Horn Maximum Satisfiability: Reductions, Algorithms & Applications *

Joao Marques-Silva[1], Alexey Ignatiev[1,2], and Antonio Morgado[1]

[1] LASIGE, Faculty of Science, University of Lisbon, Portugal
{jpms,aignatiev,ajmorgado}@ciencias.ulisboa.pt
[2] ISDCT SB RAS, Irkutsk, Russia

Abstract. Recent years have witnessed remarkable performance improvements in maximum satisfiability (MaxSAT) solvers. In practice, MaxSAT algorithms often target the most generic MaxSAT formulation, whereas dedicated solvers, which address specific subclasses of MaxSAT, have not been investigated. This paper shows that a wide range of optimization and decision problems are either naturally formulated as MaxSAT over Horn formulas, or permit simple encodings using HornMaxSAT. Furthermore, the paper also shows how linear time decision procedures for Horn formulas can be used for developing novel algorithms for the HornMaxSAT problem.

1 Introduction

Recent years have seen very significant improvements in MaxSAT solving technology [2, 13, 28, 33]. Currently, the most effective MaxSAT algorithms propose different ways for iteratively finding and blocking unsatisfiable cores (or subformulas). However, and despite the promising results of MaxSAT in practical settings, past work has not investigated dedicated approaches for solving subclasses of the MaxSAT problem, with one concrete example being the MaxSAT problem over Horn formulas, i.e. HornMaxSAT [1]. The HornMaxSAT optimization problem is well-known to be NP-hard [23]. In contrast to HornMaxSAT, the decision problem for Horn formulas is well-known to be in P, with linear time algorithms proposed in the 80s [15, 27]. This paper investigates practical uses of MaxSAT subject to Horn formulas, and shows that a vast number of decision and optimization problems are naturally formulated as HornMaxSAT. More importantly, as this paper also shows, a vast number of other decision and optimization problems admit simple HornMaxSAT encodings. One should observe that HornMaxSAT is NP-hard and so, by definition, any decision problem in NP admits a polynomial time reduction to HornMaxSAT. However, for many problems in NP, such reductions are not known, and may result in large (although polynomial) encodings.

With the purpose of exploiting the observation that many optimization and decision problems have natural (and simple) reductions to HornMaxSAT, this paper also proposes a novel algorithm for HornMaxSAT. The new algorithm mimics recent Implicit

* This work was supported by FCT funding of post-doctoral grants SFRH/BPD/103609/2014, SFRH/BPD/120315/2016, and LASIGE Research Unit, ref. UID/CEC/00408/2013.

[1] In contrast, for predicate logic and many of its specializations, Horn clauses are used ubiquitously. This includes logic programming, among many others applications.

Hitting Set algorithms[2] proposed for MaxSAT [13, 33], which exploiting the fact that Horn formulas can be decided in polynomial (linear) time [27], and for which minimal unsatisfiable cores (or MUSes) can be computed in polynomial time [26].

The paper is organized as follows. Section 2 introduces the definitions and notation used in the remainder of the paper. Section 3 shows that a large number of well-known optimization, but also decision problems already have simple HornMaxSAT formulations which, to the best of our knowledge, have not been exploited before. Section 4 proposes a variant of recent general-purpose MaxSAT algorithms, that is dedicated to the HornMaxSAT problem. This section also shows that the new algorithm can elicit automatic abstraction mechanisms. Section 5 overviews additional applications and generic reductions to HornMaxSAT. The potential of the work proposed in this paper is assessed in Section 6, and Section 7 concludes the paper.

2 Preliminaries

The paper assumes definitions and notation standard in propositional satisfiability (SAT) and MaxSAT [8]. Propositional variables are taken from a set $X = \{x_1, x_2, \ldots\}$. A Conjunctive Normal Form (CNF) formula is defined as a conjunction of disjunctions of literals, where a literal is a variable or its complement. CNF formulas can also be viewed as sets of sets of literals, and are represented with letters in calligraphic font, $\mathcal{A}$, $\mathcal{F}$, $\mathcal{H}$, etc. Given a formula $\mathcal{F}$, the set of variables is $\mathsf{vars}(\mathcal{F}) \subseteq X$. A clause is a *goal clause* if all of its literals are negative. A clause is a *definite clause* if it has exactly one positive literal and all the other literals are negative; the number of negative literals may be 0. A clause is Horn if it is either a goal or a definite clause. A truth assignment ν is a map from variables to $\{0, 1\}$. Given a truth assignment, a clause is satisfied if at least one of its literals is assigned value 1; otherwise it is falsified. A formula is satisfied if all of its clauses are satisfied; otherwise it is falsified. If there exists no assignment that satisfies a CNF formula $\mathcal{F}$, then $\mathcal{F}$ is referred to as *unsatisfiable*. (Boolean) Satisfiability (SAT) is the decision problem for propositional formulas, i.e. to decide whether a given propositional formula is satisfiable. Since the paper only considers propositional formulas in CNF, throughout the paper SAT refers to the decision problem for propositional formulas in CNF. Modern SAT solvers instantiate the Conflict-Driven Clause Learning paradigm [8]. For unsatisfiable (or inconsistent) formulas, MUSes (minimal unsatisfiable subsets) represent subset-minimal subformulas that are unsatisfiable (or inconsistent), and MCSes (minimal correction subsets) represent subset-minimal subformulas such that the complement is satisfiable [8].

To simplify modeling with propositional logic, one often represents more expressive constraints. Concrete examples are cardinality constraints and pseudo-Boolean constraints [8]. A cardinality constraint of the form $\sum x_i \leq k$ is referred to as an $\mathsf{AtMost}k$ constraint, whereas a cardinality constraint of the form $\sum x_i \geq k$ is referred to as an $\mathsf{AtLeast}k$ constraint. Propositional encodings of cardinality and pseudo-Boolean constraints is an area of active research [1,4–8,10,16,29,35,37]. The (plain) MaxSAT problem is to find a truth assignment that maximizes the number of satisfied clauses. For the plain MaxSAT problem, all clauses are *soft*, meaning that these may not be satisfied. Variants of the MaxSAT can consider the existence of *hard* clauses, meaning that these

[2] Throughout the paper, these are referred to as MaxHS-family of MaxSAT algorithms.

must be satisfied, and also assign weights to the soft clauses, denoting the *cost* of falsifying the clause; this is referred as the weighted MaxSAT problem, WMaxSAT. When addressing MaxSAT problems with weights, hard clauses are assigned a large weight $\top$. The HornMaxSAT problem corresponds to the MaxSAT problem when all clauses are Horn. If clauses have weights, then HornWMaxSAT denotes the Horn MaxSAT problem when the soft clauses have weights.

Throughout the paper, standard graph and set notations will be used. An undirected graph $G = (V, E)$ is defined by a set V of vertices and a set $E \subseteq \{\{u, v\} \,|\, u, v \in V, u \neq v\}$. Given $G = (V, E)$, the *complement graph* $G = (V, E^C)$ is the graph with edges $\{\{u, v\} \,|\, u, v \in V, u \neq v, \{u, v\} \notin E\}$. Moreover, it is assumed some familiarity with optimization problems defined on graphs, including minimum vertex cover, maximum independent set, maximum clique, among others. Finally, $\leq_P$ is used to represent polynomial time reducibility between problems [12, Section 34.3].

3 Basic Reductions

This section shows that a number of well-known problems can be reduced in polynomial time to the HornMaxSAT problem. Some of the reductions are well-known; we simply highlight that the resulting propositional formulas are Horn.

3.1 Optimization Problems on Graphs

Definition 1 (Minimum Vertex Cover, MinVC)**.** *Given an undirected graph $G = (V, E)$, a vertex cover $T \subseteq V$ is such that for each $\{u, v\} \in E$, $\{u, v\} \cap T \neq \emptyset$. A minimum (or cardinality minimal) vertex cover $T \subseteq V$ is a vertex cover of minimum size*[3]*.*

Reduction 1 (MinVC $\leq_P$ HornMaxSAT)**.** *For $u \in V$, let $x_u = 1$ iff u is* not *included in a vertex cover. For any $\{u, v\} \in E$, add a hard clause $(\neg x_u \vee \neg x_v)$. For each $u \in V$, add a soft clause (x_u). (Any non-excluded vertex $u \in V$ (i.e. $x_u = 0$) is in the vertex cover.)*

Remark 1. The proposed reduction differs substantially from the one originally used for proving HornMaxSAT to be NP-hard [23], but our working assumptions are also distinct, in that we consider hard and soft clauses.

Definition 2 (Maximum Independent Set, MaxIS)**.** *Given an undirected graph $G = (V, E)$, an independent set $I \subseteq V$ is such that for each $\{u, v\} \in E$ either $u \notin I$ or $v \notin I$. A maximum independent set is an independent set of maximum size.*

Reduction 2 (MaxIS $\leq_P$ HornMaxSAT)**.** *One can simply use the previous encoding, by noting the relationship between vertex covers and independent sets. For any $\{u, v\} \in E$, add a hard clause $(\neg x_u \vee \neg x_v)$. For each $u \in V$, add a soft clause (x_u).*

Definition 3 (Maximum Clique, MaxClique)**.** *Given an undirected graph $G = (V, E)$, a clique (or complete subgraph) $C \subseteq V$ is such that for two vertices $\{u, v\} \subseteq C$, $\{u, v\} \in E$. A maximum clique is a clique of maximum size.*

[3] This corresponds to requiring $T \subseteq V$ to be such that $\forall_{U \subseteq V} |U| < |T| \rightarrow \exists_{\{u,v\} \in E}, \{u, v\} \cap U = \emptyset$. Throughout the paper, we will skip the mathematical representation of minimum (but also maximum) size sets.

Reduction 3 (MaxClique $\leq_P$ HornMaxSAT)**.** *A MaxSAT encoding for MaxClique is the following. For any $\{u, v\} \in E^C$, add a hard clause $(\neg x_u \vee \neg x_v)$. For each $u \in V$, add a soft clause (x_u).*

Definition 4 (Minimum Dominating Set, MinDS)**.** *Let $G = (V, E)$ be an undirected graph. $D \subseteq V$ is a dominating set if any $v \in V \setminus D$ is adjacent to at least one vertex in D. A minimum dominating set is a dominating set of minimum size.*

Reduction 4 (MinDS $\leq_P$ HornMaxSAT)**.** *Let $x_u = 1$ iff $u \in V$ is excluded from a dominating set D. For each vertex $u \in V$ add a hard Horn clause $(\neg x_u \vee_{\{u,v\} \in E} \neg x_v)$. The soft clauses are (x_u), for $u \in V$.*

3.2 Optimization Problems on Sets

Definition 5 (Minimum Hitting Set, MinHS)**.** *Let $\mathcal{C}$ be a collection of sets of some set S. A hitting set $H \subseteq S$ is such that for any $D \in \mathcal{C}$, $H \cap D \neq \emptyset$. A minimum hitting set is a hitting set of minimum size.*

Reduction 5 (MinHS $\leq_P$ HornMaxSAT)**.** *For each $a \in S$ let $x_a = 1$ iff a is excluded from H. For each $D \in \mathcal{C}$, create a hard Horn clause $(\vee_{a \in D} \neg x_a)$. The soft clauses are (x_a), for $a \in S$.*

Remark 2. The minimum set cover (MinSC) is well-known to be equivalent to the minimum hitting set problem. Thus, the same reduction to HornMaxSAT can be applied.

Definition 6 (Maximum Set Packing, MaxSP)**.** *Let $\mathcal{T} = \{T_1, \ldots, T_k\}$ be a family of sets. $\mathcal{R} \subseteq \mathcal{T}$ is a set packing if $\forall_{T_i, T_j \in \mathcal{R}} T_i \cap T_j = \emptyset$. A maximum set packing is a set packing of maxim size.*

Reduction 6 (MaxSP $\leq_P$ HornMaxSAT)**.** *Let $x_i = 1$ iff T_i is included in the set packing. For each pair T_i, T_j, such that $T_i \cap T_j \neq \emptyset$, create a hard Horn clause $(\neg x_i \vee \neg x_j)$. The soft clauses are (x_i), for $T_i \in \mathcal{T}$.*

Remark 3. It is well-known that the maximum set packing problem can be reduced to the maximum clique problem. The reduction above exploits this result.

It also immediate to conclude that the weighted version of any of the optimization problems described in this and the previous sections can be reduced to HornWMaxSAT.

3.3 Handling Linear Constraints

This section argues that the propositional encodings of a number of linear constraints are Horn. In turn, this enables solving a number of optimization problems with HornMaxSAT.

The first observation is that *any* of the most widely used CNF encodings of AtMost$k$ constraints are composed *exclusively* of Horn clauses[4]:

Proposition 1 (CNF Encodings of AtMost$k$ constraints)**.** *The following CNF encodings of AtMost$k$ constraints are composed solely of Horn clauses: pairwise and bitwise encodings [8, Chapter 2], totalizers [6], sequential counters [35], sorting networks [16], cardinality networks [4, 5], pairwise cardinality networks [10], and modulo totalizers [29].*

Proof. Immediate by inspection of each encoding [4–6, 8, 10, 16, 29, 35]. □

[4] To our best knowledge, this property of propositional encodings has not been investigated before.

Algorithm 1: HMaxHS, a MaxHS-like [13] HornMaxSAT algorithm

Input: $\mathcal{F} = \langle \mathcal{A}, \mathcal{H} \rangle$, HornMaxSAT formula
Output: $(\mu, \texttt{Cost}(\mu))$, MaxSAT assignment and cost

```
1 begin
2     K ← ∅
3     while true do
4         S ← MinimumHS(K)
5         (st, μ, U) ← LTUR(H ∪ (A \ S))
          // If st, then μ is a satisfying assignment
          // Otherwise, U is a core/MUS
6         if st then return (μ, Cost(μ))
7         K ← K ∪ {U}
8 end
```

For the case of the more general pseudo-Boolean constraints, $\sum a_i x_i \leq b$, with a_i, b non-negative, there also exist Horn encodings:

Proposition 2 (CNF Encodings of Pseudo-Boolean Constraints)**.** *The Local Polynomial Watchdog [7] encoding for PB constraints is composed solely of Horn clauses.*

Proof. Immediate by inspection of the encoding in [7]. □

These observations have immediate impact on the range of problems that can be solved with HornMaxSAT and HornWMaxSAT. One concrete example is the Knapsack problem [12].

Definition 7 (Knapsack problem)**.** *Let $\{1, \ldots, n\}$ denote a set of n objects, each with value v_i and weight w_i, $1 \leq i \leq n$, and a maximum weight value W. The knapsack problem is to pick a subset of objects of maximum value that is consistent with the weight constraint. By letting $x_i = 1$ iff object i is picked, we get the well-known 0-1 ILP formulation max $\sum_i v_i x_i$; s.t. $\sum_i w_i x_i \leq W$.*

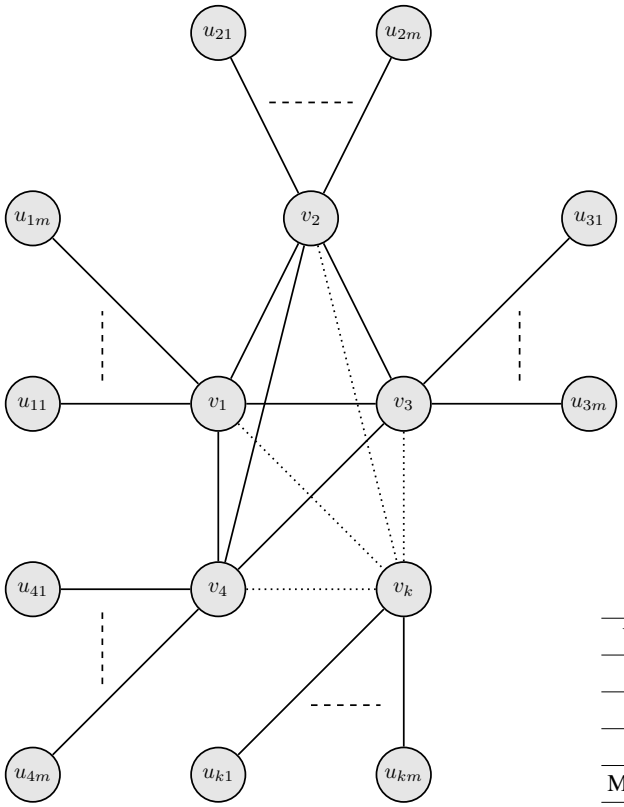
Reduction 7 (Knapsack $\leq_P$ HornMaxSAT)**.** *From Proposition 2, there exist Horn encodings for Pseudo-Boolean constraints. Thus, the hard constraint $\sum_i w_i x_i \leq W$ can be encoded with Horn clauses. The soft clauses are (x_i) for each object i, each with cost v_i. Both the soft and the hard clauses in the reduction are Horn.*

4 HornMaxSAT Algorithm with Hitting Sets

This section develops a MaxHS-like [13, 33] algorithm for HornMaxSAT. In addition, the section shows that this MaxHS-like algorithm elicits the possibility of solving large scale problems with abstraction.

4.1 A MaxHS-Like HornMaxSAT Algorithm

With the goal of exploiting the special structure of HornMaxSAT, a MaxHS-like algorithm is envisioned [13, 33]. Algorithm 1 summarizes the proposed approach. The key observation is that each call to LTUR [27] runs in linear time. (Unit propagation as implemented in modern SAT solvers, will also run in polynomial time, but it will be less efficient in practice.) The original motivation for MaxHS is that finding a minimum hitting set of $\mathcal{S}$ is expected to be much easier than solving the MaxSAT problem. This is



(a) Example graph

Vertices	$(1+m)k$
Edges	$k(k-1)/2+mk$
MaxIS	mk
MinVC	k
MaxClique	k

(b) Graph stats

Fig. 1: Example graph for computing MaxIS & MinVC

also the motivation for HMaxHS. As observed in recent work [3, 26], MUSes (minimal unsatisfiable subsets) can be computed in polynomial time in the case of Horn formulas. MUS extraction, but also MCS (minimal correction subset) extraction [26], are based on the original LTUR algorithm [27]. It should be noted that some implementations of MaxHS use an ILP (Integer Linear Programming) package (e.g. CPLEX or SCIP) [13, 33][5], whereas others exploit SAT solvers for computing minimum hitting sets [19, 21].

4.2 Automatic Abstraction-Based Problem Solving

For some of the problems described in Section 3 a possible criticism of Algorithm 1 is that it will iteratively find sets $\mathcal{U}$ consisting of a single clause, and it will essentially add to $\mathcal{K}$ all the clauses in $\mathcal{H}$. Although this is in fact a possibility for some problems (but not all, as investigated in Section 5), this section shows that even for these problems, Algorithm 1 can provide an effective problem solving approach.

Consider the example graph in Figure 1, where the goal is to compute a maximum independent set (or alternatively a minimum vertex cover). From the figure, we can

[5] SCIP and CPLEX are available, respectively, from `http://scip.zib.de/` and `https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/`.

conclude that the number of vertices is $(1+m)k$, the number of edges is $(k(k-1)/2+km)$, the size of the maximum independent set is km and the size of the minimum vertex cover is k. From the inspection of the reduction from MaxIS (or MinVC) to HornMaxSAT, and the operation of Algorithm 1, one might anticipate that Algorithm 1 would iteratively declare each hard clause as an unsatisfiable core, and replicate the clause in the list $\mathcal{K}$ of sets to hit, thus requiring a number of iterations no smaller than the number of edges. (More importantly, for a MaxHS-like algorithm, the number of iterations is worst-case exponential [13].) However, and as shown below, the operation of the HMaxHS actually *ensures* this is *not* the case.

Without loss of generality, consider any of the vertices in the clique, i.e. $v_1, \ldots, v_k$, say v_i. For this vertex, no more than $k(k-1)/2 + 2k$ edges will be replicated, i.e. added to $\mathcal{K}$. Observe that, as soon as two edges $\{v_i, u_{ij_1}\}$ and $\{v_i, u_{ij_2}\}$ are replicated, a minimum hitting set will necessarily pick v_i. As a result, after at most $k(k-1)/2+2k$ iterations, the algorithm will terminate with the answer mk. Essentially, the algorithm is capable of *abstracting* away $(m-2)k$ clauses when computing the maximum independent set. Observe that m can be made arbitrarily large. Abstraction is a well-known topic in AI, with important applications [17]. The example in this section suggests that HornMaxSAT and the HMaxHS algorithm can effectively enable automatic abstraction for solving large scale (graph) optimization problems. This remark is further investigated in Section 6.

It should be noted that the result above highlights what seems to be a fundamental property of the original MaxHS algorithm [13]. Although in the worst case, the algorithm can require an exponential number of steps to find the required set of clauses to remove to achieve consistency, the result above illustrates how the MaxHS can be effective at discarding irrelevant clauses, and focusing on the key parts of the formula, thus being able to compute solutions in a number of iterations not much larger than the minimum number of falsified clauses in the MaxHS solution. Results from recent MaxSAT Evaluations[6] confirm the practical effectiveness of MaxHS-like algorithms.

5 HornMaxSAT in Practice

Besides the reference optimization problems analyzed in Section 3, a number of practical applications can also be shown to correspond to solving HornMaxSAT or can be reduced to HornMaxSAT. This section investigates some of these problems, but also proposes generic HornMaxSAT encodings for SAT and CSP.

5.1 Sample Problems

Different optimization problems in practical settings are encoded as HornMaxSAT. The winner determination problem (WDP) finds important applications in combinatorial auctions. An immediate observation is that the encoding proposed in [18] corresponds to HornMaxSAT. The problem of coalition structure generation (CSG) also finds important applications in multi-agent systems. An immediate observation is that some of the encodings proposed in [24] correspond to HornMaxSAT. HornMaxSAT also finds application in the area of axiom pinpointing for $\mathcal{EL}^+$ description logic, but also for other lightweight description logics. For the concrete case of $\mathcal{EL}^+$, the problem

[6] http://www.maxsat.udl.cat/.

encoding is well-known to be Horn [34], with the soft clauses being unit positive. The use of LTUR-like algorithms has been investigated in [3].

As shown in the sections below, it is actually simple to reduce different decision (and optimization[7]) problems into HornMaxSAT.

5.2 Reducing SAT to HornMaxSAT

Let $\mathcal{F}$ be a CNF formula, with N variables $\{x_1 \ldots, x_N\}$ and M clauses $\{c_1, \ldots, c_M\}$. Given $\mathcal{F}$, the reduction creates a Horn MaxSAT problem with hard clauses $\mathcal{H}$ and soft clauses $\mathcal{S}$, $\langle \mathcal{H}, \mathcal{S} \rangle = \mathsf{HEnc}(\mathcal{F})$. For each variable $x_i \in X$, create new variables p_i and n_i, where $p_i = 1$ iff $x_i = 1$, and $n_i = 1$ iff $x_i = 0$. Thus, we need a hard clause $(\neg p_i \lor \neg n_i)$, to ensure that we do not simultaneously assign $x_i = 1$ and $x_i = 0$. (Observe that the added clause is Horn.) For each clause c_j we require c_j to be satisfied, by requiring that one of its literals *not* to be falsified. For each literal x_i use $\neg n_i$ and for each literal $\neg x_i$ use $\neg p_i$. Thus, c_j is encoded with a new (hard) clause c'_j with the same number of literals as c_i, but with only negative literals on the p_i and n_i variables, and so the resulting clause is also Horn. The set of soft clauses $\mathcal{S}$ is given by (p_i) and (n_i) for each of the original variables x_i. If the resulting Horn formula has a HornMaxSAT solution with at least N variables assigned value 1, then the original formula is satisfiable; otherwise the original formula is unsatisfiable. (Observe that, by construction, the HornMaxSAT solution cannot assign value 1 to more than N variables.) Clearly, the encoding outlined in this section can be subject to different improvements, e.g. not all clauses need to be goal clauses.

The transformation proposed can be related with the well-known dual-rail encoding, used in different settings [9,22,25,30,31]. To our best knowledge, the use of a dual-rail encoding for deriving a pure Horn formula has not been proposed in earlier work.

5.3 Reducing CSP to HornMaxSAT

This section investigates reductions of Constraint Satisfaction Problems (CSP) into HornMaxSAT. Standard definitions are assumed [32]. A CSP is a triple $\langle X, D, C \rangle$, where $X = \langle x_1, \ldots, x_N \rangle$ is an n-tuple of variables, D is a corresponding N-tuple of domains $D = \langle D_1, \ldots, D_N \rangle$, such that $x_i \in D_i$, and C is a t tuple of constraints $C = \langle C_1, \ldots, C_t \rangle$. C_j is a pair $\langle R_{S_j}, S_j \rangle$, where R_{S_j} is a relation on the variables in S_j, and represents a subset of the Cartesian product of the domains of the variables in S_j.

One approach to encode CSPs as HornMaxSAT is to translate the CSP to SAT (e.g. [36]), and then apply the Horn encoder outlined in Section 5.2. There are however, alternative approaches, one of which we now detail. We show how to adapt the well-known direct encoding of CSP into SAT [36]. The set of variables is x_{iv}, such that $x_{iv} = 1$ iff x_i is assigned value $v \in D_i$. Moreover, we consider the *disallowed* combinations of values of each constraint C_j. For example, if the combination of values $x_{i_1} = v_{i_1} \land x_{i_2} = v_{i_2} \land \cdots \land x_{i_q} = v_{i_q}$ is disallowed, i.e. no tuple of the relation S_j associated with C_j contains these values, then add a (Horn) clause $(\neg x_{i_1 v_{i_1}} \lor \cdots \lor \neg x_{i_q v_{i_q}})$. For each x_i, require that no more than one value can be used: $\sum_{v \in D_i} x_{iv} \leq 1$; this AtMost1 constraint can be encoded with Horn clauses as shown in Proposition 1. Finally, the

[7] Due to lack of space, details are omitted.

goal is to assign as many variables as possible, and so add a soft clause $(x_{i,v})$ for each x_i and each $v \in D_i$. It is immediate that the CSP is satisfiable iff the HornMaxSAT formulation has a solution with at least N satisfied soft clauses (and by construction it cannot assign value 1 to more than N variables).

5.4 Reducing PHP to HornMaxSAT

The previous sections show that the optimization and decision problems with simple reductions to HornMaxSAT are essentially endless, as any decision problem that can be reduced to SAT or CSP can also be reduced to HornMaxSAT. However, it is also possible to develop specific reductions, that exploit the original problem formulation. This section investigates how to encode the representation of the pigeonhole principle (PHP) as HornMaxSAT, for which propositional encodings are well-known and extensively investigated [11].

Definition 8 (Pigeonhole Principle, PHP [11])**.** *The pigeonhole principle states that if $m+1$ pigeons are distributed by m holes, then at least one hole contains more than one pigeon. A more formal formulation is that there exists no injective function mapping $\{1, 2, ..., m+1\}$ to $\{1, 2, ..., m\}$, for $m \geq 1$.*

Propositional formulations of PHP encode the negation of the principle, and ask for an assignment such that the $m+1$ pigeons are placed into m holes [11]. Given a propositional encoding and the reduction proposed in Section 5.2, we can encode PHP formulas into HornMaxSAT. We describe below an alternative reduction.

Reduction 8 (PHP $\leq_P$ HornMaxSAT)**.** *Let $x_{ij} = 1$ iff pigeon i, with $1 \leq i \leq m+1$, is placed in hole j, with $1 \leq j \leq m$. For each hole j, $1 \leq j \leq m$, at most 1 pigeon can be placed in hole j:*

$$\sum_{i=1}^{m+1} x_{ij} \leq 1 \qquad 1 \leq j \leq m \tag{1}$$

which can be encoded with Horn clauses, by Proposition 1.
For each pigeon i, $1 \leq i \leq m+1$, the pigeon is placed in at most 1 hole:

$$\sum_{j=1}^{m} x_{ij} \leq 1 \qquad 1 \leq i \leq m+1 \tag{2}$$

which can also be encoded with Horn clauses, by Proposition 1.
The soft clauses are (x_{ij}), with $1 \leq i \leq m+1, 1 \leq j \leq m$. The PHP problem is satisfiable iff the HornMaxSAT problem has a solution satisfying at least $m+1$ soft clauses, i.e. $m+1$ are placed.

6 Experimental Results

This section provides a preliminary investigation into exploiting reductions to HornMaxSAT in practice. All the experiments were run in Ubuntu Linux on an Intel Xeon E5-2630 2.60GHz processor with 64GByte of memory. The time limit was set to 1800s and the memory limit to 10GByte for each process to run. Two classes of problem instances were considered. The first being a set of 46 PHP instances that were generated ranging the number of holes from 10 up to 100. The second set of benchmarks corresponds to 100 instances generated according to the example in Figure 1, with k ranging from 10 to 100 and m ranging from k to $20k$. In the experiments six different MaxSAT solvers were considered. Some solvers are core-guide [28] (namely, OpenWBO16, WPM3,

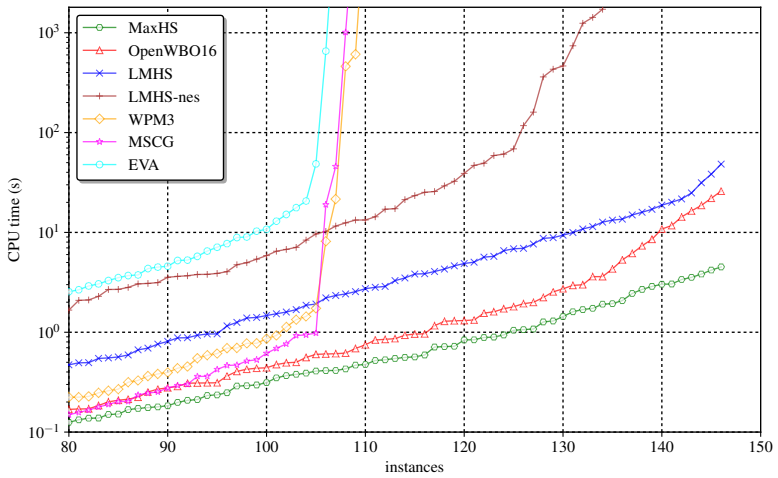


Fig. 2: Cactus plot for selected solvers on PHP and MaxIS benchmarks.

Table 1: Statistics on benchmarks generated according to the example in Figure 1.

k	10		20		30		40		50		60		70		80		90	
m	100	200	200	400	300	600	400	800	500	1000	600	1200	700	1400	800	1600	900	1800
UB	65	65	230	230	495	495	860	860	1325	1325	1890	1890	2555	2555	3320	3320	4185	4185
#DC	9	7	13	13	27	26	25	25	50	50	49	36	70	70	48	49	63	63
#I	19	35	71	132	53	72	211	356	50	50	225	693	70	70	2140	768	747	812

MSCG and Eva), whereas others are based on implicit hitting sets (namely, MaxHS and LMHS) [28]. Additionally, a variant of LMHS was considered for which the option "–no-equiv-seed" was set (LMHS-nes). The results are summarized in the cactus plot shown in Figure 2. As can be observed, solvers based on implicit hitting sets (i.e. the MaxHS family of MaxSAT algorithms), but also OpenWBO16, perform very well on the instances considered[8]. The differences to the other solvers are solely due to the PHP instances. While propositional encodings of PHP are well-known to be extremely hard for SAT solvers, the proposed MaxSAT encoding scales well for MaxHS-like algorithms, but also for the core-guided MaxSAT solver OpenWBO16.

Analysis of the number of iterations. In order to validate the abstraction mechanism described in Section 4.2, we considered the LMHS-nes variant, and the benchmarks generated according to the example in Figure 1. The reason to consider LMHS-nes is that soft clauses are all unit and the set of soft clauses includes the complete set of variables of the formula. If the option is not set, then the complete CNF formula is replicated inside the MIP solver (CPLEX), *as a preprocessing step*, which results in exactly one call to CPLEX [14].

Table 1 presents the results obtained, where first and second row show the k and the m parameters of the instance. The third row (UB) shows the upper bound on the number of iterations presented in Section 4.2. The fourth and fifth rows show the number of

[8] Any implementation of the MaxHS-family of MaxSAT algorithms, by using a CDCL SAT solver, implements a basic version of the algorithm proposed in Section 4.

disjoint cores (#DC) and the number of iterations (#I) reported by LMHS-nes. As can be concluded from the table, the number of iterations is always smaller than the upper bound, suggesting that the algorithm is able to abstract clauses more effectively than in the worst case scenario. The ability of HMaxHS algorithms to find good abstractions is expected to represent a significant step into deploying HornMaxSAT problem solvers.

7 Conclusions & Research Directions

The practical success of recent MaxSAT solvers not only motivates investigating novel applications, but it also justifies considering subclasses of the general MaxSAT problem. This paper investigates the subclass of MaxSAT restricted to Horn clauses, i.e. HornMaxSAT. The paper shows that a comprehensive set of optimization and decision problems are either formulated as HornMaxSAT or admit simple reductions to HornMaxSAT. The paper also shows that fundamental decision problems, including SAT and CSP, can be reduced to HornMaxSAT. The role of HornMaxSAT in tackling the limits of resolution is investigated in recent work [20]. Although NP-hardness of HornMaxSAT guarantees that such reductions must exist, the paper develops simple reductions, some of which were unknown to our best knowledge. The paper also proposes a HornMaxSAT algorithm, based on a well-known family of MaxSAT algorithms [13, 33], but which exploits the fact that the formulas to be analyzed are Horn. The experimental results show the promise of reductions of HornMaxSAT and motivate investigating further the use of HornMaxSAT as a generic problem solving approach. This also motivates the development of more efficient implementations of HMaxHS and of alternative approaches to HMaxHS.

References

1. I. Abío, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. BDDs for pseudo-Boolean constraints - revisited. In *SAT*, pages 61–75, 2011.
2. C. Ansótegui, M. L. Bonet, and J. Levy. SAT-based MaxSAT algorithms. *Artif. Intell.*, 196:77–105, 2013.
3. M. F. Arif, C. Mencía, and J. Marques-Silva. Efficient MUS enumeration of Horn formulae with applications to axiom pinpointing. In *SAT*, pages 324–342, 2015.
4. R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. Cardinality networks and their applications. In *SAT*, pages 167–180, 2009.
5. R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. Cardinality networks: a theoretical and empirical study. *Constraints*, 16(2):195–221, 2011.
6. O. Bailleux and Y. Boufkhad. Efficient CNF encoding of Boolean cardinality constraints. In *CP*, pages 108–122, 2003.
7. O. Bailleux, Y. Boufkhad, and O. Roussel. New encodings of pseudo-Boolean constraints into CNF. In *SAT*, pages 181–194, 2009.
8. A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.
9. R. E. Bryant, D. L. Beatty, K. S. Brace, K. Cho, and T. J. Sheffler. COSMOS: A compiled simulator for MOS circuits. In *DAC*, pages 9–16, 1987.
10. M. Codish and M. Zazon-Ivry. Pairwise cardinality networks. In *LPAR*, pages 154–172, 2010.
11. S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *J. Symb. Log.*, 44(1):36–50, 1979.

12. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.
13. J. Davies and F. Bacchus. Solving MAXSAT by solving a sequence of simpler SAT instances. In *CP*, pages 225–239, 2011.
14. J. Davies and F. Bacchus. Exploiting the power of mip solvers in maxsat. In *SAT*, pages 166–181, 2013.
15. W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *J. Log. Program.*, 1(3):267–284, 1984.
16. N. Eén and N. Sörensson. Translating pseudo-Boolean constraints into SAT. *JSAT*, 2(1-4):1–26, 2006.
17. F. Giunchiglia and T. Walsh. A theory of abstraction. *Artif. Intell.*, 57(2-3):323–389, 1992.
18. F. Heras, J. Larrosa, S. de Givry, and T. Schiex. 2006 and 2007 max-sat evaluations: Contributed instances. *JSAT*, 4(2-4):239–250, 2008.
19. A. Ignatiev, A. Morgado, and J. Marques-Silva. Propositional abduction with implicit hitting sets. In *ECAI*, pages 1327–1335, 2016.
20. A. Ignatiev, A. Morgado, and J. Marques-Silva. On tackling the limits of resolution in SAT solving. In *SAT*, 2017.
21. A. Ignatiev, A. Previti, M. H. Liffiton, and J. Marques-Silva. Smallest MUS extraction with minimal hitting set dualization. In *CP*, pages 173–182, 2015.
22. S. Jabbour, J. Marques-Silva, L. Sais, and Y. Salhi. Enumerating prime implicants of propositional formulae in conjunctive normal form. In *JELIA*, pages 152–165, 2014.
23. B. Jaumard and B. Simeone. On the complexity of the maximum satisfiability problem for Horn formulas. *Inf. Process. Lett.*, 26(1):1–4, 1987.
24. X. Liao, M. Koshimura, H. Fujita, and R. Hasegawa. Solving the coalition structure generation problem with MaxSAT. In *ICTAI*, pages 910–915, 2012.
25. V. M. Manquinho, P. F. Flores, J. Marques-Silva, and A. L. Oliveira. Prime implicant computation using satisfiability algorithms. In *ICTAI*, pages 232–239, 1997.
26. J. Marques-Silva, A. Ignatiev, C. Mencía, and R. Peñaloza. Efficient reasoning for inconsistent Horn formulae. In *JELIA*, pages 336–352, 2016.
27. M. Minoux. LTUR: A simplified linear-time unit resolution algorithm for Horn formulae and computer implementation. *Inf. Process. Lett.*, 29(1):1–12, 1988.
28. A. Morgado, F. Heras, M. H. Liffiton, J. Planes, and J. Marques-Silva. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints*, 18(4):478–534, 2013.
29. T. Ogawa, Y. Liu, R. Hasegawa, M. Koshimura, and H. Fujita. Modulo based CNF encoding of cardinality constraints and its application to MaxSAT solvers. In *ICTAI*, pages 9–17, 2013.
30. A. Previti, A. Ignatiev, A. Morgado, and J. Marques-Silva. Prime compilation of non-clausal formulae. In *IJCAI*, pages 1980–1988, 2015.
31. J. Roorda and K. Claessen. A new SAT-based algorithm for symbolic trajectory evaluation. In *CHARME*, pages 238–253. Springer, 2005.
32. F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*. Elsevier, 2006.
33. P. Saikko, J. Berg, and M. Järvisalo. LMHS: A SAT-IP hybrid MaxSAT solver. In *SAT*, pages 539–546, 2016.
34. R. Sebastiani and M. Vescovi. Axiom pinpointing in lightweight description logics via Horn-SAT encoding and conflict analysis. In *CADE*, pages 84–99, 2009.
35. C. Sinz. Towards an optimal CNF encoding of Boolean cardinality constraints. In *CP*, pages 827–831, 2005.
36. T. Walsh. SAT v CSP. In *CP*, pages 441–456, 2000.
37. J. P. Warners. A linear-time transformation of linear inequalities into conjunctive normal form. *Inf. Process. Lett.*, 68(2):63–69, 1998.