downarrow \beta'$ implies $A|_\beta = A|_{\beta'}$.
4. $(A \Downarrow \beta)|_\beta = A|_\beta$, although sometimes we will use the notation $(A \Downarrow \beta)|_\beta$ as this more clearly indicates that reduction has two stages.
5. When the clauses of $soft(\mathcal{F}^i)$ have more than one b-variable it can be the case that $soft(F^i) \Downarrow \beta = soft(F^i) \Downarrow \beta'$ even when $\beta \neq \beta'$.

Example 1. Consider formula $\mathcal{F}^i$ with soft clauses c_1, c_2 and c_3 (as well as other hard clauses). Say a run of Fu & Malik discovers the sequence of cores (specified as clause indicies) $\kappa_1 = \{1, 2\}$, $\kappa_2 = \{2, 3\}$, and $\kappa_3 = \{1, 2, 3\}$. Using b-variables with a superscript to indicate the core number and a subscript to indicate the clause number, $soft(\mathcal{F}^3) \supset \{(c_1, b_1^1, b_1^3), (c_2, b_2^1, b_2^2, b_2^3), (c_3, b_3^2, b_3^3)\}$, and $\mathbf{card}^3 = \{CNF(b_1^1 + b_2^1 = 1), CNF(b_2^2 + b_3^2 = 1), CNF(b_1^3 + b_2^3 + b_3^3 = 1)\}$.

There are 12 different solutions to $\mathbf{card}^3$. However if we compute the residue of $A^3 = \{(c_1, b_1^1, b_1^3), (c_2, b_2^1, b_2^2, b_2^3), (c_3, b_3^2, b_3^3)\}$ with respect to these 12 solutions we obtain only 5 different residues: $\{(c_3, b_3^2, b_3^3)\}$, $\{(c_2, b_2^1, b_2^2, b_2^3)\}$, $\{(c_1, b_1^1, b_1^3), (c_3, b_3^2, b_3^3)\}$, $\{(c_1, b_1^1, b_1^3)\}$, and $\{\}$.

4.2 The Relationship between $Cores(\mathcal{F}^i)$ and $Cores(\mathcal{F})$

We can now present the paper's main result: a formalization of the relationship between the cores of $\mathcal{F}^i$ and the cores of the original MAXSAT formula $\mathcal{F}$. The next theorem shows that each core of $\mathcal{F}^i$ corresponds to a union of many cores of $\mathcal{F}$, and that every solution of $\mathbf{card}^i$ adds a core of $\mathcal{F}$ to this union.

Theorem 1. $\kappa^i \in \mathit{Cores}(F^i)$ *if and only if*

$$\kappa^i = \bigcup_{\beta \in \mathbf{soln}(\mathbf{card}^i)} \kappa^\beta \textbf{ \textit{where} } \kappa^\beta \subseteq (\mathit{soft}(\mathcal{F}^i) \Downarrow \beta) \textbf{ \textit{and} } \kappa^\beta|_\beta \in \mathit{Cores}(\mathcal{F})$$

Each κ^β in this union is a set of soft clauses of $\mathcal{F}^i$ that remain after removing all clauses satisfied by β (i.e., $\kappa^\beta \subseteq \mathit{soft}(\mathcal{F}^i) \Downarrow \beta$), such that its reduction by β ($\kappa^\beta|_\beta$) is a core of $\mathcal{F}$.

Proof. Note that $\mathit{hard}(\mathcal{F}^i) = \mathit{hard}(\mathcal{F}) \wedge \mathbf{card}^i$. Thus a core of $\mathcal{F}^i$ is a subset of $\mathit{soft}(\mathcal{F}^i)$ that together with $\mathit{hard}(\mathcal{F}) \wedge \mathbf{card}^i$ is unsatisfiable.

First we show that if κ^i is a core of $\mathcal{F}^i$ then it is a union of sets κ^β satisfying the stated conditions. For any $\beta \in \mathbf{soln}(\mathbf{card}^i)$ let $\kappa^\beta = (\kappa^i \Downarrow \beta)$. Then we observe that (a) since $\kappa^i \subseteq \mathit{soft}(\mathcal{F}^i)$ then $\kappa^\beta = \kappa^i \Downarrow \beta \subseteq (\mathit{soft}(\mathcal{F}^i) \Downarrow \beta)$ (by observation 1 above), and (b) $\kappa^\beta|_\beta \in \mathit{Cores}(\mathcal{F})$. To see that (b) holds we observe that if $\kappa^\beta|_\beta$ is not a core of $\mathcal{F}$ then there exists a truth assignment π to the variables of $\mathcal{F}$ such that $\pi \models \mathit{hard}(\mathcal{F}) \wedge \kappa^\beta|_\beta$. Since κ^β is a residue induced by β, no clause of κ^β is satisfied by β (the residue operation removes all clauses satisfied by β). Therefore, we have that $\pi \models \mathit{hard}(\mathcal{F}) \wedge \kappa^\beta$ even though π does not assign a value to any b-variable. Then $\langle \beta, \pi \rangle \models \mathit{hard}(\mathcal{F}) \wedge \mathbf{card}^i \wedge \kappa^i$ since β satisfies all clauses in $\kappa^i - \kappa^\beta$ (these were the clauses removed from κ^i when taking its residue with respect to β because they were satisfied by β) and all clauses in $\mathbf{card}^i$, while π satisfies κ^β and $\mathit{hard}(\mathcal{F})$. That is, if (b) does not hold we obtain a contradiction of the premise that κ^i is a core of $\mathcal{F}^i$. Since this argument holds for every $\beta \in \mathbf{soln}(\mathbf{card}^i)$ we see that $\kappa^i = \cup\, \kappa^\beta$.

Second, we show that $\cup\, \kappa^\beta$ is a core of $\mathcal{F}^i$. Say that it is not. Then there exists a truth assignment $\langle \gamma, \pi \rangle$ that satisfies $\mathit{hard}(\mathcal{F}) \wedge \mathbf{card}^i \wedge (\cup\, \kappa^\beta)$, where γ assigns the variables in $\mathcal{B}^i$ and satisfies $\mathbf{card}^i$ while π assigns all of the other variables. Since $\gamma \in \mathbf{soln}(\mathbf{card}^i)$ we have that $\kappa^\gamma \subseteq (\cup\, \kappa^\beta)$. Furthermore, since $\kappa^\gamma \subseteq (\mathit{soft}(\mathcal{F}^i) \Downarrow \gamma)$ no clause in κ^γ is satisfied by γ; therefore, all literals of $\mathcal{B}^i$ in κ^γ must be falsified by γ. Since $\langle \gamma, \pi \rangle \models \kappa^\gamma$ and we must also have that $\langle \gamma, \pi \rangle \models \kappa^\gamma|_\gamma$, and since $\kappa^\gamma|_\gamma$ has no variables of $\mathcal{B}^i$, we must have that $\pi \models \kappa^\gamma|_\gamma$. This, however, is a contradiction as $\pi \models \mathit{hard}(\mathcal{F})$ and $\kappa^\gamma|_\gamma$ is a core of $\mathcal{F}$.

5 Residue Subsumption in $\mathbf{card}^i$

Theorem 1 allows us to identify a potential inefficiency of MAXSAT algorithms that compute cores after adding cardinality constraints.

Definition 7 (Residue Subsumption). *Let β and β' be two solutions of $\mathbf{card}^i$. We say that β **residue subsumes** β' if (1) $\beta \neq \beta'$ and (2) $\mathit{soft}(\mathcal{F}^i) \Downarrow \beta \subseteq \mathit{soft}(\mathcal{F}^i) \Downarrow \beta'$.*

Residue subsumption means that β relaxes (satisfies) all or more of the soft clauses of $\mathcal{F}^i$ that are relaxed by β'.

Example 2. Continuing with Example 1, we observed that there are only 5 different residues of A^3 generated by the 12 different solutions to $\mathbf{card}^3$: $\{(c_3, b_3^2, b_3^3)\}$, $\{(c_2, b_2^1, b_2^2, b_2^3)\}$, $\{(c_1, b_1^1, b_1^3), (c_3, b_3^2, b_3^3)\}$, $\{(c_1, b_1^1, b_1^3)\}$, and $\{\}$. The empty residue $\{\}$ is generated by three solutions of $\mathbf{card}^3$ one of which is β which sets b_1^1, b_2^2 and b_3^3 to true. Hence, β residue subsumes *all* other solutions to $\mathbf{card}^i$.

Our next result shows that when computing a core of $\mathcal{F}^i$ it is possible to ignore residue subsumed solutions of $\mathbf{card}^i$.

Proposition 1. *Let $RS \subset \mathbf{soln}(\mathbf{card}^i)$ be a set of solutions such that for all $\rho' \in RS$ there exists a $\rho \in (\mathbf{soln}(\mathbf{card}^i) - RS)$ such that ρ residue subsumes ρ'. Then any*

$$\kappa^i = \bigcup_{\beta \in (\mathbf{soln}(\mathbf{card}^i) - RS)} \kappa^\beta \textbf{ \textit{where} } \kappa^\beta \subseteq (soft(\mathcal{F}^i) \Downarrow \beta) \textbf{ \textit{and} } \kappa^\beta|_\beta \in Cores(\mathcal{F})$$

is a core of F^i.

Proof. Let

$$\kappa^+ = \bigcup_{\beta \in \mathbf{soln}(\mathbf{card}^i)} \kappa^\beta \textbf{ where } \kappa^\beta \subseteq (soft(\mathcal{F}^i) \Downarrow \beta) \textbf{ and } \kappa^\beta|_\beta \in Cores(\mathcal{F}).$$

By Theorem 1 κ^+ is a core of $\mathcal{F}^i$. Let $\rho' \in RS$ be residue subsumed by $\rho \in (\mathbf{soln}(\mathbf{card}^i) - RS)$. κ^+ is a union of sets including sets $\kappa^{\rho'}$ and κ^ρ both of which satisfy the above conditions.

By substituting $\kappa^{\rho'}$ by κ^ρ for each $\rho' \in RS$ we see that κ^+ becomes equal to κ^i and thus κ^i must also be a core of $\mathcal{F}^i$ if this substitution is valid. To show that the substitution is valid we must show that κ^ρ satisfies the two conditions required for $\kappa^{\rho'}$. First, $\kappa^\rho \subseteq (soft(F^i) \Downarrow \rho) \subseteq (soft(\mathcal{F}^i) \Downarrow \rho')$. Second, since κ^ρ is a subset of both $soft(\mathcal{F}^i) \Downarrow \rho$, and $soft(\mathcal{F}^i) \Downarrow \rho'$, neither ρ not ρ' satisfy any clauses of κ^ρ. Furthermore, both ρ and ρ' assign all b-variables so all b-variables left in κ^ρ must be falsified by both ρ and ρ'. This means that $\kappa^\rho|_{\rho'} = \kappa^\rho|_\rho$ and $\kappa^\rho|_\rho$ is already known to be a core of $\mathcal{F}^i$.

Theorem 1 shows that when the SAT solver computes a core κ^i of $\mathcal{F}^i$ it must refute all solutions of $\mathbf{card}^i$. The SAT solver might not need to refute two residue subsuming solutions ρ and ρ' separately—it might be able to find a single conflict that eliminates both candidate solutions. Nevertheless, it is possible that the solver ends up constructing two separate and different refutations of ρ and ρ', eventually unioning them into a refutation of $\mathcal{F}^i$. This can make the core extracted from the refutation larger, and also requires more time for the SAT solver. From Prop. 1 it can be seen that the solver need only refute ρ, finding the required κ^ρ. Adding κ^ρ to the core suffices to refute both ρ and ρ'.

For the Fu & Malik algorithm it has previously been noted that many symmetries exist over the introduced b-variables [1], and symmetry breaking constraints can be introduced over the b-variables to remove some of these symmetries. These symmetry breaking constraints serve to reduce the set of solutions $\mathbf{soln}(\mathbf{card}^i)$. Residue subsumption offers a more general way achieving this result. In particular, Lemma 15 of [4] introduces an ordering (weight) over all solutions in $\mathbf{soln}(\mathbf{card}^i)$ and shows that the introduced symmetry breaking constraints block solutions that are residue subsumed by other higher weight solutions. In fact, the introduced symmetries define a mapping between solutions that induce *equivalent* residues of $soft(\mathcal{F}^i)$, i.e., solutions ρ and ρ' where each residue subsumes the other. Residue subsumption as we have defined it here is more general than symmetry breaking as we don't need ρ and ρ' to subsume each other, we only require that ρ subsumes ρ'.

Example 3. Continuing with Example 2. Given that β which sets b_1^1, b_2^2 and b_3^3 to true residue subsumes all other solutions to $\mathbf{card}^3$, Prop. 1 shows that the SAT solver need only check if $\mathcal{F}^3$ is satisfiable under β. This makes intuitive sense, we have found 3 cores over 3 soft clauses, indicating that they all need to be relaxed. Fu & Malik would have the SAT solver refute all 12 solutions to $\mathbf{card}^3$. Note also that any form of symmetry reduction restricted to blocking solutions with equivalent residues, would still have to check at least 5 solutions.

5.1 Exploiting Residue Subsumption

There are two issues that arise when trying to exploit residue subsumption in core based solvers.

The first issue is that the correctness of the algorithms used in these solvers relies on certain properties of the formulas $\mathcal{F}^i$ constructed at each iteration. For example, a common way of proving the correctness of these algorithms is to prove that the original MAXSAT formula $\mathcal{F}$ is MaxSat reducible to each $\mathcal{F}^i$ [4]. If we alter $\mathbf{card}^i$ so as to block residue subsumed solutions, this would change $\mathcal{F}^i$ and we would have to verify that the new formula continues to satisfy the properties required of it by each algorithm. Unfortunately, proving these properties can be quite intricate.

However, this first issue is easily resolved. Let $\mathcal{F}^{i+}$ be a modification of $\mathcal{F}^i$ that blocks some residue subsumed solutions of $\mathbf{card}^i$. Applying Theorem 1 to $\mathcal{F}^{i+}$ we see that any core of $\mathcal{F}^{i+}$ returned by the SAT solver is a union over the solutions of $\mathbf{card}^i$ that have not been blocked in $\mathcal{F}^{i+}$. Then Prop. 1 shows that the returned core of $\mathcal{F}^{i+}$ is in fact also a core of $\mathcal{F}^i$. That is, the SAT solver will still return cores of $\mathcal{F}^i$ even if it has been modified to block residue subsumed solutions of $\mathbf{card}^i$. Furthermore, if the SAT solver returns a satisfying assignment this assignment must also satisfy $\mathcal{F}^i$ as it satisfies a more constrained version of $\mathcal{F}^i$.

Core based algorithms use the SAT solver as a black-box, expecting it to return a solution or core of $\mathcal{F}^i$, and as explained above modifying the solver so as to block residue subsumed solutions of $\mathbf{card}^i$ does not impact this functionality. Thus, any core based algorithm can exploit residue subsumption without affecting its correctness.

The second issue is more difficult to resolve, and remains an open research question. This is the issue of modifying the SAT solver so as to efficiently block residue subsumed solutions of $\mathbf{card}^i$. Potentially, extra clauses could be added to $\mathbf{card}^i$, an SMT-like theory could be consulted during search, or some modification could be made to the solver's search. How best to accomplish this is a problem we are continuing to work on.

6 Conclusion

In this paper we have presented a formal characterization of the relationship between the cores computed by core based MAXSAT algorithms using cardinality constraints and cores of the original MAXSAT formula. Our main result allowed us to identify a condition, residue subsumption, that could potentially be used to improve these algorithms.

References

1. Ansótegui, C., Bonet, M.L., Gabàs, J., Levy, J.: Improving SAT-based weighted MaxSAT solvers. In: Milano, M. (ed.) CP 2012. LNCS, vol. 7514, pp. 86–101. Springer, Heidelberg (2012)
2. Ansótegui, C., Bonet, M.L., Levy, J.: Solving (weighted) partial MaxSAT through satisfiability testing. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 427–440. Springer, Heidelberg (2009)
3. Ansótegui, C., Bonet, M.L., Levy, J.: A new algorithm for weighted partial MaxSAT. In: Proceedings of the AAAI National Conference, AAAI (2010)
4. Ansótegui, C., Bonet, M.L., Levy, J.: SAT-based MaxSAT algorithms. Artificial Intelligence 196, 77–105 (2013)
5. Ansótegui, C., Gabàs, J.: Solving (weighted) partial MaxSAT with ILP. In: International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR), pp. 403–409 (2013)
6. Berre, D.L., Parrain, A.: The sat4j library, release 2.2. Journal on Satisfiability, Boolean Modeling and Computation (JSAT) 7(2-3), 6–59 (2010)
7. Chvátal, V., Reed, B.A.: Mick gets some (the odds are on his side). In: Symposium on Foundations of Computer Science (FOCS). pp. 620–627 (1992)
8. Davies, J., Bacchus, F.: Postponing optimization to speed up MaxSAT solving. In: Schulte, C. (ed.) CP 2013. LNCS, vol. 8124, pp. 247–262. Springer, Heidelberg (2013)
9. Fu, Z., Malik, S.: On solving the partial max-sat problem. In: Biere, A., Gomes, C.P. (eds.) SAT 2006. LNCS, vol. 4121, pp. 252–265. Springer, Heidelberg (2006)
10. Heras, F., Larrosa, J., Oliveras, A.: MiniMaxSAT: An efficient weighted Max-SAT solver. Journal of Artificial Intelligence Research (JAIR) 31, 1–32 (2008)
11. Koshimura, M., Zhang, T., Fujita, H., Hasegawa, R.: QMaxSAT: A partial Max-SAT solver. Journal on Satisfiability, Boolean Modeling and Computation (JSAT) 8(1/2), 95–100 (2012)
12. Li, C.M., Manyà, F.: MaxSAT, hard and soft constraints. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, pp. 613–631. IOS Press (2009)
13. Li, C.M., Manyà, F., Mohamedou, N.O., Planes, J.: Resolution-based lower bounds in MaxSAT. Constraints 15(4), 456–484 (2010)
14. Manquinho, V., Marques-Silva, J., Planes, J.: Algorithms for weighted boolean optimization. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 495–508. Springer, Heidelberg (2009)
15. Morgado, A., Heras, F., Marques-Silva, J.: Improvements to core-guided binary search for MaxSAT. In: Cimatti, A., Sebastiani, R. (eds.) SAT 2012. LNCS, vol. 7317, pp. 284–297. Springer, Heidelberg (2012)