DRMaxSAT with MaxHS: First Contact *

A. Morgado[1], A. Ignatiev[1,4], M. L. Bonet[2], J. Marques-Silva[1], and S. Buss[3]

[1] Faculty of Science, University of Lisbon, Portugal
{ajmorgado,aignatiev,jpms}@ciencias.ulisboa.pt
[2] Computer Science, Universidad Politécnica de Cataluña, Barcelona, Spain
bonet@cs.upc.edu
[3] Department of Mathematics, University of California, San Diego, USA
sbuss@ucsd.edu
[4] ISDCT SB RAS, Irkutsk, Russia

Abstract. The proof system of Dual-Rail MaxSAT (DRMaxSAT) was recently shown to be capable of efficiently refuting families of formulas that are well-known to be hard for resolution, concretely when the MaxSAT solving approach is either MaxSAT resolution or core-guided algorithms. Moreover, DRMaxSAT based on MaxSAT resolution was shown to be stronger than general resolution. Nevertheless, existing experimental evidence indicates that the use of MaxSAT algorithms based on the computation of minimum hitting sets (MHSes), i.e. MaxHS-like algorithms, are as effective, and often more effective, than core-guided algorithms and algorithms based on MaxSAT resolution. This paper investigates the use of MaxHS-like algorithms in the DRMaxSAT proof system. Concretely, the paper proves that the propositional encoding of the pigenonhole and doubled pigenonhole principles have polynomial time refutations when the DRMaxSAT proof system uses a MaxHS-like algorithm.

1 Introduction

The practical success of Conflict-Driven Clause Learning (CDCL) Boolean Satisfiability (SAT) solvers demonstrates the reach of the (general) resolution proof system. Nevertheless, from a proof complexity point of view, resolution is generally viewed as a rather weak proof system. As a result, there have been recent efforts towards developing practically efficient implementations of stronger proof systems. Concrete examples include extended resolution [2, 19, 21] (ExtRes), the cutting planes (CP) proof system [4, 17] and, more recently, the dual rail (DR) maximum satisfiability (MaxSAT) (i.e. DRMaxSAT) proof system [9, 20][1]. Although expected to be weaker than CP, and so weaker than ExtRes, DRMaxSAT can in practice build on practically efficient MaxSAT solvers, and so indirectly tap on the practical efficiency of CDCL SAT solvers.

* This work was supported by FCT grants ABSOLV (PTDC/CCI-COM/28986/2017), FaultLocker (PTDC/CCI-COM/29300/2017), SAFETY (SFRH/BPD/120315/2016), and SAMPLE (CEECIND/04549/2017); grant TIN2016-76573-C2-2-P (TASSAT 3); and Simons Foundation grant 578919.

[1] Despite ongoing efforts, and with the exception of families of problem instances known to be hard for resolution, the performance of implementations of proof systems stronger than resolution is still far from what CDCL SAT solvers achieve in practice.

The DRMaxSAT proof system translates a proposition formula using the dual-rail encoding [11, 20], and then uses a MaxSAT algorithm. Initial results for DRMaxSAT focused on MaxSAT resolution and on core-guided MaxSAT solvers [20]. Nevertheless, the empirical evidence from earlier work also indicated the MaxSAT solvers based on the iterative computation of minimum hitting sets (MHSes), concretely MaxHS [13] and LMHS [34], performed comparably, if not better, than the use of core-guided MaxSAT solvers, on different families of problems known to be hard for resolution. The reason for this good performance was left as open research.

This paper investigates MHS-based MaxSAT algorithms (also referred to MaxHS-like) on two dual-rail encoded families of problems, encoding respectively the pigeonhole and the doubled pigeonhole problems. These two problems are known to be hard to general resolution, and were studied in earlier work [9, 20]. Moreover, the paper proves that these two families of problems can be refuted in polynomial time when DRMaxSAT proof system uses a MaxHS-like MaxSAT solver. These results thus provide a theoretical justification for the good performance observed in practice for MaxHS-like algorithms (in both families of problems).

2 Motivation

Experimental results presented in [20] and later in [9] confirmed the ability of the DRMaxSAT proof system to refute pigeonhole principle formulas and also doubled pigeonhole formulas in polynomial time. (These families of formulas are defined later.) Figure 1 summarizes the results from both works.[2] Here, formulas encoding the pigeonhole and doubled pigeonhole principles were generated with up to 100 holes[3]. Core-guided MaxSAT solvers are represented by MSCG [28, 30] and OpenWBO16 [27], while Eva500a [31] acts for *core-guided* MaxSAT resolution. For comparison purposes, Figure 1 also depicts the performance of lingeling [5, 6] as best performing CDCL SAT solver (see [9,20]). Additionally, earlier work [9,20] assessed the performance of hitting set based MaxSAT solvers MaxHS [3, 13–15] and LMHS [34]. Note that the efficiency of the default versions of MaxHS and LMHS is not surprising and can be attributed to a number of powerful heuristics used [3, 14, 15, 34] for improving the *basic MaxHS algorithm* of MaxHS and LMHS [13]. However, the remarkable efficiency of this basic algorithm (referred to as LMHS-nes in Figure 1) is yet to be explained. The following sections represent a first attempt to understand the power of the *basic* hitting set based MaxSAT with respect to two families of pigeonhole formulas, namely PHP and 2PHP.

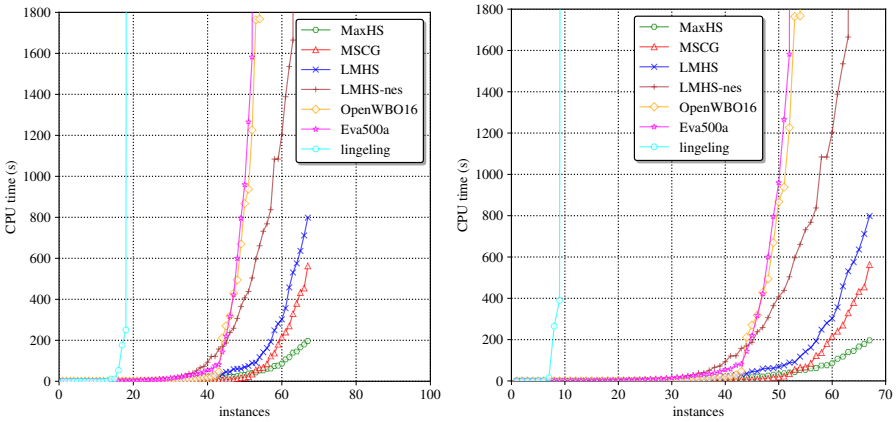
3 Preliminaries

The paper assumes definitions and notation standard in propositional satisfiability (SAT) and maximum satisfiability (MaxSAT) [7].

SAT. A conjunctive normal form (CNF) formula $\mathcal{F}$ is a conjunction of clauses, a clause is a disjunction of literals, and a literal is a variable or its complement. A truth

[2] The plots show the performance of the competitors for a specifically constructed PHP and 2PHP formulas, with $\mathcal{P}$ clauses being disabled. (The notation and rationale will be explained below. A reader can also refer to [9, 20] for details.)

[3] Some of the (2)PHP instances are skipped in the plots. Please, refer to [9, 20] for details.



(a) PHP (b) 2PHP

Fig. 1: Performance of MaxSAT solvers and lingeling on PHP and 2PHP formulas.

assignment $\mathcal{A}$ is a mapping from variables to $\{0, 1\}$. Given $\mathcal{A}$, a clause is satisfied iff at least one of its literals is assigned value 1. A formula is satisfied iff all of its clauses are satisfied. If there is no satisfying assignment for $\mathcal{F}$, then $\mathcal{F}$ is referred to as *unsatisfiable*.

MaxSAT. For unsatisfiable formulas, the maximum satisfiability (MaxSAT) problem is to find an assignment that maximizes the number of satisfied clauses [24]. This paper considers the *partial MaxSAT* problem $\langle \mathcal{H}, \mathcal{S} \rangle$, which allows for a set of *hard* clauses $\mathcal{H}$ (that must be satisfied), and a set of *soft* clauses $\mathcal{S}$ (represent a preference to satisfy these clauses). In the paper, a MaxSAT *solution* represents either a maximum cardinality set of satisfied soft clauses or an assignment that satisfies all hard clauses and also maximizes (minimizes) the number of satisfied (falsified, resp.) soft clauses. The number of clauses falsified by a MaxSAT solution is referred to as its *cost*. A few other optimization problems exist with respect to MaxSAT: (1) computing *minimal correction subsets* (MCSes) and (2) computing *minimal unsatisfiable subsets* (MUSes). Given an unsatisfiable formula $\langle \mathcal{H}, \mathcal{S} \rangle$, $\mathcal{M} \subseteq \mathcal{S}$ is an MUS iff $\langle \mathcal{H}, \mathcal{M} \rangle$ is unsatisfiable and $\forall_{\mathcal{M}' \subsetneq \mathcal{M}} \langle \mathcal{H}, \mathcal{M}' \rangle$ is satisfiable. Given an unsatisfiable formula $\langle \mathcal{H}, \mathcal{S} \rangle$, $\mathcal{C} \subseteq \mathcal{S}$ is an MCS iff $\langle \mathcal{H}, \mathcal{S} \setminus \mathcal{C} \rangle$ is satisfiable and $\forall_{\mathcal{C}' \subsetneq \mathcal{C}} \langle \mathcal{H}, \mathcal{S} \setminus \mathcal{C}' \rangle$ is unsatisfiable. MCSes and MUSes of an unsatisfiable formula are known to be connected through the *hitting set duality* [8, 33], i.e. MCSes are minimal hitting sets of MUSes, and vice versa.

MaxHS Algorithm. Many algorithms for MaxSAT have been proposed over the years [24]. The most widely investigated ones can be broadly organized into branch and bound [24], iterative-search [4, 18, 22], core-guided [1, 18, 25, 26, 28, 29, 31], and minimum hitting sets [13, 34]. This paper focuses solely on the minimum hitting set MaxSAT algorithm referred to as basic MaxHS [13]. Its setup is shown in Algorithm 1. MaxHS builds on the minimal hitting set duality between MCSes and MUSes. Every iteration of the algorithm computes a minimum size hitting set (MHS) h of a set K of unsatisfiable cores found so far and checks if h is an MCS of the formula, i.e. it tests satisfiability of $\mathcal{H} \cup \mathcal{S} \setminus h$. If it is, the algorithm returns a model of $\mathcal{H} \cup \mathcal{S} \setminus h$. Otherwise, a new unsatisfiable core is extracted, added to K, and the algorithm proceeds.

Algorithm 1: MaxHS Algorithm

Input: MaxSAT formula $\langle \mathcal{H}, \mathcal{S} \rangle$
Output: MaxSAT assignment μ

```
1 K ← ∅
2 while true do
3     h ← MinimumHS(K)
4     (st, μ) ← SAT(H ∪ S \ h)
5     if st then return μ            // If st, then μ is an assignment
6     else K ← K ∪ {μ}               // Otherwise, μ is a core
```

Dual-Rail MaxSAT. The proof system of *dual-rail MaxSAT* (DRMaxSAT) was proposed in [20] and heavily relies on the variant of the *dual-rail encoding* (DRE) [11, 32]. Let $\mathcal{F}$ be a CNF formula on the set of N variables $X = \{x_1 \ldots, x_N\}$. Given $\mathcal{F}$, the dual-rail MaxSAT encoding [9, 20] creates a (Horn) MaxSAT problem $\langle \mathcal{H}, \mathcal{S} \rangle$, where $\mathcal{H}$ is the set of hard clauses and $\mathcal{S}$ is the set of soft clauses s.t. $|\mathcal{S}| = 2N$. For each variable $x_i \in X$, DRE associates two new variables p_i and n_i, where $p_i = 1$ iff $x_i = 1$, and $n_i = 1$ iff $x_i = 0$. It also adds the soft clauses (p_i) and (n_i) to $\mathcal{S}$ while adding a hard clause $(\neg p_i \vee \neg n_i)$ to $\mathcal{H}$, to ensure that $x_i = 1$ and $x_i = 0$ are not set simultaneously. (Hard clauses of this form are referred to as $\mathcal{P}$ clauses.) For each clause $c \in \mathcal{F}$, we obtain a hard clause $c' \in \mathcal{H}$, as follows: if $x_i \in c$, then $\neg n_i \in c'$, and if $\neg x_i \in c$ then $\neg p_i \in c'$. The formula encoded by DRE is then $\mathsf{DREnc}(\mathcal{F}) \triangleq \langle \mathcal{H}, \mathcal{S} \rangle$. One of the major results of [20] is the following.

Theorem 1. *CNF formula $\mathcal{F}$ is satisfiable iff there exists a truth assignment that satisfies $\mathcal{H}$ and at least N clauses in $\mathcal{S}$.*

Applying DRE followed either by MaxSAT resolution [10, 23] or core-guided MaxSAT solving [29] consitutes the DRMaxSAT proof system [9], which is able to refute pigeonhole principle [20] and doubled pigeonhole principle [9] in polynomial time.

4 Polynomial Time Refutations with DRMaxHS

4.1 Pigeonhole Principle

Definition 1 (PHP [12]). *The pigeonhole principle states that if $m + 1$ pigeons are distributed by m holes, then at least one hole contains more than one pigeon.*

The propositional encoding of the PHP_m^{m+1} problem is as follows. Let the variables be x_{ij}, with $i \in [m+1]$, $j \in [m]$, with $x_{ij} = 1$ iff the i^{th} pigeon is placed in the j^{th} hole. The constraints state that each pigeon must be placed in at least one hole, and each hole must not have more than one pigeon: $\bigwedge_{i=1}^{m+1} \mathsf{AtLeast1}(x_{i1}, \ldots, x_{im}) \wedge \bigwedge_{j=1}^{m} \mathsf{AtMost1}(x_{1j}, \ldots, x_{m+1,j})$. Each AtLeast1 constraint can be encoded with a single clause. For the AtMost1 constraints there are different encodings, including [7, 16, 35]. In this work, we will consider the pairwise cardinality constraint encoding [7], which encodes $\mathsf{AtMost1}(x_{1j}, \ldots, x_{m+1,j})$ as $\wedge_{i_1=1}^{m} \wedge_{i_2=i_1+1}^{m+1} (\neg x_{i_1 j} \vee \neg x_{i_2 j})$.

The reduction of the PHP_m^{m+1} problem into MaxSAT using the Dual-Rail encoding is as follows [20]. With each variable x_{ij}, $i \in [m+1], j \in [m]$, we associate two new variables: n_{ij} and p_{ij}. The set of clauses $\mathcal{P}$ is $\{(\neg n_{ij} \vee \neg p_{ij}) \,|\, i \in [m+1], j \in [m]\}$. Let $\mathcal{L}_i$ represent the encoding of each AtLeast1 constraint; $\mathcal{L}_i = (\neg n_{i1} \vee$

$\ldots \vee \neg n_{im})$, and $\mathcal{M}_j$ represent the encoding of each AtMost1 constraint; $\mathcal{M}_j = \wedge_{i_1=1}^{m} \wedge_{i_2=i_1+1}^{m+1} (\neg p_{i_1 j} \vee \neg p_{i_2 j})$. The soft clauses $\mathcal{S}$ are given by $\{(n_{11}), \ldots, (n_{(m+1)m}), (p_{11}), \ldots, (p_{(m+1)m})\}$, with $|\mathcal{S}| = 2m(m+1)$. The complete encoding of PHP is:

$$\mathsf{DREnc}\left(\mathrm{PHP}_m^{m+1}\right) \triangleq \langle \mathcal{H}, \mathcal{S} \rangle = \left\langle \wedge_{i=1}^{m+1} \mathcal{L}_i \wedge \wedge_{j=1}^{m} \mathcal{M}_j \wedge \mathcal{P}, \mathcal{S} \right\rangle \tag{1}$$

From Theorem 1 [20], unsatisfiability of PHP_m^{m+1} implies that $\mathsf{DREnc}\left(\mathrm{PHP}_m^{m+1}\right)$ has a MaxSAT cost of at least $m(m+1)+1$. The following shows that the basic MaxHS algorithm can derive this MaxSAT cost for $\mathsf{DREnc}\left(\mathrm{PHP}_m^{m+1}\right)$ in polynomial time. Observe that if the $\mathcal{P}$ clauses from $\mathsf{DREnc}\left(\mathrm{PHP}_m^{m+1}\right)$ are ignored, then the formula can be partitioned into the disjoint formulas $\mathcal{L}_i$, $i \in [m+1]$, and the disjoint formulas $\mathcal{M}_j$, $j \in [m]$. Thus, one can compute a solution for each of these formulas separately and obtain a lower bound on the MaxSAT solution for the complete formula $\mathsf{DREnc}\left(\mathrm{PHP}_m^{m+1}\right)$. We show that the contribution of each $\mathcal{L}_i$ to the total cost is 1. Since there are $m+1$ such formulas, the contribution of all $\mathcal{L}_i$ formulas is $m+1$. Then, we show that each $\mathcal{M}_j$ contributes with a cost of m, and since there are m such formulas, the contribution of all $\mathcal{M}_j$ formulas is m^2. Therefore, we have a lower bound on the total cost of $m(m+1)+1$, proving the original formula to be unsatisfiable.

Proposition 1. *Given a formula $\langle \mathcal{L}_i, \mathcal{S} \rangle$, where $\mathcal{L}_i$ and $\mathcal{S}$ are from $\mathsf{DREnc}\left(\mathit{PHP}_m^{m+1}\right)$, there is an execution of the basic MaxHS algorithm that computes a MaxSAT solution of cost 1 in polynomial time.*

Proof. (Sketch) Consider Algorithm 1. In the first iteration, an empty MHS is computed in line 3. The SAT solver (line 4) tests the satisfiability of the only clause $(\neg n_{i1} \vee \ldots \vee \neg n_{im})$ with the complement of the MHS as unit clauses, that is, $(n_{i1}), \ldots, (n_{im})$. The formula is unsatisfiable and a new set to hit is added to K, corresponding to the complete set of variables $\{n_{i1}, \ldots, n_{im}\}$ (line 6). Observe that the SAT solver proves the formula to be unsatisfiable by unit propagation.

In the second iteration, K contains only 1 set to hit, in which any of its elements can be selected as a minimum hitting set. The SAT solver tests for the satisfiability of $(\neg n_{i1} \vee \ldots \vee \neg n_{im})$ with the complement of the (unit size) MHS, reporting the formula to be satisfiable. The reported cost of the solution is 1. □

Before presenting the result for the $\mathcal{M}_j$ formulas, we make a few observations.

Observation 1. *Consider a complete graph G, i.e. a* clique*, of $m+1$ vertices. A vertex cover of a G can be computed in polynomial time and has size m. Simply randomly pick one of the vertices to be out of the cover.*

Observation 2. *Let graph G be composed of a clique of size $m-1$ plus one extra vertex that is connected to at least one of the vertices of the clique. Then a vertex cover of G has size $m-2$, and can be computed in polynomial time by including all vertices, except for the two of them that have the* smallest degree*, i.e.* number of adjacents*.*

Observation 3. *Assume two possible cases: (1) graph G is a clique, or (2) G is composed of a clique plus one extra vertex connected to some of the vertices of the clique. Then checking which of the cases holds can be done by checking if all vertices of G have the same degree (in this case G is a clique).*

Proposition 2. *Given a formula $\langle \mathcal{M}_j, \mathcal{S} \rangle$ s.t. $\mathcal{M}_j$ and $\mathcal{S}$ are from $\mathsf{DREnc}\left(PHP_m^{m+1}\right)$, there is an execution of the basic MaxHS algorithm that computes a MaxSAT solution of cost m in polynomial time.*

Proof. (Sketch) The idea of the proof is to show that there is a possible ordering of the set of cores returned by the SAT solver that will make the sets in K to induce a graph that is either a clique or composed of a clique plus one extra vertex connected to some of the other vertices. Then from the previous observations a MHS can be computed in polynomial time. In the final iteration, the graph induced by the sets in K will correspond to a clique of size $m+1$. As such, the final MHS will have size m, thus reporting a solution with a cost of m.

Consider an order of the clauses to consider in $\mathcal{M}_j$, induced by the following choice of variables. First, consider the clauses that contain $\{p_{1j}, p_{2j}\}$ (only 1 clause); then the clauses that contain $\{p_{1j}, p_{2j}, p_{3j}\}$ (2 more clauses); then $\{p_{1j}, p_{2j}, p_{3j}, p_{4j}\}$ (3 more clauses); and so on until all variables/clauses are considered. Observe that due to the structure of $\mathcal{M}_j$, every unsatisfiable core returned by the SAT solver is of size 2 (obtained by unit propagation). Consequently, the chosen order of variables implies that all pairs of the current set of variables are added to K before considering a new variable. The first set added to K this way is $\{p_{1j}, p_{2j}\}$, followed by $\{p_{1j}, p_{3j}\}$, $\{p_{2j}, p_{3j}\}$, etc.

Since sets in K are pairs, then each set can be regarded as an edge of an induced graph. Given the previous ordering of the variables (and consequently of the sets in K), the induced graph forms a "growing" clique, that is, it is either a clique with all the variables considered so far, or it is a clique with the previous variables plus a new variable connected to some of the previous variables.

Finally, since each clause in $\mathcal{M}_j$ produces an unsatisfiable core returned by the SAT solver (corresponding to a new set to hit in K), then the total number of iterations is equal to the number of clauses in $\mathcal{M}_j$ plus 1, which is $C_2^{m+1} + 1 = \frac{(m+1)m}{2} + 1$. □

4.2 Doubled Pigeonhole Principle

Here we consider an extension of the pigeonhole principle, which targets m holes, $2m+1$ pigeons, and each hole has a maximum capacity of 2 pigeons.

Definition 2 (2PHP). *The doubled pigeonhole principle states that if $2m+1$ pigeons are distributed evenly by m holes, then at least one hole contains more than 2 pigeons.*

The propositional encoding of the 2PHP_m^{2m+1} problem is as follows. Let the variables be x_{ij}, with $i \in [2m+1]$, $j \in [m]$, with $x_{ij} = 1$ iff pigeon i is placed in the hole j. The constraints state that each pigeon must be placed in at least one hole, and each hole must not have more than 2 pigeons, $\bigwedge_{i=1}^{2m+1} \mathsf{AtLeast1}(x_{i1}, \ldots, x_{im}) \wedge \bigwedge_{j=1}^{m} \mathsf{AtMost2}(x_{1j}, \ldots, x_{(2m+1)j})$. Similar to the PHP case, each $\mathsf{AtLeast1}$ constraint is encoded with a single clause. The $\mathsf{AtMost2}$ constraints are encoded into clauses using the pairwise encoding [7], as $\bigwedge_{i_1=1}^{2m-1} \bigwedge_{i_2=i_1+1}^{2m} \bigwedge_{i_3=i_2+1}^{2m+1} (\neg x_{i_1 j} \vee \neg x_{i_2 j} \vee \neg x_{i_3 j})$.

The reduction of the 2PHP_m^{2m+1} problem into MaxSAT using the Dual-Rail encoding is similar to the PHP case as follows. With each variable x_{ij}, $i \in [2m+1]$, $j \in [m]$, we associate two new variables: n_{ij} and p_{ij}. The $\mathcal{P}$ clauses are encoded as $\{(\neg n_{ij} \vee \neg p_{ij}) \,|\, i \in [2m+1], j \in [m]\}$. $\mathcal{L}_i$ represents the encoding of an $\mathsf{AtLeast1}$ constraint for pigeon i: $\mathcal{L}_i = (\neg n_{i1} \vee \ldots \vee \neg n_{im})$. $\mathcal{M}_j$ represents the encoding of an $\mathsf{AtMost2}$ constraint for hole j: $\mathcal{M}_j = \wedge_{i_1=1}^{2m-1} \wedge_{i_2=i_1+1}^{2m} \wedge_{i_3=i_2+1}^{2m+1} (\neg p_{i_1 j} \vee \neg p_{i_2 j} \vee \neg p_{i_3 j})$.

The soft clauses $\mathcal{S}$ are given by $\{(n_{11}), \ldots, (n_{(2m+1)m}), (p_{11}), \ldots, (p_{(2m+1)m})\}$ with $|\mathcal{S}| = 2m(2m+1)$. Thus, the complete encoding of 2PHP is:

$$\mathsf{DREnc}\left(2\mathrm{PHP}_m^{2m+1}\right) \triangleq \langle \mathcal{H}, \mathcal{S} \rangle = \left\langle \wedge_{i=1}^{2m+1} \mathcal{L}_i \wedge \wedge_{j=1}^{m} \mathcal{M}_j \wedge \mathcal{P}, \mathcal{S} \right\rangle \tag{2}$$

$2\mathrm{PHP}_m^{2m+1}$ is unsatisfiable if and only if the cost of $\mathsf{DREnc}\left(2\mathrm{PHP}_m^{2m+1}\right)$ is at least $m(2m+1)+1$ (from Theorem 1 [20]). Similar to the PHP case, if the $\mathcal{P}$ clauses from $\mathsf{DREnc}\left(2\mathrm{PHP}_m^{2m+1}\right)$ are ignored, then the resulting formula can be partitioned into the disjoint formulas $\mathcal{L}_i$ ($i \in [2m+1]$) and the disjoint formulas $\mathcal{M}_j$ ($j \in [m]$). One can compute a MaxSAT solution for each of $\mathcal{L}_i$ and $\mathcal{M}_j$ separately and obtain a lower bound on the cost of the MaxSAT solution for the complete formula $\mathsf{DREnc}\left(2\mathrm{PHP}_m^{2m+1}\right)$. Processing each formula $\mathcal{L}_i$ can be done as in the PHP case (see Proposition 1).

As shown below, the contribution of each $\mathcal{M}_j$ to the MaxSAT cost is $2m-1$, and since there are m such formulas, then the contribution of all $\mathcal{M}_j$ formulas is $m(2m-1)$. As a result, the lower bound on the total cost for $\mathsf{DREnc}\left(2\mathrm{PHP}_m^{2m+1}\right)$ is $m(2m-1)+2m+1 = m(2m+1)+1$, thus, proving formula $2\mathrm{PHP}_m^{2m+1}$ to be unsatisfiable. We also show that the basic MaxHS algorithm is able to derive the MaxSAT cost for each $\mathcal{M}_j$ in polynomial time. To proceed, let us first show that the following holds.

Proposition 3. *Let X be a set of elements of size $|X| = s+2$. Let K be a set of all possible triples $\{x_i, x_j, x_r\}$ of elements of X, $1 \leq i < j < r \leq s+2$. Then any set of s different elements from X is a minimum hitting set for K.*

Proof. (Sketch) Proof by induction on s. Base case, $s = 1$. X contains 3 elements and, thus, $|K| = 1$. Randomly selecting 1 element from X is an MHS of the only set to hit.

Step case $s = n+1$. Suppose that K contains all the combinations of size 3 of elements from X, with $|X| = n+3$. Select randomly 1 element from X, let it be p. Partition K into the sets that contain p and the sets to hit that do not contain p. The sets to hit that dot not contain p form a minimum hitting set subproblem, with an initial set of elements of size $n+2$ (i.e. $s' = n$), and all sets to hit of size 3. By induction hypotheses, this minimum hitting set subproblem has a solution of size n (and is minimum). The solution to the subproblem is not a an MHS of K. This can be seen by considering w.l.o.g. the solution to be $\{x_1, \ldots, x_n\}$, then in K there is the set to hit $\{x_{n+1}, x_{n+2}, p\}$, which is not covered by $\{x_1, \ldots, x_n\}$. Nevertheless, we can extend the solution of the subproblem into a MHS of K by including p in the solution. The size of the new solution is $n+1 = s$. Note that a smaller solution is not possible; otherwise, we could disregard p from the solution, and that would correspond to a solution to the subproblem of the induction hypothesis with fewer than $n = s'$ elements, thus, contradicting the induction hypothesis. □

Proposition 4. *Let X be a set of elements of size $|X| = s+2$, and an additional element p not in X. Let K be a set of all possible triples $\{x_i, x_j, x_r\}$ of elements of X, $1 \leq i < j < r \leq s+2$, together with some (not all) triples $\{x_i, x_j, p\}$, $x_i, x_j \in X$, $1 \leq i < j \leq s+2$. A minimum hitting set of K has size s and does not contain p.*

Proof. (Sketch) Consider by contradiction that the size of the solution is smaller than s. Since the sets to hit contain all the possible combinations of 3 elements from X, then these sets would have a solution smaller than s, which contradicts Proposition 3.

Consider now by contradiction that the solution has size s and includes p, then we could disregard p from the solution, and hit all the possible combinations of 3 elements from X with fewer than s elements, again contradicting Proposition 3.

Consider now that h is a set with s elements from X not including p. Suppose w.l.o.g. that $\{x_{s+1}, x_{s+2}, p\}$ is missing from the set of elements to hit, and that $h = \{x_1, \cdots, x_s\}$. Since h has size s, then by Proposition 3, h covers the sets to hit that do not contain p. On the other hand, if a set to hit contains p, then it doesn't contain both x_{s+1} and x_{s+2}. So it contains an element from h. Thus h is a MHS of K. □

Proposition 5. *Given a formula $\langle \mathcal{M}_j, \mathcal{S} \rangle$ s.t. $\mathcal{M}_j$ and $\mathcal{S}$ are from DREnc $\left(2PHP_m^{2m+1}\right)$, there is execution of the basic MaxHS algorithm that computes a MaxSAT solution of cost $2m-1$ in polynomial time.*

Proof. (Sketch) The proof illustrates a possible setup of the MHS-algorithm that does a polynomial number of iterations s.t. each minimum hitting set is computed in polynomial time. This setup is achieved by ordering the cores computed by the SAT solver (line 4 of Algorithm 1). Similar to the PHP case, we can order the clauses in the SAT solver, by considering an order on the variables. We consider the clauses that contain $\{p_{1j}, p_{2j}, p_{3j}\}$ (only 1 clause), then the clauses that contain $\{p_{1j}, p_{2j}, p_{3j}, p_{4j}\}$ (the 3 clauses $\neg p_{1j} \vee \neg p_{2j} \vee \neg p_{4j}$, $\neg p_{1j} \vee \neg p_{3j} \vee \neg p_{4j}$ and $\neg p_{2j} \vee \neg p_{3j} \vee \neg p_{4j}$), and so on until all variables/clauses are considered. In contrast to the PHP case, when considering the clauses with a new element, these are also ordered. For example, after considering all clauses with $\{p_{1j}, p_{2j}, p_{3j}, p_{4j}\}$, we will consider the clauses with new element p_{5j}. We order these clauses by first considering the clauses that contain $\{p_{1j}, p_{2j}, p_{5j}\}$ (1 clause), then the clauses that contain $\{p_{1j}, p_{2j}, p_{3j}, p_{5j}\}$ (2 more clauses), and finally the clauses that contain $\{p_{1j}, p_{2j}, p_{3j}, p_{4j}, p_{5j}\}$ (3 more clauses). Note that, the new sets to hit include the new element being added (in the example above the element p_{5j}). On the other hand, by Proposition 4, the minimum hitting set solution does not include the element being added. As such, if we disregard the new element being added in the new sets to hit, then we have pairs which can be regarded as edges of a graph. The graph induced by the pairs in the hitting sets will be a growing clique, as in the PHP case Proposition 2. The orderings of the variables guarantee that the sets to hit in K either contain all the possible combinations of size 3 of the variables we are considering, or they induce a "growing" clique. In the first case a minimum hitting set is obtained in polynomial time using the result of Proposition 3. For the second case, we obtain a minimum hitting set in polynomial time similarly to Proposition 2, using Proposition 4. The process of creating a core (and the corresponding set to hit in K) is repeated for each clause in $\mathcal{M}_j$, thus the total number of iterations is equal to the number of clauses plus 1, which is $\mathcal{C}_3^{2m+1} + 1 = \frac{(2m+1)(2m)(2m-1)}{6} + 1$. Additionally, the reported cost corresponds to the size of the MHS found in the last iteration, i.e., when all variables are considered. Thus, by Proposition 3, the reported cost is $2m-1$. □

5 Conclusions

This paper is motivated by the unexpected good performance of MaxHS-like MaxSAT algorithms on dual-rail encoded families of instances that are hard for general resolution [9, 20]. We prove that for the PHP and 2PHP principles, the DRMaxSAT proof system has polynomial time refutations when a MaxHS-like algorithm is used. Future research will seek to understand how MaxHS-like algorithms compare with core-guided algorithms in the DRMaxSAT proof system.

References

1. Ansótegui, C., Bonet, M.L., Levy, J.: SAT-based MaxSAT algorithms. Artif. Intell. **196**, 77–105 (2013)
2. Audemard, G., Katsirelos, G., Simon, L.: A restriction of extended resolution for clause learning SAT solvers. In: AAAI (2010)
3. Bacchus, F., Hyttinen, A., Järvisalo, M., Saikko, P.: Reduced cost fixing in maxsat. In: CP. pp. 641–651 (2017)
4. Berre, D.L., Parrain, A.: The Sat4j library, release 2.2. JSAT **7**(2-3), 59–6 (2010)
5. Biere, A.: Lingeling, plingeling and treengeling entering the SAT competition 2013. In: Balint, A., Belov, A., Heule, M., Järvisalo, M. (eds.) Proceedings of SAT Competition 2013, Department of Computer Science Series of Publications B, vol. B-2013-1, pp. 51–52. University of Helsinki (2013)
6. Biere, A.: Lingeling essentials, A tutorial on design and implementation aspects of the SAT solver lingeling. In: Pragmatics of SAT workshop. p. 88 (2014)
7. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (2009)
8. Birnbaum, E., Lozinskii, E.L.: Consistent subsets of inconsistent systems: structure and behaviour. J. Exp. Theor. Artif. Intell. **15**(1), 25–46 (2003)
9. Bonet, M.L., Buss, S., Ignatiev, A., Marques-Silva, J., Morgado, A.: Maxsat resolution with the dual rail encoding. In: AAAI. pp. 6565–6572 (2018)
10. Bonet, M.L., Levy, J., Manyà, F.: Resolution for Max-SAT. Artif. Intell. **171**(8-9), 606–618 (2007)
11. Bryant, R.E., Beatty, D.L., Brace, K.S., Cho, K., Sheffler, T.J.: COSMOS: A compiled simulator for MOS circuits. In: DAC. pp. 9–16 (1987)
12. Cook, S.A., Reckhow, R.A.: The relative efficiency of propositional proof systems. J. Symb. Log. **44**(1), 36–50 (1979)
13. Davies, J., Bacchus, F.: Solving MAXSAT by solving a sequence of simpler SAT instances. In: CP. pp. 225–239 (2011)
14. Davies, J., Bacchus, F.: Exploiting the power of mip solvers in maxsat. In: SAT. pp. 166–181 (2013)
15. Davies, J., Bacchus, F.: Postponing optimization to speed up MAXSAT solving. In: CP. pp. 247–262 (2013)
16. Eén, N., Sörensson, N.: Translating pseudo-Boolean constraints into SAT. JSAT **2**(1-4), 1–26 (2006)
17. Elffers, J., Nordström, J.: Divide and conquer: Towards faster pseudo-boolean solving. In: IJCAI. pp. 1291–1299. ijcai.org (2018)
18. Fu, Z., Malik, S.: On solving the partial MAX-SAT problem. In: SAT. pp. 252–265 (2006)
19. Huang, J.: Extended clause learning. Artif. Intell. **174**(15), 1277–1284 (2010)
20. Ignatiev, A., Morgado, A., Marques-Silva, J.: On tackling the limits of resolution in SAT solving. In: SAT. pp. 164–183 (2017)
21. Kiesl, B., Rebola-Pardo, A., Heule, M.J.H.: Extended resolution simulates DRAT. In: IJCAR. Lecture Notes in Computer Science, vol. 10900, pp. 516–531. Springer (2018)
22. Koshimura, M., Zhang, T., Fujita, H., Hasegawa, R.: QMaxSAT: A partial Max-SAT solver. JSAT **8**(1/2), 95–100 (2012)
23. Larrosa, J., Heras, F., de Givry, S.: A logical approach to efficient Max-SAT solving. Artif. Intell. **172**(2-3), 204–233 (2008)
24. Li, C.M., Manyà, F.: MaxSAT. In: Biere et al. [7], pp. 613–631
25. Marques-Silva, J., Planes, J.: On using unsatisfiability for solving maximum satisfiability. CoRR **abs/0712.1097** (2007)

26. Martins, R., Joshi, S., Manquinho, V.M., Lynce, I.: Incremental cardinality constraints for MaxSAT. In: CP. pp. 531–548 (2014)
27. Martins, R., Manquinho, V.M., Lynce, I.: Open-WBO: A modular MaxSAT solver,. In: SAT. pp. 438–445 (2014)
28. Morgado, A., Dodaro, C., Marques-Silva, J.: Core-guided MaxSAT with soft cardinality constraints. In: CP. pp. 564–573 (2014)
29. Morgado, A., Heras, F., Liffiton, M.H., Planes, J., Marques-Silva, J.: Iterative and core-guided MaxSAT solving: A survey and assessment. Constraints **18**(4), 478–534 (2013)
30. Morgado, A., Ignatiev, A., Marques-Silva, J.: MSCG: Robust core-guided MaxSAT solving. JSAT **9**, 129–134 (2015)
31. Narodytska, N., Bacchus, F.: Maximum satisfiability using core-guided MaxSAT resolution. In: AAAI. pp. 2717–2723 (2014)
32. Palopoli, L., Pirri, F., Pizzuti, C.: Algorithms for selective enumeration of prime implicants. Artif. Intell. **111**(1-2), 41–72 (1999)
33. Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. **32**(1), 57–95 (1987)
34. Saikko, P., Berg, J., Järvisalo, M.: LMHS: A SAT-IP hybrid MaxSAT solver. In: SAT. pp. 539–546 (2016)
35. Sinz, C.: Towards an optimal CNF encoding of Boolean cardinality constraints. In: CP. pp. 827–831 (2005)