Maximal Falsifiability: Definitions, Algorithms, and Applications

Alexey Ignatiev[1,4], Antonio Morgado[1], Jordi Planes[3], and Joao Marques-Silva[1,2]

[1] IST/INESC-ID, Lisbon, Portugal
[2] University College Dublin, Ireland
[3] Universitat de Lleida, Spain
[4] ISDCT SB RAS, Irkutsk, Russia
{aign,ajrm}@sat.inesc-id.pt, jplanes@diei.udl.cat, jpms@ucd.ie

Abstract. Similarly to Maximum Satisfiability (MaxSAT), Minimum Satisfiability (MinSAT) is an optimization extension of the Boolean Satisfiability (SAT) decision problem. In recent years, both problems have been studied in terms of exact and approximation algorithms. In addition, the MaxSAT problem has been characterized in terms of Maximal Satisfiable Subsets (MSSes) and Minimal Correction Subsets (MCSes), as well as Minimal Unsatisfiable Subsets (MUSes) and minimal hitting set dualization. However, and in contrast with MaxSAT, no such characterizations exist for MinSAT. This paper addresses this issue by casting the MinSAT problem in a more general framework. The paper studies *Maximal Falsifiability*, the problem of computing a subset-maximal set of clauses that can be simultaneously falsified, and shows that MinSAT corresponds to the complement of a largest subset-maximal set of simultaneously falsifiable clauses, i.e. the solution of the *Maximum Falsifiability* (MaxFalse) problem. Additional contributions of the paper include novel algorithms for Maximum and Maximal Falsifiability, as well as minimal hitting set dualization results for the MaxFalse problem. Moreover, the proposed algorithms are validated on practical instances.

1 Introduction

Maximum and Minimum Satisfiability (resp. MaxSAT and MinSAT) are two well-known optimization extensions of Boolean Satisfiability (SAT) (e.g. [30,34,39]). While the goal of MaxSAT is to compute an assignment that *maximizes* the number of satisfied clauses, the goal of MinSAT is to compute an assignment that *minimizes* the number of satisfied clauses. Besides the plain versions, where all clauses are *soft* and so relaxable, both MaxSAT and MinSAT admit weighted versions as well as the existence of hard clauses, i.e. clauses that *must* be satisfied. MinSAT has been studied since the mid 90s [8,9,27,37], with the original focus being on the computational complexity of the problem and on approximation algorithms. In recent years there has been a renewed interest in MinSAT, with the focus being on branch-and-bound and iterative algorithms, but also on encodings of MinSAT to MaxSAT [5,6,21,28,31,33,34,45].

Like MaxSAT, MinSAT finds a growing number of practical applications (e.g. [13, 14, 16, 19, 23, 26]), and it has also been used in complexity characterizations of other problems (e.g. [2, 15, 18, 20]). More importantly, given a MaxSAT problem where the soft clauses are all unit, complementing the soft clauses gives a MinSAT problem. As shown in recent work (e.g. [6, 34] among others), the resulting optimization problems can be fairly different, and so reducing MaxSAT to MinSAT can in some settings produce problem instances that are easier to solve. As a result, one can expect the integration of MinSAT algorithms in portfolios of MaxSAT algorithms in the near future.

MaxSAT has been extensively studied in the context of reasoning about inconsistent sets of constraints. It is well-known that each MaxSAT solution represents a largest Maximal Satisfiable Subset (MSS) [12, 36]. The complement of an MSS is a Minimal Correction Subset (MCS), i.e. a subset-minimal relaxation of a formula that renders the formula satisfiable. Moreover, another well-known result is that each MCS is a minimal hitting set of the Minimal Unsatisfiable Subsets (MUSes), and each MUS is a minimal hitting set of the MCSes [10, 12, 36, 43]. In contrast, and despite the vast body of work on MinSAT, similar results for the case of MinSAT are non-existent.

This paper addresses this issue and conducts a more comprehensive characterization of the MinSAT problem. The main contributions of the paper can be summarized as follows. First, the paper introduces *Maximal Falsifiability*, which represents the problem of computing subset-maximal sets of simultaneously falsifiable clauses. Second, the paper addresses MinSAT from the perspective of the largest maximal falsifiable solution based on the connection between MinSAT solutions and the so-called *Maximum Falsifiability* (MaxFalse) solutions. Third, the paper develops algorithms for Maximal and Maximum Falsifiability, thereby indirectly developing novel algorithms for the MinSAT problem. Moreover, and for the case of plain maximal falsifiability, the paper shows that it can be reduced to the maximal independent set problem. Thus, well-known linear time algorithms for maximal independent set [25] can be used for computing a single maximal falsifiability solution. Similarly, algorithms for the enumeration of maximal independent set [24, 29] can be used for enumerating maximal falsifiability solutions. In addition, the paper also shows that a minimal hitting set relationship, which for the case of maximal satisfiability relates MCSes and MUSes [10, 12, 36, 43], also exists for the case of maximal falsifiability. Thus, enumeration problems related with Maximal Falsifiability can be tackled by hitting set dualization, similarly to what has been done in the context of maximal satisfiability [10, 36]. Finally, the paper presents some preliminary results on both Maximal and Maximum Falsifiability algorithms.

The paper is organized as follows. Section 2 introduces the basic definitions and notation used throughout the paper. Section 3 introduces the Maximal and Maximum Falsifiability problems as well as related computational problems. Section 4 develops algorithms for Maximal Falsifiability, whereas Section 5 develops algorithms for Maximum Falsifiability (and so for MinSAT). Theoretical results for enumeration problems and minimal hitting sets are presented in Section 6. Section 7 provides experimental results for Maximal and Maximum Falsifiability, and Section 8 concludes the paper. Appendix A presents a list of acronyms used in the paper followed by Appendix B containing pseudo-codes of some of the algorithms proposed in the paper.

2 Preliminaries

This section briefly introduces the definitions used throughout. Additional standard definitions can be found elsewhere (e.g. [11]). Boolean formulas are represented in calligraphic font, $\mathcal{F}, \mathcal{M}, \mathcal{S}, \mathcal{T}, \mathcal{U}, \mathcal{W}, \mathcal{F}', \ldots$ A Boolean formula in conjunctive normal form (CNF) is defined as a finite set of finite sets of literals. Where appropriate, a CNF formula will also be understood as a conjunction of disjunctions of literals, where each disjunction represents a *clause* and a literal is a variable or its complement. The variables of formula $\mathcal{F}$ are denoted by $\mathtt{var}(\mathcal{F})$. Variables are represented by $X = \{x, y, z, x_1, y_1, z_1, \ldots\}$ and literals by $\{l, l_1, l_2, \ldots\}$. The clauses of a formula are represented by $\{c, c_1, c_2, \ldots\}$. A literal l is called *pure* in formula $\mathcal{F}$ if there is a clause in formula $\mathcal{F}$ containing l but no clause in $\mathcal{F}$ that contains a complementary literal $\neg l$. An assignment is a map $\mathcal{A} : \mathtt{var}(\mathcal{F}) \mapsto \{0, 1\}$. A clause is satisfied by an assignment if one of its literals is assigned value 1. A model of $\mathcal{F}$ is an assignment that satisfies all clauses in $\mathcal{F}$.

The standard definitions of MaxSAT are assumed (e.g. [30]). Moreover, the following definitions also apply. Given a CNF formula $\mathcal{F}$, sets of clauses $\mathcal{S}$, $\mathcal{S} \subseteq \mathcal{F}$, and $\mathcal{C}$, $\mathcal{C} = \mathcal{F} \setminus \mathcal{S}$, are called a *Maximal Satisfiable Subset* (MSS) and a *Minimal Correction Subset* (MCS), respectively, if $\mathcal{S}$ is satisfiable and $\forall_{c \in \mathcal{C}}$ set $\mathcal{S} \cup \{c\}$ is unsatisfiable. A set of clauses $\mathcal{U}$, $\mathcal{U} \subseteq \mathcal{F}$, is called a *Minimal Unsatisfiable Subset* (MUS) if $\mathcal{U}$ is unsatisfiable and $\forall_{c \in \mathcal{U}}$ set $\mathcal{U} \setminus \{c\}$ is satisfiable. The reader is referred to [12, 36] for further details. In the context of MaxSAT and MinSAT, a formula $\mathcal{F}$ is viewed as a 2-tuple $(\mathcal{H}, \mathcal{R})$, where $\mathcal{H}$ denotes the *hard* clauses, which must be satisfied, and $\mathcal{R}$ denotes the *soft* (or *relaxable*) clauses. A weight can be associated with each clause, such that hard clauses have a special weight $\top$. Hence, the weight function is a map $w : \mathcal{H} \cup \mathcal{R} \to \{\top\} \cup \mathbb{N}$, such that $\forall_{c \in \mathcal{H}} w(c) = \top$ and $\sum_{c \in \mathcal{R}} w(c) < \top$. If no weight function is specified, it is assumed that $\forall_{c \in \mathcal{R}} w(c) = 1$.

The paper also considers a number of optimization problems in graphs. Given an undirected graph $\mathcal{G} = (V, E)$, an *Independent Set* (IS) is a set $I \subseteq V$ such that $\forall_{u,v \in I}, (u, v) \notin E$. A *vertex cover* is a set $C \subseteq V$ such that $\forall_{(u,v) \in E}, u \in C \vee v \in C$. Finally, a *clique* (or complete subgraph) is a set $L \subseteq V$ such that $\forall_{u,v \in L}, u \neq v \Rightarrow (u, v) \in E$. Given an independent set $I \subseteq V$, a well-known result is that $V \setminus I$ is a vertex cover of $\mathcal{G}$ and I is a clique of $\mathcal{G}^C$, the complemented graph. The *Maximum Independent Set* (MIS) problem consists in computing an IS of largest size. The *Maximal Independent Set* (MxIS) problem consists in computing a subset-maximal IS. Both problems can be generalized to the case when a weight is associated with each vertex. More importantly, given the above relationships between ISes, VCes and cliques, solutions of the MIS and MxIS problems also represent, respectively, solutions for the *Minimum Vertex Cover* (MVC) and a *Minimal Vertex Cover* (MnVC) of graph of a graph $\mathcal{G}$, as well as a *Maximum Clique* (MaxClique) and a *Maximal Clique* (MxClique) of the complemented graph $\mathcal{G}^C$. A well-known result is that a maximal independent set can be computed in linear time [25]. The topic of enumeration of maximal independent sets has also been extensively studied (e.g. [1, 24, 29, 44]).

3 Maximal and Maximum Falsifiability

This section starts by introducing the plain maximal and maximum falsifiability problems. In this case, $\mathcal{H} = \emptyset$ and so $\mathcal{F} = \mathcal{R}$, i.e. all clauses are soft (and so relaxable) and their cost is 1. Generalizations of the basic problems are considered later in this section.

Definition 1 (All-Falsifiable). *A set of clauses $\mathcal{U}$ is* All-Falsifiable *if there exists a truth assignment $\mathcal{A}$ such that $\mathcal{A}$ falsifies* all *clauses in $\mathcal{U}$.*

Proposition 1. *A set of clauses $\mathcal{U}$ is all-falsifiable iff all the literals of $\mathcal{U}$ are pure.*

Proof. Let $\mathcal{U}$ be all-falsifiable. Assume, that not all the literals of $\mathcal{U}$ are pure. This means that there exist a literal l and clauses c_i and c_j in $\mathcal{U}$ such that $l \in c_i$ and $\neg l \in c_j$. But every complete truth assignment $\mathcal{A}$ satisfies at least one of these clauses, because literals l and $\neg l$ cannot be falsified simultaneously. Hence, our initial assumption — that not all the literals of $\mathcal{U}$ are pure — must be false.

Let all the literals of $\mathcal{U}$ be pure. And let us choose a complete assignment $\mathcal{A}$ in the following way: $\mathcal{A}(\texttt{var}(l)) = \neg l, \forall_{l \in \mathcal{U}}$. Then assignment $\mathcal{A}$ falsifies all clauses of $\mathcal{U}$, i. e. $\mathcal{U}$ is all-falsifiable. □

Definition 2 (MFS). *Given a formula $\mathcal{F}$, a* Maximal Falsifiable Subset *(MFS) of $\mathcal{F}$ is a subset $\mathcal{M} \subseteq \mathcal{F}$ such that:*

1. *$\mathcal{M}$ is all-falsifiable.*
2. *For any subformula $\mathcal{P}$, $\mathcal{F} \supseteq \mathcal{P} \supsetneq \mathcal{M}$, $\mathcal{P}$ is not all-falsifiable.*

Definition 3 (Maximum Falsifiability). *Given a formula $\mathcal{F}$,* Maximum Falsifiability *(MaxFalse) denotes the problem of computing the largest (in terms of the number of clauses) MFS of $\mathcal{F}$.*

Definition 4 (Minimum Satisfiability). *Given a formula $\mathcal{F}$,* Minimum Satisfiability *(MinSAT) is the problem of computing the smallest number of simultaneously satisfied clauses of $\mathcal{F}$ (while the other clauses of $\mathcal{F}$ are falsified).*

Proposition 2. *$\mathcal{M}$ represents a MaxFalse solution iff $\mathcal{F} \setminus \mathcal{M}$ represents a MinSAT solution.*

Notice that the proof of Proposition 2 is quite trivial and, thus, is omitted here. Nevertheless, Proposition 2 indicates that, in addition to recent algorithms for MinSAT [5, 28, 31, 33, 34], possible alternatives include dedicated algorithms for the MaxFalse problem, and also solutions based on the enumeration of MFSes.

Besides MFSes, additional minimal sets are of interest. One example is a minimal set of clauses which, if removed from $\mathcal{F}$, yield an all-falsifiable set of clauses.

Definition 5 (MCFS). *Given a formula $\mathcal{F}$, a* Minimal Correction (for Falsifiability) Subset *(MCFS) is a set $\mathcal{C} \subseteq \mathcal{F}$ such that:*

1. *$\mathcal{F} \setminus \mathcal{C}$ is all-falsifiable.*
2. *$\forall_{c \in \mathcal{C}}$, $\mathcal{F} \setminus (\mathcal{C} \setminus \{c\})$ is* not *all-falsifiable.*

Definition 6 (MNFS). *Given a formula $\mathcal{F}$, a* Minimal Non-Falsifiable Subset *(MNFS) is a set $\mathcal{N} \subseteq \mathcal{F}$ such that:*

1. *$\mathcal{N}$ is* not *all-falsifiable.*
2. *$\forall_{c \in \mathcal{N}}$, $\mathcal{N} \setminus \{c\}$ is all-falsifiable.*

Example 1. Consider the following formula:

$$F \triangleq \overset{c_1}{(x_1)} \wedge \overset{c_2}{(\bar{x}_1)} \wedge \overset{c_3}{(\bar{x}_1 \vee x_2)} \wedge \overset{c_4}{(\bar{x}_2)} \wedge \overset{c_5}{(x_3)}$$

The sets $\{c_2, c_3, c_5\}$, $\{c_1, c_4\}$, and $\{c_1, c_2\}$ denote, respectively, examples of an MFS, an MCFS and an MNFS.

A relevant result is the relationship between plain maximal falsifiability and maximal independent sets. Given $\mathcal{F}$, let $\mathcal{G} = (V, E)$ be an undirected graph such each clause of $\mathcal{F}$ is represented by a vertex of $\mathcal{G}$. Moreover, there exists an edge between two vertices iff the corresponding clauses have complemented literals. Clearly (see Proposition 1) clauses with complemented literals *cannot* be simultaneously falsified. Hence, an MFS of $\mathcal{F}$ represents a MxIS of $\mathcal{G}$ and a MxClique of the complemented graph. Moreover, an MCFS corresponds to an MnVC of $\mathcal{G}$. Thus, for plain maximal falsifiability, an MFS can be computed in linear time [25].

The relationship between MFSes and MxISes yields a somewhat straightforward hitting set relationship. For any maximal independent set I, $V \setminus I$ represents a minimal vertex cover. An immediate observation is:

Proposition 3. *Given a graph $\mathcal{G} = (V, E)$ with a set of MnVCes $\mathbb{C}$, the minimal hitting sets of $\mathbb{C}$ are the edges of $\mathcal{G}$ and the minimal hitting sets of the edges of $\mathcal{G}$ are the MnVCes $\mathbb{C}$ of $\mathcal{G}$.*

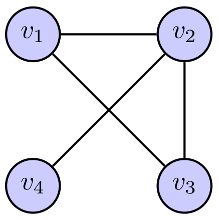
As a result, for the case of plain maximal falsifiability the following holds:

Proposition 4. *Let $\mathcal{F}$ be a set of soft clauses. Then:*
- *The MNFSes of $\mathcal{F}$ are the minimal hitting sets of the MCFSes $\mathcal{F}$ and vice-versa.*
- *Each MNFS of $\mathcal{F}$ consists of exactly two clauses and represents an edge in the graph $\mathcal{G}$ defined above.*
- *The number of MNFSes of $\mathcal{F}$ is $\mathcal{O}(m^2)$, where m denotes the number of clauses in $\mathcal{F}$.*

Reductions of MaxClique to MaxSAT and MinSAT are well-known (e.g. [32]). For example, such reductions also allow solving MIS, MVC, with MaxSAT (and MinSAT) algorithms. A *new* encoding of MIS into MinSAT can be devised, which does not use hard clauses. Given an undirected graph $\mathcal{G} = (V, E)$, one can construct a set of clauses $\mathcal{F}$ such that each vertex $v_i \in V$ is represented by a clause $c_i \in \mathcal{F}$. For each edge $e = (v_i, v_j)$ a new variable x_e is introduced in $\mathcal{F}$ such that $x_e \in c_i$ and $\neg x_e \in c_j$.

Example 2. Consider the graph $\mathcal{G} = (V, E)$, with $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_2, v_4)\}$, shown in Figure 1a. Each vertex v_i is represented by a clause c_i and for each edge (v_i, v_j) a new variable x_{v_i,v_j} is introduced. The graph can be represented by a set of clauses $\mathcal{F}$ (Figure 1b) in the following way: $c_1 = x_{v_1,v_2} \vee x_{v_1,v_3}$, $c_2 = \neg x_{v_1,v_2} \vee x_{v_2,v_3} \vee x_{v_2,v_4}$, $c_3 = \neg x_{v_1,v_3} \vee \neg x_{v_2,v_3}$, $c_4 = \neg x_{v_2,v_4}$. A maximal independent set of $\mathcal{G}$ corresponds to an MFS of $\mathcal{F}$.

We now consider other formulations of maximal falsifiability, where $\mathcal{H} \neq \emptyset$ and where each soft clause c is associated a non-unit weight. As a result, a weight is also associated with each MFS, MCFS and MNFS.



(a) Graph $\mathcal{G}$ (see Example 2)

$$\mathcal{F} = \left\{ \begin{array}{l} c_1 = x_{v_1,v_2} \vee x_{v_1,v_3}, \\ c_2 = \neg x_{v_1,v_2} \vee x_{v_2,v_3} \vee x_{v_2,v_4}, \\ c_3 = \neg x_{v_1,v_3} \vee \neg x_{v_2,v_3}, \\ c_4 = \neg x_{v_2,v_4} \end{array} \right\}$$

(b) Set of clauses $\mathcal{F}$ for $\mathcal{G}$

Fig. 1. From maximal independent set to maximal falsifiability

Similar to the MaxSAT case, the problems considered for MaxFalse/MinSAT are plain ($\mathcal{H} = \emptyset$ and unit weights), partial ($\mathcal{H} \neq \emptyset$ and unit weights), weighted ($\mathcal{H} = \emptyset$ and arbitrary weights), and partial weighted ($\mathcal{H} \neq \emptyset$ and arbitrary weights) MaxFalse/MinSAT. Observe that these definitions follow earlier work for the concrete case of MinSAT (e.g. [34]).

MFSes for (partial) (weighted) maximal falsifiability problems are defined similarly to plain case, but $\mathcal{H}$ is required to be satisfied for the truth assignment that identifies the MFS. Moreover, the weighted versions of the MaxFalse and MinSAT problems are defined as follows.

Definition 7 (Partial Weighted Maximum Falsifiability). *Given a formula $\mathcal{F}$, with hard clauses $\mathcal{H}$, $\mathcal{H}$ is satisfiable, and soft clauses $\mathcal{R}$,* Maximum Falsifiability *(MaxFalse) denotes the problem of computing the MFS of $\mathcal{F}$ with the largest weight.*

Definition 8 (Partial Weighted Minimum Satisfiability). *Given a formula $\mathcal{F}$, with hard clauses $\mathcal{H}$ and soft clauses $\mathcal{R}$,* Minimum Satisfiability *(MinSAT) is the problem of computing a subset of clauses of $\mathcal{R}$ with the smallest weight, that together with $\mathcal{H}$ are simultaneously satisfiable (while the other clauses of $\mathcal{R}$ are falsified).*

A simple observation is that although weights can be associated with MFSes, MCFSes and MNFSes, their number is independent of the weight function (e.g. [41]).

For the cases where $\mathcal{H} \neq \emptyset$, the problems of computing MFSes and MxISes are no longer equivalent. Observe that, when $\mathcal{H} \neq \emptyset$, finding an MFS becomes NP-hard. A proof is immediate, since we can reduce SAT to MaxFalse: $\mathcal{H}$ corresponds to the original clauses and there are no soft clauses. Section 6 revisits the difference between MFSes and MxISes in the case $\mathcal{H} \neq \emptyset$, and also general minimal hitting results.

4 Algorithms for Maximal Falsifiability

As indicated in Section 3, there are linear time algorithms for plain maximal falsifiability, while the general case of the problem is NP-hard. Since there are several encodings of MinSAT into MaxSAT (e. g. [21, 31, 45]), these encodings can be also used for solving the maximal falsifiability problem. One of the most effective encodings of MinSAT is the so-called $\top$-encoding[1]. It consists in negating each soft clause c of the formula,

[1] The reader is referred to [21] for the details of $\top$-encoding.

Algorithm 1. Basic linear search (BLS)

```
Function BLS($\mathcal{F} = \mathcal{H} \cup \mathcal{R}$)
    $(\mathsf{st}, \mathcal{A}) \leftarrow \mathtt{SAT}(\mathcal{H})$                         # initial SAT call
    if $st = \mathsf{false}$ then
        return $(\mathsf{false}, \emptyset)$

    $\mathcal{C} \leftarrow \mathcal{R}$
    FilterFalsifiedClauses($\mathcal{H}, \mathcal{C}, \mathcal{A}$)

    foreach $c \in \mathcal{C}$ do                  # trying to falsify each clause
        $(\mathsf{st}, \mathcal{A}) \leftarrow \mathtt{SAT}(\mathcal{H} \cup \{\neg c\})$
        if $\mathsf{st} = \mathsf{true}$ then
            $\mathcal{H} \leftarrow \mathcal{H} \cup \{\neg c\}$
            $\mathcal{C} \leftarrow \mathcal{C} \setminus \{c\}$
            FilterFalsifiedClauses($\mathcal{H}, \mathcal{C}, \mathcal{A}$)

    return $(\mathsf{true}, \mathcal{R} \setminus \mathcal{C})$
```

which results in constructing a group of unit clauses $g = \{\neg c\}$. Group g is then relaxed by a relaxation variable r and made hard, while the clause $\neg r$ is added to the soft part of the formula.

Let $\mathcal{F}$ be a CNF formula, and $\top$-Enc($\mathcal{F}$) be its $\top$-encoding into MaxSAT. Observe that there is a one to one correspondence between an MFS of $\mathcal{F}$ and an MSS of $\top$-Enc($\mathcal{F}$), an MCFS of $\mathcal{F}$ and an MCS of $\top$-Enc($\mathcal{F}$) and, finally, an MNFS of $\mathcal{F}$ and an MUS of $\top$-Enc($\mathcal{F}$). Moreover, the size of an MFS of $\mathcal{F}$ equals the size of the corresponding MSS of $\top$-Enc($\mathcal{F}$), and similarly for MCFSes/MCSes and MNFSes/MUSes of $\mathcal{F}$ and $\top$-Enc($\mathcal{F}$), respectively. Thus, one approach for finding an MFS of formula $\mathcal{F}$ is to find an MSS (or its complement — MCS) of $\top$-Enc($\mathcal{F}$), e.g. with recently proposed algorithms for computing MCSes [38, 42]. Nevertheless, this paper proposes instead *native* algorithms for both maximal and maximum falsifiability.

Let $\mathcal{H}$ and $\mathcal{R}$ denote the hard and soft clauses of $\mathcal{F}$, respectively. To find an MFS of $\mathcal{F}$, one needs to determine a subset of $\mathcal{R}$ that is a maximally all-falsifiable set, subject to the models of $\mathcal{H}$. Therefore, during the search it is necessary to call a SAT oracle.

Algorithm 1 shows the pseudo-code of the *Basic Linear Search* (*BLS*) algorithm for the general case of maximal falsifiability, inspired on algorithms for MCSes [38, 42]. Given a $\mathcal{H}$ and $\mathcal{R}$, denoting the hard and soft clauses of $\mathcal{F}$, the algorithm finds an MFS $\mathcal{M}$, $\mathcal{M} \subseteq \mathcal{R}$, of formula $\mathcal{F}$. Algorithm 1 is based on the connection between MinSAT and MaxFalse and at first finds a solution of the minimal satisfiability problem for $\mathcal{F}$, i. e. an MCFS $\mathcal{C}$, $\mathcal{C} \subset \mathcal{R}$, and then uses it to compute the complementary MFS $\mathcal{M} = \mathcal{R} \setminus \mathcal{C}$. First, Algorithm 1 checks whether the hard part of the formula is satisfiable or not (see line 3). If it is not, the BLS algorithm returns an empty MFS. Otherwise, it initializes the correction set $\mathcal{C}$ to be equal to $\mathcal{R}$ (line 5). At each iteration of the main loop (lines 7–12) Algorithm 1 tries to reduce $\mathcal{C}$ by removing a single clause $c \in \mathcal{C}$. This is done by checking whether clause c can be falsified together with

Algorithm 2. MaxFalse Binary Search (MFBS)

```
1  Function MFBS($\mathcal{F} = \mathcal{H} \cup \mathcal{R}$)
2      $(\mathcal{F}_w, R, W) \leftarrow (\mathcal{H}, array[\mathcal{R}.\texttt{size}()], array[\mathcal{R}.\texttt{size}()])$
3      foreach $(c_i, w_i) \in \mathcal{R}$ do
4          $(R[i], W[i]) \leftarrow (r_i, w_i)$                # $r_i$ fresh relaxation variable
5          foreach $l_{i_j} \in c_i$ do $\mathcal{F}_w \leftarrow \mathcal{F}_w \cup \{(\neg l_{i_j} \vee r_i)\}$
6      $(\lambda, \mu, last\mathcal{A}) \leftarrow (\texttt{ComputeLB}(\mathcal{R}), \texttt{ComputeUB}(\mathcal{F}_w), \emptyset)$
7      while $\lambda \neq \mu$ do
8          $\kappa \leftarrow \lfloor \frac{\lambda+\mu}{2} \rfloor$
9          $(\mathsf{st}, \mathcal{A}) \leftarrow \texttt{SAT}(\mathcal{F}_w \cup \texttt{CNF}(\sum_{i=0}^{\mathcal{R}.size()-1} W[i] \times R[i] \leq \kappa))$
10         if $\mathsf{st} = \mathsf{true}$ then $(last\mathcal{A}, \mu) \leftarrow (\mathcal{A}, \texttt{GetSolution}(\mathcal{R}, \mathcal{A}))$
11         else $\lambda \leftarrow \texttt{SubSetSum}(W, \kappa)$
12     return Falsified($\mathcal{R}, last\mathcal{A}$)
```

clauses that were falsified at previous iterations (line 8). A possible improvement of the BLS algorithm is that instead of removing just one clause c from $\mathcal{C}$ per iteration, one can filter all clauses from $\mathcal{C}$ that were falsified by each SAT call. This is done by calling a function FilterFalsifiedClauses($\mathcal{H}, \mathcal{C}, \mathcal{A}$) (line 6 and line 12), where $\mathcal{A}$ is a model of $\mathcal{H} \cup \{\neg c\}$ returned by the oracle. Every clause falsified by $\mathcal{A}$ is removed from $\mathcal{C}$, and its negation is then made hard (added to $\mathcal{H}$). Note that calling the function FilterFalsifiedClauses($\mathcal{H}, \mathcal{C}, \mathcal{A}$) can significantly reduce the number of SAT calls.

5 Algorithms for Maximum Falsifiability

A solution to the MaxFalse problem can be obtained by computing a solution to the MinSAT problem (Proposition 2). On the other hand, and as mentioned in Section 4 several encodings have been proposed to translate MinSAT into MaxSAT. Thus, the MaxFalse problem can be computed by encoding the problem into MaxSAT. This paper proposes instead *native* algorithms for the MaxFalse problem.

The three algorithms proposed are based on iteratively calling a SAT solver, to determine if a subset of the soft clauses with a maximum current cost exists. The idea is similar to the classical iterative SAT-based MaxSAT solvers. Initially each soft clause is relaxed by associating to the clause a relaxation variable (a fresh Boolean variable). This process of relaxing a soft clause, guarantees that whenever a soft clause is satisfied by an assignment, then its associated relaxation variable is assigned true. In each iteration a constraint is added to the formula sent to the SAT solver, in order to force a current maximum cost on the set of relaxation variables assigned true. The current cost of each iteration depends on the bounds being refined, either a lower bound, an upper bound or both. The three algorithms proposed correspond to the three types of search possible (to refine the bounds): Binary search (MFBS) (which refines both an upper and a lower bound); Linear search starting from a lower bound (named Linear search UNSAT-SAT, MFLSUS); and Linear search starting from an upper bound (named Linear search SAT-UNSAT, MFLSSU). In the following the pseudo-code of the Binary search algorithm is presented and described, while the pseudo-codes of the Linear search algorithms are presented in appendix B.

Algorithm 2 shows the pseudo-code of the *MaxFalse Binary Search (MFBS)* algorithm for maximum falsifiability. First, Algorithm 2 obtains a working formula $\mathcal{F}_w$ by relaxing all the soft clauses in $\mathcal{R}$ together with all the hard clauses. In this case, the relaxation of the soft clauses is not the usual relaxation as in MaxSAT. Instead, the algorithm follows the relaxation of soft clauses as in the Model-Guided algorithm [21] for MinSAT. A fresh relaxation variable is associated with the original soft clause, and a set of binary clauses is added to the working formula, each containing the negation of a literal of the soft clause, and the associated relaxation variable (line 5).

The algorithm proceeds by computing an initial lower and upper bound in line 6. The upper bound is computed by calling the SAT solver on the working formula with preferences set for the relaxation variables, that is the relaxation variables are prefered to be falsified. Then the sum of weights of the relaxation variables set to true corresponds to the upper bound.

The lower bound is computed by a greedy heuristic. Each variable is associated with two sums: the sum of weights of the soft clauses where the variable appears as a positive literal, and the sum of weights of the soft clauses where the variable appears as a negative literal. The minimum value of the two sums is obtained, and associated with the variable, as the minimum weight necessary to satisfy due to an assignment to the variable. Then the maximum among all variables is computed and added to the lower bound. Afterwards the clauses associated to the variable with the maximum value are processed by deleting all clauses associated to the minimum sum. The process is repeated until there are no more variables.

Lines 7-11 present the main loop of the MFBS algorithm. In each iteration the algorithm computes a value κ in the middle of the bounds, and makes a call to the SAT solver with the working formula $\mathcal{F}_w$ together with a constraint (encoded into CNF) enforcing the maximum allowed cost to be at most κ. If the SAT solver returns true, then the satisfying assignment is recorded and the upper bound μ is updated accordingly. If the SAT solver returns false, then the lower bound is updated to the next allowed weight considering the set of weights. Such weight is obtained by the `SubSetSum`() function similar to [3].

The algorithm iterates until both bounds are the same (line 7), and returns a set of soft clauses falsified by the last assignment with function `Falsified`() in line 12.

6 Minimal Hitting Sets and Enumeration Problems

One of the most important practical applications of the Maximal Satisfiability problem is enumeration of MUSes of a CNF formula. The idea of the method is based on the well-known relationship of minimal hitting set duality between MCSes and MUSes: each MCS (MUS) of a CNF formula is a minimal hitting set (or a *minimal set cover*) of the complete set of MUSes (MCSes) of the formula. The corresponding theoretical results were considered in [12,43]. Enumeration of MUSes based on enumerating MCSes was done in [10,36,42]. The duality relationship between MCSes and MUSes was also used for solving the SMUS problem in [22, 35]. The approach did not consist in enumerating all MCSes and MUSes — instead, in order to get a lower bound on the size of the smallest MUS, only some MCSes were computed.

This section proves that the relationship of a minimal hitting set duality also exists for the case of Maximal Falsifiability, i. e. between MCFSes and MNFSes. The corresponding assertions are presented in the form of theorems. Two auxiliary propositions are used in the proofs. Hereinafter, letters $\mathcal{M}$, $\mathcal{N}$, and $\mathcal{C}$ are used to denote an MFS, an MNFS, and an MCFS of a CNF formula, respectively. The complete sets of MFSes, MNFSes and MCFSes of a CNF formula $\mathcal{F}$ are denoted by $\mathbb{M}(\mathcal{F})$, $\mathbb{N}(\mathcal{F})$, and $\mathbb{C}(\mathcal{F})$.

Proposition 5. *Formula $\mathcal{F}$ is not all-falsifiable iff it contains at least one MNFS.*

Proof. Such an MNFS can be constructed by a simple algorithm that finds a pair of clauses in $\mathcal{F}$ that contain a complemented literal. The opposite is trivial. □

Proposition 6. *A set of clauses $\mathcal{U}$, $\mathcal{U} \subseteq \mathcal{F}$, is all-falsifiable iff there is an MCFS $\mathcal{C}$ such that $\mathcal{U} \cap \mathcal{C} = \emptyset$.*

Proof. It follows from the fact that for any all-falsifiable subset $\mathcal{U}$, $\mathcal{U} \subseteq \mathcal{F}$, there is an MFS $\mathcal{M}$ such that $\mathcal{U} \subseteq \mathcal{M} \subseteq \mathcal{F}$. By definition, for any MFS $\mathcal{M}$ there exists a complementary MCFS $\mathcal{C} = \mathcal{F} \setminus \mathcal{M}$. It is not hard to see that $\mathcal{U} \cap \mathcal{C} = \emptyset$. □

Theorem 1. *Subformula $\mathcal{C}$, $\mathcal{C} \subset \mathcal{F}$, is an MCFS iff $\mathcal{C}$ is a minimal hitting set of $\mathbb{N}(\mathcal{F})$.*

Proof. Proposition 5 implies that subformula $\mathcal{C}$ is a hitting set of $\mathbb{N}(\mathcal{F})$ iff the complementary subformula $\mathcal{M} = \mathcal{F} \setminus \mathcal{C}$ is all-falsifiable (otherwise, $\mathcal{M}$ contains at least one MNFS that is not hit by $\mathcal{C}$).

Let $\mathcal{C} \subset \mathcal{F}$ be a *minimal* hitting set of $\mathbb{N}(\mathcal{F})$. This means that $\mathcal{M}$ is all-falsifiable, and $\forall_{c \in \mathcal{C}}$ formula $\mathcal{C} \setminus \{c\}$ is not a hitting set of $\mathbb{N}(\mathcal{F})$. Assume, that $\mathcal{M}$ is not an MFS of $\mathcal{F}$, i. e. $\exists_{c \in \mathcal{C}}$ such that $\mathcal{M} \cup \{c\}$ is still all-falsifiable. This implies that $\mathcal{C} \setminus \{c\}$ is a hitting set of $\mathbb{N}(\mathcal{F})$ — contradiction. Hence, $\mathcal{M}$ is an MFS and $\mathcal{C}$ is MCFS of $\mathcal{F}$.

Let $\mathcal{C} \subset \mathcal{F}$ be an MCFS of formula $\mathcal{F}$. Then the complementary subformula $\mathcal{M}$ is an MFS, and $\mathcal{C}$ is a hitting set of $\mathbb{N}(\mathcal{F})$. Assume, that $\mathcal{C}$ is not a minimal hitting set of $\mathbb{N}(\mathcal{F})$. Then $\exists_{c \in \mathcal{C}}$ such that $\mathcal{C} \setminus \{c\}$ is still a hitting set of $\mathbb{N}(\mathcal{F})$. This means that its complementary subformula $\mathcal{M} \cup \{c\}$ is all-falsifiable. However, this contradicts the fact that $\mathcal{M}$ is an MFS of $\mathcal{F}$. Therefore, $\mathcal{C}$ is a minimal hitting set of $\mathbb{N}(\mathcal{F})$. □

Theorem 2. *Subformula $\mathcal{N}$, $\mathcal{N} \subseteq \mathcal{F}$, is an MNFS iff $\mathcal{N}$ is a minimal hitting set of $\mathbb{C}(\mathcal{F})$.*

Proof. Proposition 6 implies that subformula $\mathcal{N}$ is not all-falsifiable iff $\mathcal{N}$ has a non-empty intersection with all the MCFSes of $\mathcal{F}$, i. e. $\mathcal{N}$ is a hitting set of $\mathbb{C}(\mathcal{F})$.

Let $\mathcal{N}$ be an MNFS of formula $\mathcal{F}$. Irreducibility of $\mathcal{N}$ ensures that any subformula $\mathcal{N}'$, $\mathcal{N}' \subset \mathcal{N}$, is an all-falsifiable formula. Hence, by Proposition 6, $\mathcal{N}'$ does not *hit* all the MCFSes of $\mathcal{F}$. Thus, $\mathcal{N}$ is a minimal hitting set of $\mathbb{C}(\mathcal{F})$.

Let $\mathcal{N}$ be a minimal hitting set of $\mathbb{C}(\mathcal{F})$. This means that $\forall_{c \in \mathcal{N}}$ formula $\mathcal{N} \setminus \{c\}$ does not hit all the MCFSes of $\mathcal{F}$, i. e. there is an MCFS $\mathcal{C}$ such that $\mathcal{N} \setminus \{c\} \cap \mathcal{C} = \emptyset$. Hence, $\mathcal{N} \setminus \{c\}$ is a subset of MFS $\mathcal{M} = \mathcal{F} \setminus \mathcal{C}$, and, therefore, is all-falsifiable. By definition, subformula $\mathcal{N}$ is an MNFS of $\mathcal{F}$. □

Observe that the proofs of the propositions presented above make use only of the general definitions of an MFS, MCFS, and MNFS described in Section 3. Therefore, the propositions hold for both plain and partial maximal falsifiability.

It should be noted that in contrast to Maximal Satisfiability, for the case of Maximal Falsifiability it can be more helpful to enumerate MNFSes instead of MCFSes. A set of MNFSes can give us a lower bound on the size of each MCFS and, hence, an upper bound on the optimal value for MaxFalse. Therefore, this can be used to bootstrap algorithms that refine an upper bound (see Section 5).

Observe that there is no correspondence between computing MFSes and MxISes for the case $\mathcal{H} \neq \emptyset$ because of the different interpretations of the hard constraints. Although the maximal independent set problem does not consider a concept of a hard constraint (in this sense computing an MFS is a more general problem than computing an MxIS), one can consider the *weighted* version of the problem. While for the case of partial maximal falsifiability each clause $c \in \mathcal{H}$ must be *satisfied*, vertices with a high weight in the weighted maximal independent set problem are preferable to be *independent*. Hence, there is no translation from one problem into another similar to the one described[2] in Section 3.

In contrast to plain maximal falsifiability, for which MNFSes are known to contain exactly two clauses (see Proposition 4), formulas with hard clauses may have MNFSes that contain just one clause. This fact is shown below.

Proposition 7. *Let $\mathcal{F}$ be a pair of sets of clauses $(\mathcal{H}, \mathcal{R})$, where clauses of $\mathcal{H}$ are hard while clauses of $\mathcal{R}$ are soft (relaxable). Then if there exists a subset of clauses $\mathcal{W} \subseteq \mathcal{R}$ such that $\mathcal{H} \models \mathcal{W}$, then $\mathcal{W}$ is included into all MCFSes of $\mathcal{F}$.*

Proof. Proof by contradiction. Let $\mathcal{W}$ be a subset of $\mathcal{R}$ such that $\mathcal{H} \models \mathcal{W}$. Assume, that there exists MCFS $\mathcal{C}$ such that $\mathcal{W} \not\subseteq \mathcal{C}$. This means that an MFS $\mathcal{M} = \mathcal{R} \setminus \mathcal{C}$ intersects $\mathcal{W}$, i. e. $\mathcal{M} \cap \mathcal{W} \neq \emptyset$. Entailment $\mathcal{H} \models \mathcal{W}$ means that each clause $c \in \mathcal{W}$ is satisfied by all models of $\mathcal{H}$. By definition, all clauses of $\mathcal{M}$ can be falsified simultaneously by some model of $\mathcal{H}$, Therefore, $\mathcal{M} \cap \mathcal{W} = \emptyset$ — contradiction. □

Corollary 1. *Let $\mathcal{F}$ be a pair of sets of clauses $(\mathcal{H}, \mathcal{R})$, where clauses of $\mathcal{H}$ are hard while clauses of $\mathcal{R}$ are soft (relaxable). Then if there exists a subset of clauses $\mathcal{W} \subseteq \mathcal{R}$ such that $\mathcal{H} \models \mathcal{W}$, then for any clause $c_i \in \mathcal{W}$ set $\{c_i\}$ is an MNFS of $\mathcal{F}$.*

Proof. Implied by Proposition 7 and Theorem 2. □

Note that Corollary 1 gives a sufficient condition of partial formulas having MNFSes of size 1. Furthermore, it can be observed that formulas with hard clauses can also have MNFSes of size > 2. Indeed, let us consider a set of clauses $\mathcal{F}' = \{c_1 = \neg x_1 \vee \neg x_2, c_2 = \neg x_1 \vee x_2, c_3 = x_1 \vee \neg x_2, c_4 = x_1 \vee x_2\}$. Clearly, this plain CNF formula is unsatisfiable (moreover, it is an MUS). By negating $\mathcal{F}'$ with the use of auxiliary variables, one can get a negation of $\mathcal{F}'$ — a partial CNF formula $\mathcal{F} = \mathcal{H} \cup \mathcal{R}$, where clauses of $\mathcal{H}$ encode equivalences of the form $t_i \equiv c_i$, $i \in \{1, \ldots, 4\}$, and $\mathcal{R} =$

[2] Recall that the connection between plain maximal falsifiability and maximal independent set in Section 3 established the correspondence between *independent* vertices and clauses that can be *simultaneously falsified*.

$\{\neg t_1, \neg t_2, \neg t_3, \neg t_4\}$. Formula $\mathcal{F}$ has only one MNFS, which corresponds to the MUS of $\mathcal{F}'$ and contains all 4 clauses of $\mathcal{R}$.

This immediately shows that the number of all MNFSes of partial formulas cannot be polynomial in general. Although this implies that enumeration of MNFSes for such formulas may not be feasible in practice, instead of enumerating MNFSes of $\mathcal{F} = \mathcal{H} \cup \mathcal{R}$, we can efficiently enumerate MNFSes of $\mathcal{R}$ (see Proposition 4) and use them to compute a lower bound for bootstrapping the algorithms for MaxFalse.

7 Experimental Results

This section describes the experimental results obtained for Maximal Falsifiability as well as for Maximum Falsifiability. The first section shows a comparison on the quality of the solution obtained for Maximal Falsifiability. Then section two presents a study on the performance of the algorithms proposed for Maximum Falsifiability.

The algorithms described in this paper were implemented in C++ using incremental SAT solvers. The experiments were performed on an HPC cluster, with quad-core Intel Xeon E5450 3 GHz nodes with 32 GB of memory. In order to evaluate the performance of the algorithms in *real* industrial problems (not random nor crafted problems), all the Industrial Partial MaxSAT and Industrial Weighted Partial benchmarks from the MaxSAT Evaluation 2013[3] were collected. The collected MaxSAT benchmarks were transformed into MaxFalse instances by selecting the ones that only contained unit soft clauses and by negating the unit literals on those instances. A total of 935 MaxFalse instances were obtained.

As explained in Section 4, a different alternative to Maximal/Maximum Falsifiability consists in transforming the MaxFalse instance by encoding it into MaxSAT via for example the $\top-$encoding, and then computing a MCS/MaxSAT of the resulting MaxSAT instance. In our experiments, whenever we compare with this encoding approach, since the original instances only contain unit soft clauses, then no aditional variables or clauses are added, and this corresponds to solving MaxSAT on the original instances obtained from the MaxSAT Evaluation.

7.1 Maximal Falsifiability

The BLS Algorithm 1 of Section 4 was implemented in a tool *mxlFalse*. The underlying SAT solver of the maximumFalse tool is the Minisat 2.2 [17].

This section studies the quality of the solutions obtained. The cost of the MCFSes obtained with mxlFalse is compared against the value of the upper bound heuristic of Section 5, as well as against the cost of the MCSes obtained by mcsls2 [38] (an efficient MCS extractor) after transforming the instance into MaxSAT.

In the experiments both the mxlFalse and mcsls2 were set to enumerate MCFSes and MCFS (respectively) for 3 min, whereupon the minimum cost MCFS/MCS was obtained. The results obtained were then divided by the optimum cost, and the values were plotted in the scatter plots of Figure 2. Figure 2 (a) compares the value of mxlFalse

[3] `http://maxsat.ia.udl.cat/`

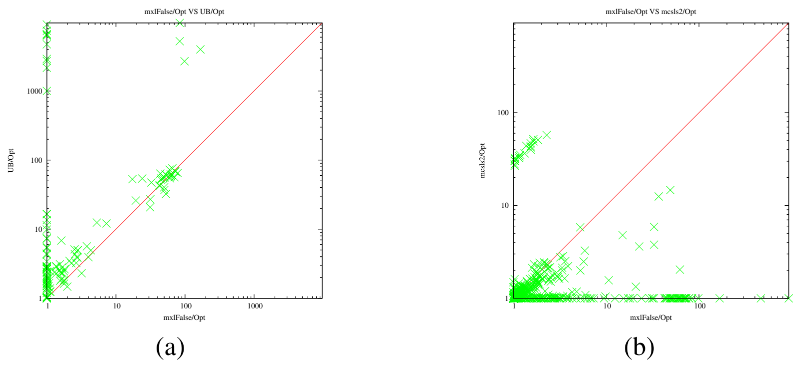


Fig. 2. (a) scatter plot between mxlfalse and ub (b) scatter plot between mxlfalse and mcsls2

(divided by the optimum cost) and the value obtained by the upper bound (divided by the optimum cost). It can be seen from the scatter plot that in the vast majority of cases the value of mxlFalse is closer to 1 (thus, closer to the optimum) than the value of the upper bound (UB). In particular all the instances in the left hand side of the plot.

Figure 2 (b) compares the value of mxlFalse (divided by the optimum cost) and the value obtained by mcsls2 (divided by the optimum cost). In this case, mcsls2 produces solutions closer to the optimum, but nevertheless there are instances that mcsls2 is 10 times over the optimum while mxlFalse is very close. In such situations, it would be worthwhile to additionally consider running mxlFalse.

7.2 Maximum Falsifiability

The Maximum False algorithms proposed in Section 5 were implemented in a tool called *maximumFalse*. Namely, the following algorithms were implemented: binary search (maximumFalse-b), linear search unsat-sat (maximumFalse-lsus) and linear search sat-unsat (maximumFalse-lssu). This section presents results on the performance of the previous algorithms running for 1800 seconds with 4GB of memory limit. The underlying SAT solver of the maximumFalse tool is the Glucose SAT solver [7].

Figure 3 presents a cactus plot for the previous algorithms on the MaxFalse benchmarks considered. From the figure it can be seen that binary search (maximumFalse-b) is much better than the linear search approaches (maximumFalse-lsus/-lssu), while the linear search unsat-sat outperforms (slighty) the linear search sat-unsat.

Figure 3 also shows the running times for two core-guided MaxSAT algorithms running on the original MaxSAT instances. Namelly, we consider bcd2 [40] and wpm1 [4] (2012 version). As expected, the core-guided MaxSAT approaches outperform maximumFalse algorithms. Note that the MaxSAT algorithms result from over a decade of research, while in this paper we are proposing a more general framework for MaxFalse. Also, it is unknown in the literature how to take advantage of core-guided algorithms natively in MaxFalse algorithms, whether if it is even possible.

Nevertheless, we considered the use of virtual best solvers (VBS) between the maximumFalse solvers and the core-guided algorithms. Interestingly, the VBSes considering

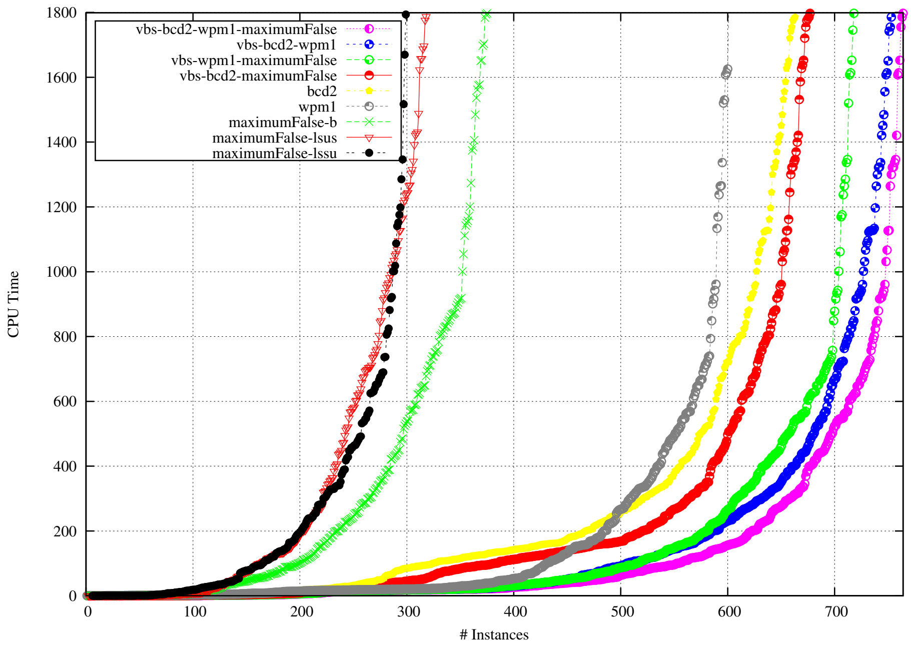


Fig. 3. Cactus plot

maximumFalse solvers consistently outperform their counterpart without the maximum-False solvers. For example, vbs-wpm1-maximumFalse is able to solver over 100 more instances than wpm1 alone.

Such results indicate that porfolio approaches for MaxSAT solver can benefit from the inclusion of maximumFalse solvers.

8 Conclusions

Motivated by the recent interest in MinSAT, this paper develops a comprehensive characterization of this problem, which follows the one developed earlier for MaxSAT. To achieve this goal, the paper introduces the problems of maximum and maximal falsifiability. The case of plain maximal falsifiability is shown to correspond to the computation of a maximal independent set in an undirected graph. Also, the paper develops a reduction of maximal independent set into maximal falsifiability (and so to minimal satisfiability), which does not involve hard clauses. Moreover, as pointed out, maximal falsifiability can be viewed as a more general formulation (with respect to maximal independent set), as it allows hard clauses to be considered. Maximal falsifiability is also used to introduce a number of new concepts: maximal falsifiable subsets (MSFes), minimal correction for falsifiability subsets (MCFSes), and minimal non-falsifiability subsets (MNFSes). In addition, the paper develops native algorithms for both maximal and maximum falsifiability, namely algorithms for computing one MFS and for solving the MaxFalse problem, and shows how these problems can be solved by reduction to MaxSAT. Finally, minimal hitting set duality between MCFSes and MNCSes is proven for the general (partial) case. The experimental results are interesting in that algorithms for maximal/maximum falsifiability show promise to be used in portfolios of algorithms for maximal/maximum satisfiability.

The work described in the paper opens a significant number of research directions. Concrete examples include additional algorithms for computing MFSes and for MaxFalse, the integration of MaxFalse algorithms in portfolios of MaxSAT algorithms, among others.

Acknowledgments. This work is partially supported by SFI PI grant BEACON (09-/IN.1/I2618), FCT grants ATTEST (CMU-PT/ELE/0009/2009) and POLARIS (PTDC-/EIA-CCO/123051/2010), INESC-ID's multiannual PIDDAC funding PEst-OE/EE-I/LA0021/2011, and by Spanish MICINN Project ARINF (TIN2009-14704-C03-01).

References

1. Akkoyunlu, E.A.: The enumeration of maximal cliques of large graphs. SIAM J. Comput. 2(1), 1–6 (1973)
2. Angel, E., Bampis, E., Gourvès, L.: On the minimum hitting set of bundles problem. Theor. Comput. Sci. 410(45), 4534–4542 (2009)
3. Ansótegui, C., Bonet, M.L., Levy, J.: A new algorithm for weighted partial maxsat. In: AAAI (2010)
4. Ansótegui, C., Bonet, M.L., Levy, J.: Sat-based maxsat algorithms. Artif. Intell. 196, 77–105 (2013)
5. Ansotegui, C., Li, C.M., Manya, F., Zhu, Z.: A SAT-based approach to MinSAT. In: CCIA, pp. 185–189 (2012)
6. Argelich, J., Li, C.-M., Manyà, F., Zhu, Z.: MinSAT versus MaxSAT for optimization problems. In: Schulte, C. (ed.) CP 2013. LNCS, vol. 8124, pp. 133–142. Springer, Heidelberg (2013)
7. Audemard, G., Simon, L.: Predicting learnt clauses quality in modern sat solvers. In: IJCAI, pp. 399–404 (2009)
8. Avidor, A., Zwick, U.: Approximating MIN k-SAT. In: Bose, P., Morin, P. (eds.) ISAAC 2002. LNCS, vol. 2518, pp. 465–475. Springer, Heidelberg (2002)
9. Avidor, A., Zwick, U.: Approximating MIN 2-SAT and MIN 3-SAT. Theory Comput. Syst. 38(3), 329–345 (2005)
10. Bailey, J., Stuckey, P.J.: Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In: Hermenegildo, M.V., Cabeza, D. (eds.) PADL 2004. LNCS, vol. 3350, pp. 174–186. Springer, Heidelberg (2005)
11. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (2009)
12. Birnbaum, E., Lozinskii, E.L.: Consistent subsets of inconsistent systems: structure and behaviour. J. Exp. Theor. Artif. Intell. 15(1), 25–46 (2003)
13. Bourke, C., Deng, K., Scott, S.D., Schapire, R.E., Vinodchandran, N.V.: On reoptimizing multi-class classifiers. Machine Learning 71(2-3), 219–242 (2008)
14. Brihaye, T., Bruyère, V., Doyen, L., Ducobu, M., Raskin, J.-F.: Antichain-based QBF solving. In: Bultan, T., Hsiung, P.-A. (eds.) ATVA 2011. LNCS, vol. 6996, pp. 183–197. Springer, Heidelberg (2011)

15. Butman, A., Hermelin, D., Lewenstein, M., Rawitz, D.: Optimization problems in multiple-interval graphs. ACM Transactions on Algorithms 6(2) (2010)
16. Chen, T., Filkov, V., Skiena, S.: Identifying gene regulatory networks from experimental data. Parallel Computing 27(1-2), 141–162 (2001)
17. Eén, N., Sörensson, N.: An Extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 502–518. Springer, Heidelberg (2004)
18. Gate, J., Stewart, I.A.: Frameworks for logically classifying polynomial-time optimisation problems. In: Ablayev, F., Mayr, E.W. (eds.) CSR 2010. LNCS, vol. 6072, pp. 120–131. Springer, Heidelberg (2010)
19. Goldstein, A., Kolman, P., Zheng, J.: Minimum common string partition problem: Hardness and approximations. Electr. J. Comb. 12 (2005)
20. Hassin, R., Monnot, J., Segev, D.: Approximation algorithms and hardness results for labeled connectivity problems. J. Comb. Optim. 14(4), 437–453 (2007)
21. Heras, F., Morgado, A., Planes, J., Marques-Silva, J.: Iterative SAT solving for minimum satisfiability. In: ICTAI, pp. 922–927 (2012)
22. Ignatiev, A., Janota, M., Marques-Silva, J.: Quantified maximum satisfiability: A core-guided approach. In: Järvisalo, M., Van Gelder, A. (eds.) SAT 2013. LNCS, vol. 7962, pp. 250–266. Springer, Heidelberg (2013)
23. Interian, Y., Corvera, G., Selman, B., Williams, R.: Finding small unsatisfiable cores to prove unsatisfiability of QBFs. In: ISAIM (2006)
24. Johnson, D.S., Papadimitriou, C.H., Yannakakis, M.: On generating all maximal independent sets. Inf. Process. Lett. 27(3), 119–123 (1988)
25. Karp, R.M., Wigderson, A.: A fast parallel algorithm for the maximal independent set problem. J. ACM 32(4), 762–773 (1985)
26. Kohli, R., Krishnamurti, R., Jedidi, K.: Subset-conjunctive rules for breast cancer diagnosis. Discrete Applied Mathematics 154(7), 1100–1112 (2006)
27. Kohli, R., Krishnamurti, R., Mirchandani, P.: The minimum satisfiability problem. SIAM J. Discrete Math. 7(2), 275–283 (1994)
28. Kügel, A.: Natural Max-SAT encoding of Min-SAT. In: Hamadi, Y., Schoenauer, M. (eds.) LION 6 2012. LNCS, vol. 7219, pp. 431–436. Springer, Heidelberg (2012)
29. Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R.: Generating all maximal independent sets: NP-hardness and polynomial-time algorithms. SIAM J. Comput. 9(3), 558–565 (1980)
30. Li, C.M., Manya, F.: MaxSAT, hard and soft constraints. In: Biere, et al. (eds.) [11], pp. 613–631
31. Li, C.M., Manyà, F., Quan, Z., Zhu, Z.: Exact MinSAT solving. In: Strichman, O., Szeider, S. (eds.) SAT 2010. LNCS, vol. 6175, pp. 363–368. Springer, Heidelberg (2010)
32. Li, C.M., Quan, Z.: Combining graph structure exploitation and propositional reasoning for the maximum clique problem. In: ICTAI, pp. 344–351 (2010)
33. Li, C.M., Zhu, Z., Manya, F., Simon, L.: Minimum satisfiability and its applications. In: IJCAI, pp. 605–610 (2011)
34. Li, C.M., Zhu, Z., Manya, F., Simon, L.: Optimizing with minimum satisfiability. Artif. Intell. 190, 32–44 (2012)
35. Liffiton, M.H., Mneimneh, M.N., Lynce, I., Andraus, Z.S., Marques-Silva, J., Sakallah, K.A.: A branch and bound algorithm for extracting smallest minimal unsatisfiable subformulas. Constraints 14(4), 415–442 (2009)
36. Liffiton, M.H., Sakallah, K.A.: Algorithms for computing minimal unsatisfiable subsets of constraints. J. Autom. Reasoning 40(1), 1–33 (2008)
37. Marathe, M.V., Ravi, S.S.: On approximation algorithms for the minimum satisfiability problem. Inf. Process. Lett. 58(1), 23–29 (1996)
38. Marques-Silva, J., Heras, F., Janota, M., Previti, A., Belov, A.: On computing minimal correction subsets. In: IJCAI (to appear 2013)

39. 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)
40. 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)
41. Morgado, A., Liffiton, M., Marques-Silva, J.: MaxSAT-based MCS enumeration. In: Biere, A., Nahir, A., Vos, T. (eds.) HVC. LNCS, vol. 7857, pp. 86–101. Springer, Heidelberg (2013)
42. Nöhrer, A., Biere, A., Egyed, A.: Managing SAT inconsistencies with HUMUS. In: VaMoS, pp. 83–91 (2012)
43. Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. 32(1), 57–95 (1987)
44. Tsukiyama, S., Ide, M., Ariyoshi, H., Shirakawa, I.: A new algorithm for generating all the maximal independent sets. SIAM J. Comput. 6(3), 505–517 (1977)
45. Zhu, Z., Li, C.-M., Manyà, F., Argelich, J.: A new encoding from MinSAT into MaxSAT. In: Milano, M. (ed.) CP 2012. LNCS, vol. 7514, pp. 455–463. Springer, Heidelberg (2012)

A List of Acronyms

CNF	Conjunctive Normal Form
SAT	Boolean Satisfiability
IS	Independent Set
MIS	Maximum Independent Set
MxIS	Maximal Independent Set
MaxClique	Maximum Clique
MxClique	Maximal Clique
MnVC	Minimal Vertex Cover
MVC	Minimum Vertex Cover
VC	Vertex Cover
MaxSAT	Maximum Satisfiability
MCS	Minimal Correction Subset
MSS	Maximal Satisfiable Subset
MUS	Minimal Unsatisfiable Subset
MaxFalse	Maximum Falsifiability
MCFS	Minimal Correction (for Falsifiability) Subset
MFS	Maximal Falsifiable Subset
MinSAT	Minimum Satisfiability
MNFS	Minimal Non-Falsifiable Subset

B Linear Search Algorithms for Maximum Falsifiability

Algorithm 3. MaxFalse Linear Search SAT-UNSAT(MFLSSU)

```
Function MFLSSU($\mathcal{F} = \mathcal{H} \cup \mathcal{R}$)
    $(\mathcal{F}_w, R, W) \leftarrow (\mathcal{H}, array[\mathcal{R}.\texttt{size}()], array[\mathcal{R}.\texttt{size}()])$
    foreach $(c_i, w_i) \in \mathcal{R}$ do
        $(R[i], W[i]) \leftarrow (r_i, w_i)$        # $r_i$ fresh relaxation variable
        foreach $l_{i_j} \in c_i$ do $\mathcal{F}_w \leftarrow \mathcal{F}_w \cup \{(\neg l_{i_j} \vee r_i)\}$
    $(\mathsf{st}, \mu, last\mathcal{A}) \leftarrow (\mathsf{true}, \texttt{ComputeUB}(\mathcal{F}_w), \emptyset)$
    while st = true do
        $(\mathsf{st}, \mathcal{A}) \leftarrow \texttt{SAT}(\mathcal{F}_w \cup \texttt{CNF}(\sum_{i=0}^{\mathcal{R}.size()-1} W[i] \times R[i] < \mu))$
        if st = true then $(last\mathcal{A}, \mu) \leftarrow (A, \texttt{GetSolution}(\mathcal{R}, \mathcal{A}))$
    return Falsified($\mathcal{R}, last\mathcal{A}$)
```

Algorithm 4. MaxFalse Linear Search UNSAT-SAT(MFLSUS)

```
Function MFLSUS($\mathcal{F} = \mathcal{H} \cup \mathcal{R}$)
    $(\mathcal{F}_w, R, W) \leftarrow (\mathcal{H}, array[\mathcal{R}.\texttt{size}()], array[\mathcal{R}.\texttt{size}()])$
    foreach $(c_i, w_i) \in \mathcal{R}$ do
        $(R[i], W[i]) \leftarrow (r_i, w_i)$                # $r_i$ fresh relaxation variable
        foreach $l_{i_j} \in c_i$ do $\mathcal{F}_w \leftarrow \mathcal{F}_w \cup \{(\neg l_{i_j} \vee r_i)\}$
    $(\mathsf{st}, \lambda, last\mathcal{A}) \leftarrow (\mathsf{false}, \texttt{ComputeLB}(\mathcal{R}), \emptyset)$
    while $\mathsf{st} = \mathsf{false}$ do
        $(\mathsf{st}, \mathcal{A}) \leftarrow \texttt{SAT}(\mathcal{F}_w \cup \texttt{CNF}(\sum_{i=0}^{\mathcal{R}.size()-1} W[i] \times R[i] \leq \lambda))$
        if $\mathsf{st} = \mathsf{false}$ then $\lambda \leftarrow \texttt{SubSetSum}(W, \lambda)$
        else $last\mathcal{A} \leftarrow A$
    return Falsified($\mathcal{R}, last\mathcal{A}$)
```