AI Communications 29 (2016) 351–370
DOI 10.3233/AIC-150685
IOS Press

Maximal falsifiability

Definitions, algorithms and applications

Alexey Ignatiev [a,*], Antonio Morgado [a], Jordi Planes [b] and Joao Marques-Silva [a,c]
[a] *INESC-ID, IST, University of Lisbon, Lisbon, Portugal*
E-mails: aign@sat.inesc-id.pt, ajrm@sat.inesc-id.pt
[b] *Universitat de Lleida, Lleida, Spain*
E-mail: jplanes@diei.udl.cat
[c] *UCD CASL, Dublin, Ireland*
E-mail: 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.

Keywords: Maximum falsifiability, minimum satisfiability, minimal hitting set duality, Boolean optimization

1. Introduction

Maximum and Minimum Satisfiability (resp. MaxSAT and MinSAT) are two well-known optimization extensions of Boolean Satisfiability (SAT) (e.g. [49,53,61]). 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 1990s [8,9,46,57], 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 [5,6,35,47,50,52,53,76].

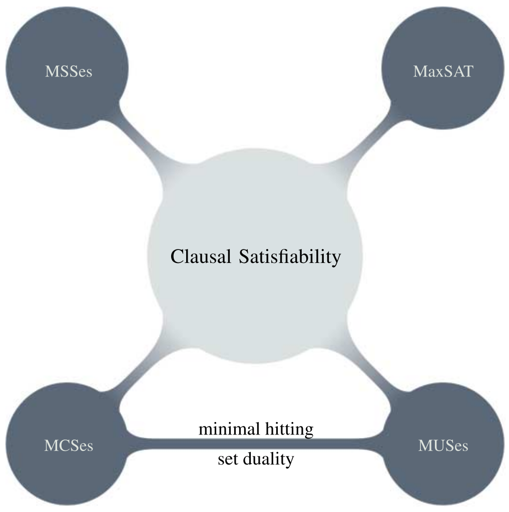
Like MaxSAT, MinSAT finds a growing number of practical applications (e.g. [15,16,23,33,37,39,45]), and it has also been used in complexity characterizations of other problems (e.g. [3,17,30,34]). More importantly, given a MaxSAT problem where the soft clauses are all unit, complementing the soft clauses gives a MinSAT problem. As shown in recent work (e.g. [6,53] 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) [13,56]. Intuitively, an MSS is a subset-maximal set of clauses that is satisfiable. 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

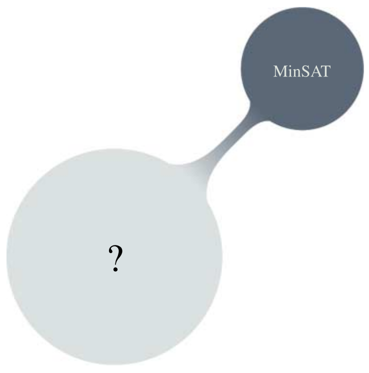
*Corresponding author. E-mail: aign@sat.inesc-id.pt.

MCS is a minimal hitting set of the Minimal Unsatisfiable Subsets (MUSes), and each MUS is a minimal hitting set of the MCSes [11,13,56,71]. In stark contrast with MaxSAT, and despite the vast body of work on MinSAT, similar results for the case of MinSAT are non-existent. What is the equivalent of an MSS in the case of MinSAT? And of an MCS? And of an MUS? Does there exist a minimal hitting set relationship in the case of MinSAT? What are the implications of these results? Given the large body of research and applications of MSSes, MCSes and MUSes, it comes through as fundamental to conduct a similar study in the case of MinSAT. This is what this paper proposes to do. Figure 1 compares the existing comprehensive body of knowledge about clausal satisfiability, and this includes MaxSAT and related problems, to the almost non-existing body of knowledge about clausal falsifiability, for which very little is currently known. The paper addresses these fundamental questions and lays the foundation for a more comprehensive characterization of the MinSAT problem. In turn, a better understanding of the MinSAT problem is expected to motivate further practical applications inasmuch the same way as for the MaxSAT case. A recent concrete example of the practical applications of MinSAT (and related concepts) is the reduction of maximum independent set to MinSAT [37]. This paper is an extended version of [38]. Its contributions are summarized as follows. First, the paper introduces *Maximal Falsifiability*, which represents the problem of computing subset-maximal sets of simultaneously falsifiable clauses. As shown in the paper, maximal falsifiability enables developing for the MinSAT case concepts similar to MSSes, MCSes and MUSes in the MaxSAT setting. 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 [44] can be used for computing a single maximal falsifiability solution. Similarly, algorithms for the enumeration of maximal independent set [41,48] 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 [11,13,56,71], 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 [11,56]. 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 through-



(a)



(b)

Fig. 1. Clausal satisfiability problems characterization. (a) Problems related to clausal satisfiability. (b) How to characterize MinSAT? (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-150685.)

out the paper. Section 3 introduces the Maximal and Maximum Falsifiability problems as well as related computational problems. Theoretical results for enumeration problems and minimal hitting sets are presented in Section 4. Section 5 develops algorithms for Maximal Falsifiability, whereas Section 6 develops algorithms for Maximum Falsifiability (and so for MinSAT). Section 7 provides experimental results for Maximal and Maximum Falsifiability, and Section 8 concludes the paper.

2. Preliminaries

This section provides the notation and the definitions used throughout the paper.

2.1. *Boolean satisfiability*

Boolean variables are represented by $X = \{x, y, z, x_1, y_1, z_1, \ldots\}$. A literal for a variable x is either a *positive* literal x or its negation $\neg x$. Hereinafter, literals are denoted by $\{l, l_1, l_2, \ldots\}$. Boolean formulas are represented in calligraphic font, $\mathcal{F}$, $\mathcal{M}$, $\mathcal{S}$, $\mathcal{T}$, $\mathcal{U}$, $\mathcal{W}$, $\mathcal{F}'$, etc. A Boolean formula in *conjunctive normal form* (*CNF*) is defined as a finite set of finite sets of literals. Whenever appropriate, a CNF formula will also be understood as a conjunction of disjunctions of literals, where each disjunction represents a *clause*. Formula $\mathcal{F}$ is assumed to be defined over the set of variables $\texttt{var}(\mathcal{F})$. 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 mapping $\mathcal{A} : \texttt{var}(\mathcal{F}) \mapsto \{0, 1\}$. The notion of assignment can be naturally extended to literals by setting $\mathcal{A}(\neg x) = 1 - \mathcal{A}(x)$. A clause is said to be *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 $\mathcal{F}$. If no model for formula $\mathcal{F}$ exists, the formula is called *unsatisfiable*.

2.2. *Maximum satisfiability*

The standard definition of the well-known optimization generalization of Boolean satisfiability called *maximum satisfiability* (or *MaxSAT*) is used, which is presented below.

Definition 1. Given a CNF formula $\mathcal{F}$, the *maximum satisfiability* (MaxSAT) problem for $\mathcal{F}$ consists in computing a maximum size (in terms of the number of clauses) subformula $\mathcal{F}' \subseteq \mathcal{F}$ such that $\mathcal{F}$ is satisfiable.

The following definitions also apply.

Definition 2 (Maximal satisfiable subset and minimal correction (for satisfiability) subset)**.** Sets $\mathcal{S}$ and $\mathcal{C}$ of clauses s.t. $\mathcal{S} \subseteq \mathcal{F}$ and $\mathcal{C} = \mathcal{F} \setminus \mathcal{S}$ are called a *Maximal Satisfiable Subset* (*MSS*) and a *Minimal Correction Subset* (*MCS*) of $\mathcal{F}$, respectively, if $\mathcal{S}$ is satisfiable and $\forall_{c \in \mathcal{C}}$ set $\mathcal{S} \cup \{c\}$ is unsatisfiable.

Definition 3 (Minimal unsatisfiable subset)**.** A set $\mathcal{U}$ of clauses, $\mathcal{U} \subseteq \mathcal{F}$, is called a *Minimal Unsatisfiable Subset* (*MUS*) if $\mathcal{U}$ is unsatisfiable and $\forall_{c \in \mathcal{U}}\ \mathcal{U} \setminus \{c\}$ is satisfiable. An MUS $\mathcal{U}$ of $\mathcal{F}$ of the smallest size is called a *smallest MUS* (*SMUS*).

MUSes and MCSes of a CNF formula are connected by the concept of *minimal hitting set* defined below.

Definition 4 (Minimal hitting set)**.** Given a collection Γ of sets from a universe $\mathbb{U}$, a hitting set for Γ is a set h such that $\forall_{S \in \Gamma} h \cap S \neq \emptyset$. A hitting set h is called *minimal* if none of its proper subsets is a hitting set.

The minimal hitting set duality between MUSes and MCSes is well known (e.g. see [13,56,71]).

Proposition 1 (Minimal hitting set duality)**.** *Given a CNF formula $\mathcal{F}$, let* MUSes($\mathcal{F}$) *and* MCSes($\mathcal{F}$) *be the set of all MUSes and MCSes of $\mathcal{F}$, respectively. Then the following holds*:

(1) *A subset $\mathcal{U}$ of $\mathcal{F}$ is an MUS iff $\mathcal{U}$ is a minimal hitting set of* MCSes($\mathcal{F}$).
(2) *A subset $\mathcal{C}$ of $\mathcal{F}$ is an MCS iff $\mathcal{C}$ is a minimal hitting set of* MUSes($\mathcal{F}$).

The reader is referred to [13,56,71] for further details on MUSes and MCSes, as well as the minimal hitting set duality between them.

Additionally, in the context of *partial* MaxSAT, 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. And so the partial MaxSAT problem consists in computing a maximum subset of the soft clauses $\mathcal{R}$ that are satisfiable along with all the clauses of $\mathcal{H}$. 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 $\forall_{c \in \mathcal{R}} w(c) = 1$. Whenever required, a relaxable clause c with weight w (i.e. $w = w(c)$) is denoted by a pair (c, w).

2.3. *Graph problems*

The paper also considers a number of optimization problems in graphs. Given an undirected graph $\mathcal{G} =$

(V, E), an *independent set* (*IS*) of $\mathcal{G}$ 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 a maximum size IS of a graph. This problem 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, a solution of the MIS problem for graph $\mathcal{G}$ also represents, respectively, a solution for the well-known *minimum vertex cover* (*MVC*) of graph $\mathcal{G}$, as well as a *maximum clique* (*MaxCLQ*) of the complemented graph $\mathcal{G}^C$.

2.4. Cardinality-optimality vs. subset-optimality

Regarding the considered optimization problems in propositional logic and also in graphs where one deals with *cardinality-optimal* solutions (i.e. with solutions of the largest/smallest possible size), it is common to refer to these problems with respect to *optimum* (i.e. *maximum/minimum*) solutions, e.g. *maximum* satisfiability, *maximum* independent set, *minimum* vertex cover, *maximum* clique, etc. Alternatively, one can opt to optimize with respect to subset-optimal (i.e. *maximal/minimal*) solutions when it is needed to compute a subset that cannot be increased (or decreased) anymore. For example and regarding MaxSAT, a problem of computing *one* MSS/MCS of a CNF formula is called the *maximal* satisfiability problem. Regarding the mentioned optimization problems in graphs, namely MIS, MVC and MaxCLQ, the problems of computing a subset-maximal independent set, a subset-minimal vertex cover, and a subset-maximal clique are called *maximal independent set* (*MxIS*), *minimal vertex cover* (*MnVC*) and *maximal clique* (*MxCLQ*), respectively.

Note that a subset-optimal solution for any of the considered problems is usually seen as an approximation of the cardinality-optimal solutions and is known to be much easier to compute. As an example, a well-known result is that a maximal independent set (and, hence, a maximal vertex cover as well as a maximal clique) can be computed in linear time [44] while the maximum independent problem is known to be NP-complete [29,43]. Additionally, the topic of enumeration of subset-maximal solutions in different contexts has been extensively studied, e.g. for MaxSAT [10,58,60,64], as well as for MIS [1,41,48,75].

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 5 (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 2. *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 $\mathcal{A}$ as follows: $\mathcal{A}(\mathtt{var}(l)) = \neg l$, $\forall_{l \in \mathcal{U}}$. Then assignment $\mathcal{A}$ falsifies all clauses of $\mathcal{U}$, i.e. $\mathcal{U}$ is all-falsifiable. □

Definition 6 (Maximal falsifiable subset)**.** 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.

Note that analogously to maximal satisfiable subsets (MSSes) in the context of MaxSAT, a formula can have many maximal falsifiable subsets. Thus, one might be interested in enumerating MFSes in order to approximate the solution of the maximum falsifiability problem defined below.

Definition 7 (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 8 (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 3. *$\mathcal{M}$ represents a MaxFalse solution iff $\mathcal{F} \setminus \mathcal{M}$ represents a MinSAT solution.*

Notice that the proof of Proposition 3 is quite trivial and, thus, is omitted here. Nevertheless, Proposition 3 indicates that, in addition to recent algorithms for MinSAT [5,47,50,52,53], 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 9 (Minimal correction (for falsifiability) subset)**.** 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 10 (Minimal non-falsifiable subset)**.** 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:

$$F \triangleq \overset{c_1}{(x_1)} \wedge \overset{c_2}{(\neg x_1)} \wedge \overset{c_3}{(\neg x_1 \vee x_2)} \wedge \overset{c_4}{(\neg x_2)} \wedge \overset{c_5}{(x_3)}.$$

Observe that the sets $\mathcal{M} = \{c_2, c_3, c_5\}$, $\mathcal{C} = \{c_1, c_4\}$ and $\mathcal{N} = \{c_1, c_2\}$ denote, respectively, examples of an MFS, an MCFS and an MNFS. Indeed, all clauses of $\mathcal{M}$ can be falsified by a partial assignment $\mathcal{A} = \{x_1, \neg x_2, \neg x_3\}$. One can immediately notice that in contrast to $\mathcal{M}$, it is not possible to simultaneously falsify all clauses of $\mathcal{M} \cup \{c_1\}$ and $\mathcal{M} \cup \{c_4\}$ because these sets of clauses contain complementary literals either for variable x_1 or x_2. Hence, by Definition 9, the set $\mathcal{C} = \mathcal{F} \setminus \mathcal{M} = \{c_1, c_4\}$ represents a minimal correction subset (i.e. MCFS) of $\mathcal{F}$. Another observation is that clauses c_1 and c_2 cannot be falsified simultaneously (because of variable x_1) but can be falsified separately of each other and, thus, they comprise an example of a minimal non-falsifiable subset (i.e. MNFS) of $\mathcal{F}$. Other examples of MNFSes of $\mathcal{F}$ include the sets $\{c_1, c_3\}$ (because of variable x_1) and $\{c_3, c_4\}$ (because of variable x_2).

Observe that the definitions of MFSes, MCFSes and MNFS presented above in the context of maximal falsifiability are tightly related to the widely used concepts of MSSes, MCSes and MUSes introduced in the area of maximal satisfiability. This connection is shown in Table 1. Whereas the MaxSAT problem consists in computing the maximum number of clauses that are simultaneously satisfied, the MaxFalse problem targets on determining the maximum number of simultaneously falsified clauses. Analogously, the concept of a maximal satisfiable subset (MSS) in MaxSAT corresponds to the concept of a maximal falsifiable subset (MFS) in MaxFalse. The same relation connects the notion of a minimal unsatisfiable subset (MUS) in MaxSAT and the notion of a minimal non-falsifiable subset (MNFS) in MaxFalse as well as correction subsets (MCSes in MaxSAT and MCFSes in MaxFalse).

Table 1
The connection between MaxSAT and MaxFalse/MinSAT

MaxSAT	MaxFalse/MinSAT
Goal: *maximize the number of* ***simultaneously satisfied*** *clauses*	Goal: *maximize the number of* ***simultaneously falsified*** *clauses*
MSS (Maximal **Satisfiable** Subset)	MFS (Maximal **Falsifiable** Subset)
MCS (Minimal Correction (for **Satisfiability**) Subset)	MCFS (Minimal Correction (for **Falsifiability**) Subset)
MUS (Minimal **Unsatisfiable** Subset)	MNFS (Minimal **Non-Falsifiable** Subset)

A relevant result is the relationship between plain maximal falsifiability and maximal independent sets. Given $\mathcal{F}$, let $\mathcal{G}_\mathcal{F} = (V, E)$ be an undirected graph such that each clause of $\mathcal{F}$ is represented by a vertex of $\mathcal{G}_\mathcal{F}$. Moreover, there exists an edge between two vertices in $\mathcal{G}_\mathcal{F}$ iff the corresponding clauses have complemented literals. Clearly (see Proposition 2) clauses with complemented literals *cannot* be simultaneously falsified. Hence, an MFS of $\mathcal{F}$ represents a MxIS of $\mathcal{G}_\mathcal{F}$ and a MxClique of the complemented graph. Moreover, an MCFS corresponds to an MnVC of $\mathcal{G}_\mathcal{F}$. Thus, for plain maximal falsifiability, an MFS can be computed in linear time [44].

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 4. *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 $\mathcal{G}$.*

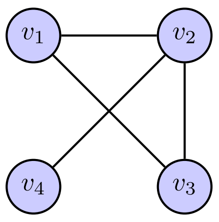
As a result, for the case of plain maximal falsifiability the following holds:

Proposition 5. *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 $\mathcal{G}_\mathcal{F}$ 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 are well known (e.g. [51]). For example, such reductions also allow solving MIS, MVC with MaxSAT 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 $\neg x_e \in c_j$. This *naive* encoding of a graph into a set of clauses gives a way to solve not only the MIS problem with the use of MaxFalse/MinSAT technology but also the MVC and MaxCLQ problems tightly related to MIS. Taking into account the wide range of practical applications of the considered graph problems (e.g. see [2,12,14,18–22,31,40–42,48,65,66,68–70,72–75] and references therein), one can apply MaxFalse/MinSAT algorithm for solving a number of important problems, e.g. including radio network optimization [22], multi-hop wireless network optimization [40], DNA sequence similarity [42], optimization in all-optical networks [70], among many others. The naive encoding of a graph into a set of clauses as well as a number of improvements to this encoding were proposed for dealing with the MIS problem and characterized in detail in [37].

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 Fig. 2(a). 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 $\mathcal{F}$ (Fig. 2(b)) in the following way: $c_1 = x_{v_1,v_2} \vee x_{v_1,v_3}$, $c_2 = \neg x_{v_1,v_2} \vee x_{v_2,v_3} \vee x_{v_2,v_4}$, $c_3 = \neg x_{v_1,v_3} \vee \neg x_{v_2,v_3}$, $c_4 = \neg x_{v_2,v_4}$. Note that an edge of graph $\mathcal{G}$ corresponds to a pair of clauses that have complementary literals, i.e. an MNFS of formula $\mathcal{F}$, while a maximal independent set of $\mathcal{G}$ corresponds to an MFS of $\mathcal{F}$. Moreover, a table showing the correspondence between other well-known graph concepts and their "counterparts" for the CNF encoding is shown in Fig. 2(c).



(a)

$$\mathcal{F} = \left\{ \begin{aligned} c_1 &= x_{v_1,v_2} \vee x_{v_1,v_3}, \\ c_2 &= \neg x_{v_1,v_2} \vee x_{v_2,v_3} \vee x_{v_2,v_4}, \\ c_3 &= \neg x_{v_1,v_3} \vee \neg x_{v_2,v_3}, \\ c_4 &= \neg x_{v_2,v_4} \end{aligned} \right\} \tag{1}$$

(b)

Graph $\mathcal{G}$	Formula $\mathcal{F}$
Edge	MNFS (Minimal Non-Falsifiable Subset)
MIS (*Maximum* IS)	MaxFalse solution
MxIS (*Maximal* IS)	MFS (Maximal Falsifiable Subset)
MVC (*Minimum* VC)	MinSAT solution
MnVC (*Minimal* VC)	MCFS (Minimal Correction (for Falsifiability) Subset)

(c)

Fig. 2. From maximal independent set to maximal falsifiability. (a) Graph $\mathcal{G}$ (see Example 2). (b) Set of clauses $\mathcal{F}$ for graph $\mathcal{G}$. (c) The relation of $\mathcal{F}$ and graph $\mathcal{G}$. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-150685.)

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

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. [53]).

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 11 (Partial weighted maximum falsifiability)**.** Given a formula $\mathcal{F}$, with hard clauses $\mathcal{H}$, $\mathcal{H}$ is satisfiable, and soft clauses $\mathcal{R}$, *Maximum Falsifiabil-*

ity (MaxFalse) denotes the problem of computing the MFS of $\mathcal{F}$ with the largest weight.

Definition 12 (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. see [62]).

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 4 revisits the difference between MFSes and MxISes in the case $\mathcal{H} \neq \emptyset$, and also general minimal hitting results.

As it was mentioned above, a number of encodings of MinSAT to MaxSAT were proposed in the past [5,6,35,47,50,52,53,76]. One of the most efficient encodings of MinSAT is the so-called $\top$-encoding [35]. Given a set of clauses $\mathcal{F}$ it consists in associating each clause $c_i \in \mathcal{F}$ with an auxiliary variable $t_i \in T$ and considering a *new* partial CNF formula $\mathcal{H} \cup \mathcal{R}$, where $\mathcal{H} = \{t_i \equiv c_i \mid c_i \in \mathcal{F}\}$ and $\mathcal{R} = \{\neg t_i \mid t_i \in T\}$. Observe that following the ideas of [35], one can immediately conclude the following proposition holds.

Proposition 6. *Let $\mathcal{F}$ be a set of clauses and* $\top$-Enc$(\mathcal{F})$ *be the $\top$-encoded partial MaxSAT formula of the MinSAT problem for $\mathcal{F}$. The following holds*:

(1) *$\mathcal{N} \subseteq \mathcal{F}$ is an MNFS of $\mathcal{F}$ iff $\mathcal{U} \subseteq$* $\top$-Enc$(\mathcal{F})$ *is an MUS of $\mathcal{F}$ s.t. $|\mathcal{N}| = |\mathcal{U}|$.*
(2) *$\mathcal{C} \subseteq \mathcal{F}$ is an MCFS of $\mathcal{F}$ iff $\mathcal{C}' \subseteq$* $\top$-Enc$(\mathcal{F})$ *is an MCS of $\mathcal{F}$ s.t. $|\mathcal{C}| = |\mathcal{C}'|$.*
(3) *$\mathcal{M} \subseteq \mathcal{F}$ is an MFS of $\mathcal{F}$ iff $\mathcal{S} \subseteq$* $\top$-Enc$(\mathcal{F})$ *is an MSS of $\mathcal{F}$ s.t. $|\mathcal{M}| = |\mathcal{S}|$.*

4. Minimal hitting sets and enumeration problems

One of the most practically important problems in the context of Maximal Satisfiability is enumeration of MUSes of a CNF formula (e.g. see [54,56,67]). One way to enumerate all MUSes of a formula is the method 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 of the complete set of MUSes (MCSes) of the formula (see Proposition 1).

The corresponding theoretical results were considered in [13,71]. Enumeration of MUSes based on enumerating MCSes was done in [11,56,64]. The duality relationship between MCSes and MUSes was also used for solving the SMUS problem in [36,55]. 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 7. *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 8. *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 7 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 8 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 8, $\mathcal{N}'$ 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 6).

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[1] in Section 3.

In contrast to plain maximal falsifiability, for which MNFSes are known to contain exactly two clauses (see Proposition 5), formulas with hard clauses may have MNFSes that contain just one clause. This fact is shown below.

Proposition 9. *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 $\mathcal{H} \models\models \mathcal{W}$, 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 $\mathcal{W} \nsubseteq \mathcal{C}$. 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 9 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 greater 2. Indeed, this can be illustrated with the following example.

Example 3. Consider a set of clauses

$$\mathcal{F}' \triangleq \overset{c_1}{(x_1)} \wedge \overset{c_2}{(x_2)} \wedge \overset{c_3}{(\neg x_1 \vee \neg x_2)}.$$

[1]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*.

Clearly, this plain CNF formula is unsatisfiable (moreover, it is an MUS). By $\top$-encoding $\mathcal{F}'$ with the use of auxiliary variables $\{t_1, t_2, t_3\}$, one can transform $\mathcal{F}'$ and get a new partial CNF formula $\mathcal{F} = \mathcal{H} \cup \mathcal{R}$, where $\mathcal{H} = \{t_i \equiv c_i \mid c_i \in \mathcal{F}\}$ and $\mathcal{R} = \{\neg t_1, \neg t_2, \neg t_3\}$. Observe that formula $\mathcal{F}$ has exactly one MNFS $\mathcal{U}$, which corresponds to the MUS of $\mathcal{F}'$ (see Proposition 6) and $\mathcal{U} = \mathcal{R}$, i.e. $|\mathcal{U}| = 3$.

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 5) and use them to compute a lower bound for bootstrapping the algorithms for MaxFalse.

Algorithm 1. Basic linear search (BLS)

```
1  Function BLS($\mathcal{F} = \mathcal{H} \cup \mathcal{R}$)
2      (st, $\mathcal{A}$) ← SAT($\mathcal{H}$)  # initial SAT call
3      if st = false then
4          return (false, $\emptyset$)
5      $\mathcal{C} \leftarrow \mathcal{R}$
6      HardenFalsified($\mathcal{H}, \mathcal{C}, \mathcal{A}$)
7      foreach $c \in \mathcal{C}$ do
8          (st, $\mathcal{A}$) ← SAT($\mathcal{H} \cup \{\neg c\}$)
9          if st = true then
10             $\mathcal{H} \leftarrow \mathcal{H} \cup \{\neg c\}$
11             $\mathcal{C} \leftarrow \mathcal{C} \setminus \{c\}$
12             HardenFalsified($\mathcal{H}, \mathcal{C}, \mathcal{A}$)
13     return (true, $\mathcal{R} \setminus \mathcal{C}$)
```

5. 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 (i.e. when a partial CNF formula $\mathcal{F} = \mathcal{H} \cup \mathcal{R}$ is considered) is NP-hard. Since there are several encodings of MinSAT into MaxSAT (e.g. [35,50,76]), these encodings can be also used for solving the maximal falsifiability problem. For example, when $\top$-encoding a CNF formula $\mathcal{F}$, one approach for finding an MFS of formula $\mathcal{F}$ is to find an MSS (or its complement – MCS) of its $\top$-encoded formula $\top$-Enc($\mathcal{F}$), e.g. with recently proposed algorithms for computing MCSes [10,58,60,64]. Nevertheless, this paper proposes instead *native* algorithms for both maximal and maximum falsifiability.

5.1. *Basic linear search*

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 [58,64]. Given $\mathcal{H}$ 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 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 `HardenFalsified`($\mathcal{H}, \mathcal{C}, \mathcal{A}$) (lines 6 and 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 `HardenFalsified`($\mathcal{H}, \mathcal{C}, \mathcal{A}$) can significantly reduce the number of SAT calls.

5.2. *Clause D-like algorithm*

Additionally, inspired by the results presented in [58], one can construct an algorithm for computing MFSes based on the ideas lying behind the *clause D algorithm* (*CLD*). The original CLD algorithm [58] proposed for computing an MCS of a CNF formula $\mathcal{F} = \mathcal{H} \cup \mathcal{R}$ aims at reducing the number of calls to the SAT oracle, which is done by considering a disjunction of the original soft clauses of the formula, i.e. $D = \{\bigvee_{c_i \in \mathcal{R}} c_i\}$ instead of $\mathcal{R}$, and testing if $\mathcal{H} \cup D$ is satisfiable. If it is, then its model returned by the SAT oracle is used to refine the clause D by removing all the original clauses $c_i \in \mathcal{R}$ that are satisfied by the model. At the end, the CLD algorithm returns an MCS of formula $\mathcal{F}$ in the form of the refined clause D.

The same idea (however, with a serious drawback) can be applied to the problem of computing an MFS of a CNF formula. Algorithm 2 shows the pseudo-code of the CLD-like algorithm adapted to maximal falsifiability. Given a formula in the form of $\mathcal{H}$ and $\mathcal{R}$, Algorithm 2 computes its MFS in the following way. First, it encodes the negation of each soft (relaxable) clause c_i with a fresh auxiliary variable t_i and adds the corresponding clauses into $\mathcal{H}$. Note that now instead of the set of soft clauses $\mathcal{R}$, we consider a new set of soft clauses $\mathcal{C}$, which are unit clauses over the auxiliary variables $\{t_i\}$. The clause comprising all literals of $\mathcal{C}$, i.e. $\{\bigvee_{t_i \in \mathcal{C}} t_i\}$, acts as the clause D in the original algorithm CLD (see line 11). After this, Algorithm 2 checks if the hard part $\mathcal{H}$ of the formula is satisfiable. If it is not, the algorithm returns an empty MFS. Otherwise, the main loop starts, which refines $\mathcal{C}$ until formula $\mathcal{H} \cup \{\bigvee_{t_i \in \mathcal{C}} t_i\}$ gets unsatisfiable. It should be noted that at each iteration of the loop, $\mathcal{C}$ is refined by hardening each clause $\{t_i\}$ satisfied by model $\mathcal{A}$ (meaning that the original clause c_i s.t. $t_i \equiv \neg c_i$ is falsified by $\mathcal{A}$) and removing it from $\mathcal{C}$. The algorithm stops when it is not possible to satisfy clause D, i.e. $\{\bigvee_{t_i \in \mathcal{C}} t_i\}$, along with the hard part $\mathcal{H}$. Note that although the CLD algorithm is intended to reduce the number of calls to the SAT oracle and, thus, increase the performance of MFS extraction, Algorithm 2 has a significant drawback, which is the use of additional auxiliary variables. The algorithm has to encode all clauses of $\mathcal{R}$ introducing $|\mathcal{R}|$ new variables and, thus, making calls to the SAT oracle harder. For formulas with a large number of relaxable clauses this can be a significant drawback, which will be indeed confirmed by the experimental results (see Section 7).

Algorithm 2. Clause D-like algorithm (CLD)

```
1  Function CLD($\mathcal{F} = \mathcal{H} \cup \mathcal{R}$)
2      $\mathcal{C} \leftarrow \emptyset$
3      foreach $c_i \in \mathcal{R}$ do
4          $\mathcal{C} \leftarrow \mathcal{C} \cup \{t_i\}$
5          $\mathcal{H} \leftarrow \mathcal{H} \cup \{t_i \equiv \neg c_i\}$
6      (st, $\mathcal{A}$) $\leftarrow$ SAT($\mathcal{H}$)
7      if st = false then
8          return (false, $\emptyset$)
9      repeat
10         HardenFalsified($\mathcal{H}, \mathcal{C}, \mathcal{A}$)
11         (st, $\mathcal{A}$) $\leftarrow$ SAT($\mathcal{H} \cup \{\bigvee_{t_i \in \mathcal{C}} t_i\}$)
12     until st = false
13     return (true, $\mathcal{R} \setminus \{c_i \in \mathcal{R} \mid \neg c_i \equiv t_i, t_i \in \mathcal{C}\}$)
```

6. Algorithms for maximum falsifiability

A solution to the MaxFalse problem can be obtained by computing a solution to the MinSAT problem (Proposition 3). On the other hand, and as mentioned in Section 5, several encodings have been proposed to translate MinSAT into MaxSAT. Thus, the MaxFalse problem can be solved by encoding the problem into MaxSAT and solving the corresponding MaxSAT problem. This paper proposes instead *native* algorithms for the MaxFalse problem including three iterative algorithms and an algorithm based on the minimal hitting set duality.

6.1. Iterative algorithms

The three algorithms proposed in this section are based on iterative calls to 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 a relaxation variable to the clause (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. At each iteration, the SAT solver deals with the working formula and a constraint enforcing 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).

All the iterative algorithms start by obtaining a working formula $\mathcal{F}_w$ by relaxing the soft clauses in $\mathcal{R}$ together with all the hard clauses (lines 2–6 of the pseudo-codes of the Algorithms 3, 4, 5). In this case,

Algorithm 3. MaxFalse Binary Search (MFBS)

```
1  Function MFBS($\mathcal{F} = \mathcal{H} \cup \mathcal{R}$)
2      $(\mathcal{F}_w, R, W, last\mathcal{A}) \leftarrow (\mathcal{H}, \emptyset, \emptyset, \emptyset)$
3      foreach $(c, w) \in \mathcal{R}$ do
4          $R \leftarrow R \cup \{r\}$
5          $W \leftarrow W \cup \{w\}$
6          $\mathcal{F}_w \leftarrow \mathcal{F}_w \cup \{c \rightarrow r\}$
7      $(\lambda, \mu) \leftarrow$ (ComputeLB($\mathcal{R}$), ComputeUB($\mathcal{F}_w$))
8      while $\lambda \neq \mu$ do
9          $\kappa \leftarrow \lfloor \frac{\lambda+\mu}{2} \rfloor$
10         (st, $\mathcal{A}$) $\leftarrow$ SAT($\mathcal{F}_w \cup$ CNF($\sum_{(w,r)\in W\times R} w \cdot r \leqslant \kappa$))
11         if st = true then
12             $(last\mathcal{A}, \mu) \leftarrow$ ($\mathcal{A}$, GetSolution($\mathcal{R}, \mathcal{A}$))
13         else
14             $\lambda \leftarrow$ SubSetSum($W, \kappa$)
15     return Falsified($\mathcal{R}, last\mathcal{A}$)
```

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

```
1  Function MFLSSU($\mathcal{F} = \mathcal{H} \cup \mathcal{R}$)
2      $(\mathcal{F}_w, R, W, \mathcal{A}) \leftarrow (\mathcal{H}, \emptyset, \emptyset, \emptyset)$
3      foreach $(c, w) \in \mathcal{R}$ do
4          $R \leftarrow R \cup \{r\}$
5          $W \leftarrow W \cup \{w\}$
6          $\mathcal{F}_w \leftarrow \mathcal{F}_w \cup \{c \rightarrow r\}$
7      $(\mu, \mathcal{A}) \leftarrow$ ComputeUB($\mathcal{F}_w$)
8      st $\leftarrow$ true
9      while st = true do
10         (st, $\mathcal{A}$) $\leftarrow$ SAT($\mathcal{F}_w \cup$ CNF($\sum_{(w,r)\in W\times R} w \cdot r < \mu$))
11         if st = true then
12             $\mu \leftarrow$ GetSolution($\mathcal{R}, \mathcal{A}$)
13     return Falsified($\mathcal{R}, \mathcal{A}$)
```

the relaxation of the soft clauses is not the usual relaxation as in MaxSAT. Instead, the algorithms follow the relaxation of soft clauses as in the Model-Guided algorithm [35] for MinSAT. A fresh relaxation variable r is associated with the original soft clause c, and a set

Algorithm 5. MaxFalse Linear Search UNSAT-SAT (MFLSUS)

```
1  Function MFLSUS($\mathcal{F} = \mathcal{H} \cup \mathcal{R}$)
2      $(\mathcal{F}_w, R, W, \mathcal{A}) \leftarrow (\mathcal{H}, \emptyset, \emptyset, \emptyset)$
3      foreach $(c, w) \in \mathcal{R}$ do
4          $R \leftarrow R \cup \{r\}$
5          $W \leftarrow W \cup \{w\}$
6          $\mathcal{F}_w \leftarrow \mathcal{F}_w \cup \{c \rightarrow r\}$
7      $\lambda \leftarrow$ ComputeLB($\mathcal{R}$)
8      st $\leftarrow$ false
9      while st = false do
10         (st, $\mathcal{A}$) $\leftarrow$ SAT($\mathcal{F}_w \cup$ CNF($\sum_{(w,r)\in W\times R} w \cdot r \leqslant \lambda$))
11         if st = false then
12             $\lambda \leftarrow$ SubSetSum($W, \lambda$)
13     return Falsified($\mathcal{R}, \mathcal{A}$)
```

of binary clauses $\{c \rightarrow r\}$ is added to the working formula, each containing the negation of one literal of the soft clause, and the associated relaxation variable.

After relaxing, the bounds are computed (line 7 of the pseudo-codes of the Algorithms 3, 4 and 5). In the following we explain how the bounds are computed followed by a detailed description of each of the iterative algorithms.

6.1.1. Upper and lower bounds

The iterative algorithms proceed by refining an upper bound or a lower bound or both, depending on the algorithm. This section describes how the initial bounds are obtained from the relaxed working formula $\mathcal{F}_w$ or from the set of soft clauses $\mathcal{R}$.

In order to compute an upper bound, the algorithm calls a SAT solver on the working formula $\mathcal{F}_w$ with polarities set to false for the relaxation variables, i.e. the relaxation variables are preferred to be falsified. A similar method of using preferred polarities of a formula's variables in SAT solving was used e.g. in [27,32] for computing a maximal/minimal model of the formula. If the assignment returned by the SAT solver (assuming the hard clauses are satisfied) satisfies a relaxation variable, then the corresponding original clause is satisfied by the assignment. As such, the sum of weights of the clauses associated to the relaxation variables set to true corresponds to the upper bound.

The lower bound is computed by a greedy heuristic using only the soft clauses in $\mathcal{R}$. Initially the lower bound is assumed to be 0, and the following process is repeated. In each iteration, for each variable, two sums are obtained: 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. Among the two sums, the minimum value is associated to the variable as the minimum weight that we are forced to pay due to an assignment to the variable. Any assignment to the variable will satisfy (zero or more) soft clauses, whose cost is at least the associated minimum value.

Afterwards, the variable with the maximum associated cost is selected. Since any assignment is forced to include the costs associated to the variables, then we greedily select the one with the biggest associated cost. The cost is added to the lower bound, and all the clauses corresponding to the minimum sum associated to the selected variable are deleted. The selected variable is then disregarded in the remaining iterations. The process is repeated until there are no more variables. In the end, a lower bound has been computed.

Example 4. Consider the weighted formula $F \triangleq (c_1, w_1) \wedge (c_2, w_2) \wedge (c_3, w_3)$, where:

$$(c_1, w_1) \triangleq (x_1 \vee x_2 \vee x_3, 1),$$
$$(c_2, w_2) \triangleq (x_1 \vee \neg x_2, 5),$$
$$(c_3, w_3) \triangleq (\neg x_1 \vee \neg x_2 \vee x_3, 3).$$

For the first iteration on the computation of the lower bound, each of the variables is associated with the following values:

$$x_1 \rightsquigarrow \min(6, 3) = 3,$$
$$x_2 \rightsquigarrow \min(1, 8) = 1,$$
$$x_3 \rightsquigarrow \min(4, 0) = 0.$$

The maximum value of 3 is selected with the negative polarity of variable x_1. Then the lower bound is now updated to 3 and clause (c_3, w_3) is deleted from the formula. Now the associated values to the remaining variables are updated to:

$$x_2 \rightsquigarrow \min(1, 5) = 1,$$
$$x_3 \rightsquigarrow \min(1, 0) = 0.$$

The maximum corresponds to 1, increasing the lower bound to 4, and variable x_2 is selected with the positive polarity. Clause (c_1, w_1) is deleted from the formula. The associated value to variable x_3 is updated to $x_3 \rightsquigarrow \min(0, 0) = 0$.

Finally, variable x_3 is selected, although not increasing the lower bound. Thus, the final lower bound computed is 4.

6.1.2. *Iterative binary and linear search algorithms*

This section presents the pseudo-codes of the three proposed iterative MaxFalse algorithms. First, the pseudo-code of the binary search algorithm is shown, followed by the pseudo-code of the linear search algorithms.

Algorithm 3 shows the pseudo-code of the *MaxFalse Binary Search* (*MFBS*) algorithm for maximum falsifiability. As mentioned in the introduction of Section 6.1, iterative algorithms start by relaxing all the soft clauses in a working formula $\mathcal{F}_w$ together with the hard clauses (lines 2–6). Based on the working formula $\mathcal{F}_w$ and the set of original soft clauses $\mathcal{R}$, both the upper and lower bounds are computed in line 7, as described in Section 6.1.1.

Lines 8–14 present the main loop of the MFBS algorithm. At 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 [4].

The algorithm iterates until both bounds are the same (line 8), and returns a set of soft clauses falsified by the last assignment with function `Falsified`() in line 15.

Algorithm 4 shows the pseudo-code of the *MaxFalse Linear Search SAT-UNSAT* (*MFLSSU*) algorithm for maximum falsifiability, while Algorithm 5 shows the pseudo-code of the *MaxFalse Linear Search UNSAT-SAT* (*MFLSUS*) algorithm for maximum falsifiability. Since the pseudo-codes are similar, they are presented together.

In contrast to binary search where both upper and lower bounds are refined simultaneously, the idea of the linear search algorithms consists in refining only one of the bounds, either the upper bound in case of the Linear Search SAT-UNSAT, or the lower bound in case of the Linear Search UNSAT-SAT. Depending on where the optimum solution is located with regards to the bounds, refining only one of the bounds can be advantageous. For example, linear search MaxSAT algorithms refining lower bounds have demonstrated good performances in industrial categories in previous MaxSAT Evaluations.

As before, lines 2–7 obtain the relaxed working formula and compute the initial bounds. An upper bound is computed in the case of MFLSSU, and a lower bound is computed in the case of MFLSUS.

The main loop is presented in lines 8–13. At each iteration, line 10 makes a call to the SAT solver on the working formula $\mathcal{F}_w$ together with a constraint enforcing the maximum allowed cost encoded into CNF. Depending on the outcome of the SAT solver, then the bounds are refined in lines 11–12.

In the case of MFLSSU, since we are refining the upper bound, the maximum allowed cost is smaller than the current upper bound. The SAT solver is testing if there exists a solution smaller than the current one. If true, the SAT solver will return the new solution found and the upper bound is updated to the new value (similar to the refinement of the upper bound in the binary search algorithm (line 12 of MFBS)). MFLSSU stops when no better solution can be found.

In the case of MFLSUS, since we are refining the lower bound, the maximum allowed cost is at most the current lower bound. The SAT solver is testing if there exist a solution with a cost lower or equal to the current lower bound. If no solution exists, then the lower bound is increased to the next possible value (similar to the refinement of the lower bound in the binary search algorithm (line 14 of MFBS)). MFLSUS stops when a solution has been found, that is the current lower bound corresponds to the optimum value.

6.2. *Algorithm based on minimal hitting set duality*

Following the ideas described in Section 4, this section proposes an algorithm for the maximum falsifiability problem based on the minimal hitting set duality between sets of MCFSes and MNFSes of a CNF formula. A similar approach to MaxSAT (called MaxHS) was proposed in [24] and further improved in [25,26]. However, in contrast to MaxHS, which builts on top of the well-known MIP solver CPLEX, the algorithm proposed here is a pure SAT-based approach to the problem meaning that hitting sets are enumerated using a pure SAT-based technology without appealing to any of the integer programming algorithms. Also note that instead of computing minimal hitting sets of the set of all MNFSes, the approach being proposed enumerates minimal hitting sets of the set of unfalsifiable cores of the formula.

Algorithm 6 shows the pseudo-code of the proposed algorithm. The basic idea of the method is to divide the process into two parts. The first one makes use of an external oracle that enumerates smallest minimal hitting sets of the set of all unfalsifiable cores of the formula found so far. The second part is intended for checking if the set of clauses complement to what was found by the external oracle is all-falsifiable. If it is not then a new unsatisfiable core is extracted and the process continues. Otherwise, a solution is found and reported. In order to compute the smallest minimal hitting sets of the unfalsifiable cores found so far, a MaxSAT solver is used as the external oracle on a constructed partial MaxSAT formula $\mathcal{Q}$. Given a formula $\mathcal{F} = \mathcal{H} \cup \mathcal{R}$, the algorithm first creates a *picking* variable p_i for each clause c_i of $\mathcal{R}$ (see line 2) and adds the corresponding soft clause (p_i, w_i) to $\mathcal{Q}$ (line 3). The hard part of $\mathcal{Q}$ is initially empty. The main loop of the algorithm starts at line 4. At each iteration of the loop, the algorithm chooses a different subset of $\mathcal{R}$ and checks its all-falsifiability together with the hard part

Algorithm 6. MaxFalse algorithm based on Hitting Sets (MFHS)

```
1  Function MFHS($\mathcal{F} = \mathcal{H} \cup \mathcal{R}$)
2      $P \leftarrow \{p_i \mid (c_i, w_i) \in \mathcal{R}\}$
3      $\mathcal{Q} \leftarrow \{(p_i, w_i) \mid (c_i, w_i) \in \mathcal{R}, p_i \in P\}$
4      while true do
5          (st, $\mathcal{A}_M$) $\leftarrow$ MAXSAT($\mathcal{Q}$)
6          if st = true then
7              $(\alpha, \mathcal{F}_Q) \leftarrow (\emptyset, \mathcal{H})$
8              foreach $p_i \in P$ s.t. $\mathcal{A}_M(p_i) = $ true do
9                  $\alpha \leftarrow \alpha \cup \{p_i\}$
10                 $\mathcal{F}_Q \leftarrow \mathcal{F}_Q \cup \{p_i \to \neg c_i \mid c_i \in \mathcal{R}\}$
11             (st, $\mu_C$, $\mathcal{A}_S$) $\leftarrow$ SAT($\mathcal{F}_Q, \alpha$)
12             if st = true then
13                 return Falsified($\mathcal{R}, \mathcal{A}_S$)
14             else
15                 $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\bigvee_{p_i \in \mu_C} \neg p_i, \top)\}$
16         else
17             return $\emptyset$
```

$\mathcal{H}$ by doing a SAT call at line 11. Note that the SAT oracle receives formula $\mathcal{F}_Q$ and a set of assumptions α, both determined by a MaxSAT solution $\mathcal{A}_M$ of $\mathcal{Q}$ reported by the external oracle (see line 5). Each variable assigned true by $\mathcal{A}_M$ *activates* the corresponding clause of $\mathcal{R}$. This is done by adding a set of clauses $\{p_i \rightarrow \neg c_i\}$ to $\mathcal{F}_Q$ and an assumption literal p_i to α. If the SAT call returns true meaning that the selected subset of $\mathcal{R}$ is all-falsifiable together with $\mathcal{H}$, then the algorithm reports the solution and stops.[2] Otherwise, an unsatisfiable core μ_C of $\mathcal{F}_Q$ (which is also a new core of $\mathcal{F}$) is extracted and the corresponding blocking clause is added to $\mathcal{Q}$. Note that in this case, the next MaxSAT model of $\mathcal{Q}$ has to *hit* the new core. This means that there will be at least one negative literal $\neg p_i \in \mathcal{A}_M$ *neutralizing* the new core. The algorithm iterates until it either finds a solution for the MaxFalse problem, or proves that it does not exist.

7. Experimental results

This section describes the experimental results obtained for Maximal Falsifiability as well as for Maximum Falsifiability. Section 7.1 shows a comparison on the quality of the solutions obtained for Maximal Falsifiability. Then Section 7.2 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. The following two sets of benchmarks were considered in the evaluation. The first set, called *MIS benchmarks*, comprises all the MaxFalse instances that were considered in [37]. These instances are crafted and come from the MIS and MaxClique problems, which can be reduced to MaxFalse (the corresponding reductions were studied in [37]). The total number of instances in the MIS benchmark set is 233.

In order to evaluate the performance of the algorithms in *real* industrial problems (not random nor crafted problems), a second benchmark set was also considered. It comprises all the Partial MaxSAT Industrial and Weighted Partial Industrial benchmarks from the MaxSAT Evaluation 2014.[3] This benchmark set is called *MaxSAT benchmarks*. The benchmarks in this set were transformed into MaxFalse instances by selecting the ones that only contained unit soft clauses and by negating the unit literals on those instances. Due to this choice of benchmark instances, none of the approaches (neither MaxSAT nor MaxFalse) has to deal with *"encoded"* instances. A total of 877 MaxFalse instances were obtained.

As explained in Section 5, a different alternative to Maximal/Maximum Falsifiability consists in transforming the MaxFalse instance by encoding it into MaxSAT (e.g. via the $\top$-encoding), and then computing an MCS/MaxSAT solution of the resulting MaxSAT instance. In our experimental evaluation the $\top$-encoding was used for deriving MaxSAT instances for the MIS benchmarks. As for the MaxSAT benchmarks, the original instances considered contain only *unit* soft clauses, and thus there is no need to do the $\top$-encoding. Instead, it is enough to negate the corresponding literals in the soft clauses to get MaxSAT instances from MaxFalse, and the other way around.

7.1. Maximal falsifiability

The BLS and CLD algorithms (Algorithms 1 and 2) proposed in Section 5 were implemented in a tool called *mxlFalse*. The underlying SAT solver of the mxlFalse tool is Minisat 2.2 [28].

This section studies the quality of the solutions obtained for both sets of benchmarks. The cost of the MFSes/MCFSes obtained with BLS and CLD algorithms is compared against the cost of the MCSes obtained by MCSls [58] (an efficient MCS extractor) that was ran for the corresponding MaxSAT instances. Note that we used a number of possible configurations of MCSls and chose the best one, which uses the original CLD algorithm proposed in [58].

In the experiments both the mxlFalse and MCSls were set to enumerate MFSes/MCFSes and MCSes (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 Figs 3 and 4. Figure 3 compares the values obtained by the proposed BLS (on the left) and CLD (on the right) algorithms for the MIS benchmarks instances to the value obtained by MCSls. It can be seen from the scatter plot that in the vast majority of cases all the considered algorithms are able to find the exact optimum while enumerating approximate solutions. As for the others, the best solution reported by all competitors is usually the same. There are

[2]Note that maximality of the solution is guaranteed by maximality of solutions for $\mathcal{Q}$.

[3]See http://www.maxsat.udl.cat/14/benchmarks/.

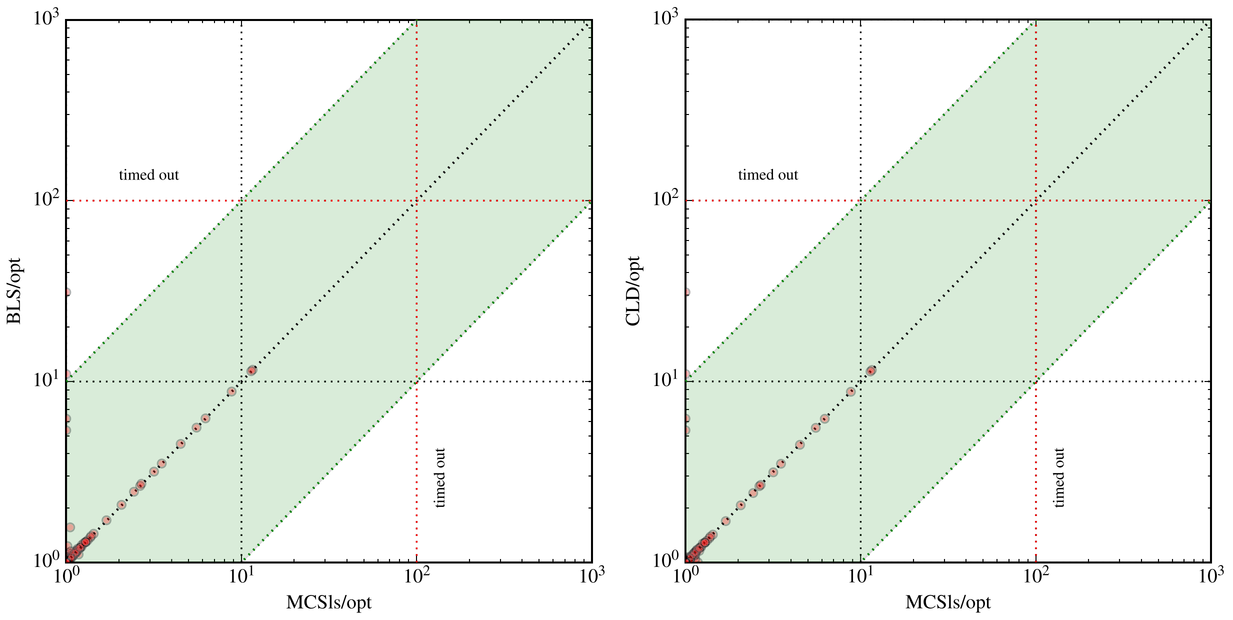


Fig. 3. Scatter plots comparing BLS and CLD algorithms to MCSls for the MIS benchmark set. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-150685.)

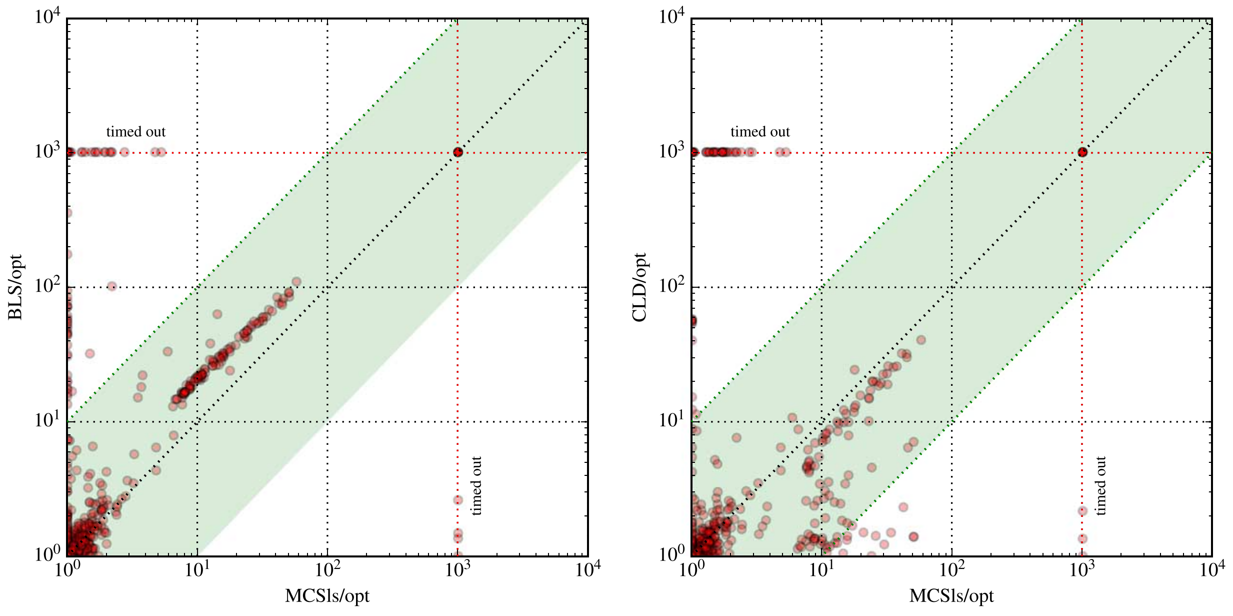


Fig. 4. Scatter plots comparing BLS and CLD algorithms to MCSls for the MaxSAT benchmark set. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-150685.)

a few outlying instances where MCSls is better than both BLS and CLD algorithms.

Figure 4 shows the comparison between the best solutions found by the proposed algorithms and MCSls for the considered MaxSAT benchmark set. In this case, even though there are instances where the BLS algorithm gets closer to the optimal solution than MCSls, in the majority of benchmarks solutions produced by MCSls are better than the ones of the BLS algorithm. As for the comparison between CLD and MCSls, apparently there is no clear winner. A number of instances where CLD was not able to find any approx-

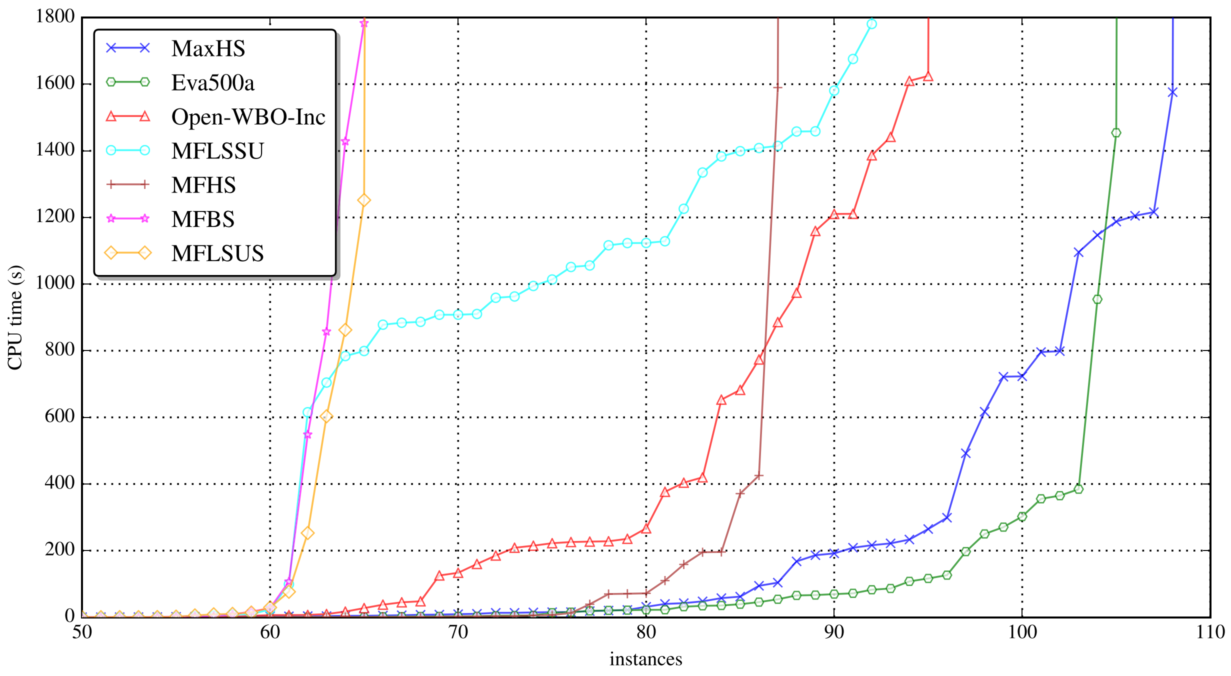


Fig. 5. Cactus plot for MIS benchmark instances. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-150685.)

imate solution is larger than the number of instances for which MCSls got timed out. As it was mentioned in Section 5.2, this can be explained by the fact that the proposed CLD algorithm has to introduce a lot of additional variables (one variable per soft clause), which makes SAT calls harder. However, as one can see at Fig. 4, for a wide majority of formulas with a reasonable number of relaxable clauses CLD is able to enumerate solutions closer to the optimum than the ones found by MCSls. Therefore, in most of the cases solutions obtained by CLD are preferred.

7.2. *Maximum falsifiability*

The Maximum Falsifiability algorithms proposed in Section 6 were implemented in a tool called *maximumFalse*. Namely, the following algorithms were implemented: binary search (MFBS), linear search unsat-sat (MFLSUS) and linear search sat-unsat (MFLSSU) as well as the minimal hitting set based algorithm (MFHS). This section presents results on the performance of the proposed algorithms running for 1800 seconds with 4 GB of memory limit. The underlying SAT solver of the maximumFalse tool is the Glucose SAT solver [7]. Additionally, best non-portfolio MaxSAT solvers (according to the results of the MaxSAT Evaluation 2014) were considered, namely Open-WBO-Inc [59] and Eva500a [63].[4] Also MaxHS, which is a well-known MaxSAT solver based on the hitting set enumeration paradigm, was considered.

A cactus plot showing the performance of the evaluated algorithms on the MIS benchmark instances is presented in Fig. 5. The largest number of instances (108) for this set of benchmarks is solved by MaxHS. This is not surprising because MaxHS is known to perform extremely well for random and crafted MaxSAT benchmarks. Eva500a is 3 instances behind MaxHS (with 105 instances solved). Open-WBO-Inc comes third being able to cope with 95 instances. Among the proposed MaxFalse algorithms the best performance is shown by linear search sat-unsat (MFLSSU), which solved 92 instances, while the linear search unsat-sat (MFLSUS) coming the last (65 instances solved). This is caused by the nature of the benchmarks – optimal values of the majority of them are close to the initial upper bounds. Being able to solve 87 instances, the MFHS algorithm is fifth. A possible explanation of this can be the lack of additional heuristics and improvements in this algorithm, e.g. widely used disjoint core enumeration and core trimming, as well as a number

[4]Open-WBO-Inc is the best non-portfolio solver in the category of MaxSAT Industrial and Partial MaxSAT Industrial while Eva500a outperforming every other solver in the category of Weighted Partial MaxSAT Industrial (see http://www.maxsat.udl.cat/14/results/).

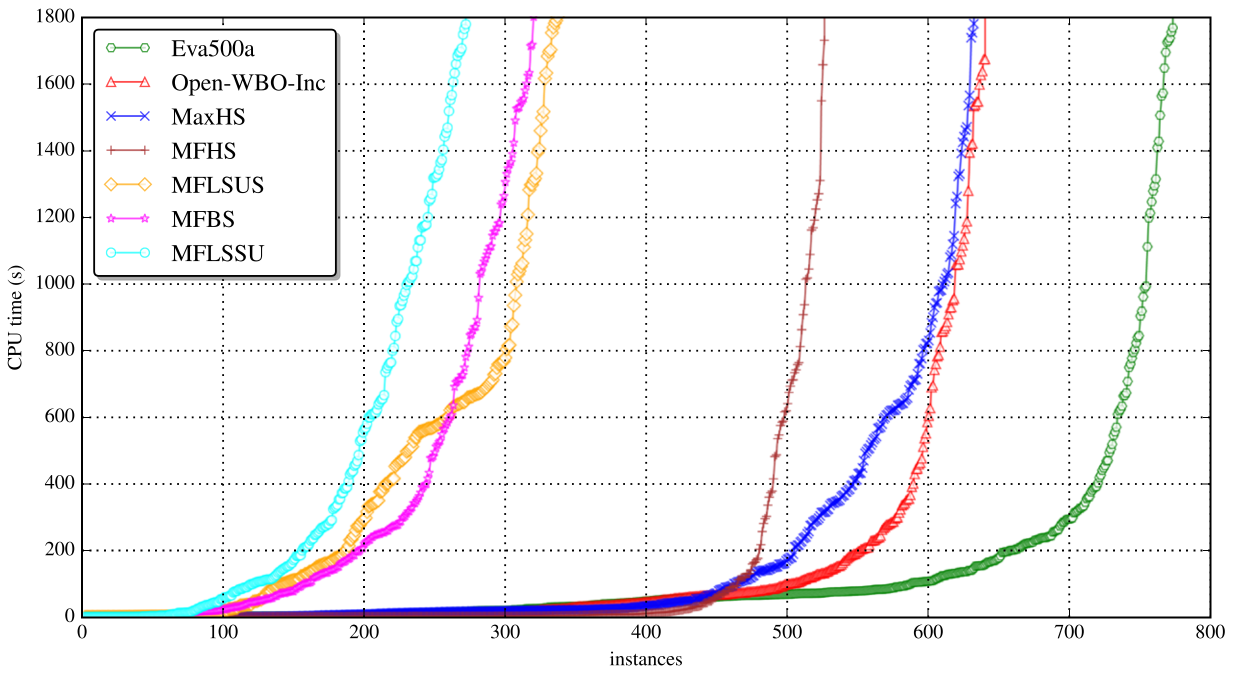


Fig. 6. Cactus plot for MaxSAT benchmark instances. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-150685.)

of efficient heuristics proposed in [24–26] specifically for hitting set enumeration based approaches and implemented in MaxHS.

Figure 6 shows the performance of the tested algorithms on the MaxSAT benchmark instances. As expected, state-of-the-art core-guided MaxSAT algorithms outperform the other competitors. Eva500a shows the best performance with 772 solved instances (out of 877). Open-WBO-Inc comes second but it is far behind Eva500a with 639 solved instances. The worst performance is shown by iterative MaxFalse algorithms (271, 319 and 334 instances solved by MFLSSU, MFBS and MFLSUS, respectively). The MFLSUS algorithm is the best among them indicating that for this benchmark set either optimal values are close to initial lower bounds or the calls to the SAT oracle with the SAT outcome are harder than the UNSAT calls. For this set of benchmarks, the MFHS algorithm performs far better than all the iterative algorithms being able to solve 525 instances. Although MaxHS has a better performance (631 instances solved), it is expected because of several reasons. First, it uses a different *solving engine* utilizing the best known MIP solver called CPLEX. In contrast, our approach is a pure SAT-based algorithm. Second (and the most important), MaxHS is being developed and heavily improved for several years resulting its performance to increase dramatically over the years. Therefore, being a basic hitting set based algorithm for the MaxFalse problem, MFHS still has potential to be improved in a manner similar to what was already done for MaxHS.

As a result, the experimental evaluation performed reveals the lack of efficient native algorithms for the MaxFalse problem. This applies to both crafted and industrial problem instances. However, while the performance of the proposed MFHS algorithm can be improved in order to be as good as MaxHS is for the crafted MaxSAT instances, for the industrial problem instances the development of efficient native core-guided algorithms is of a crucial importance. Note that although core-guided MaxSAT algorithms result from over a decade of research, it is still not known in the literature how to take advantage of core-guided search natively in MaxFalse algorithms, nor it is known to be even possible. While this paper proposes a general framework for both maximal and maximum falsifiability, the mentioned issues are important topics of future research in the MaxFalse setting.

8. Conclusions

Motivated by the recent interest in MinSAT, this paper develops a comprehensive characterization of this

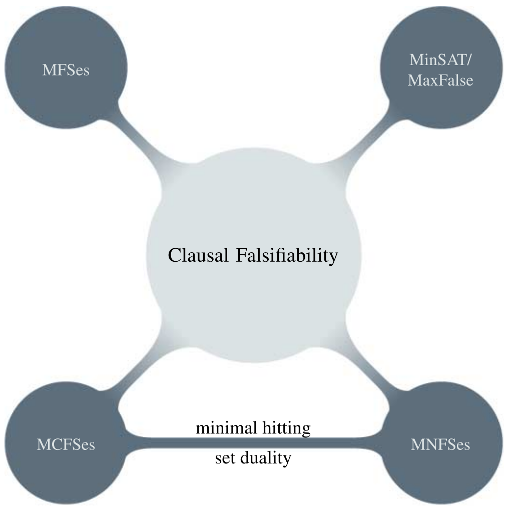


Fig. 7. Clausal falsifiability problems characterization. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-150685.)

problem, which follows the one developed earlier for MaxSAT (see Fig. 7). 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. The proposed reduction brings a number of important practical applications of the proposed maximum and maximal falsifiability framework including the well-known graph-related problems, as well as many others. 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 preliminary experimental results indicate that the proposed algorithms for maximal falsifiability can compete with the state-of-the-art algorithms enumerating MCSes in terms of both the running time and the solution quality. As for the maximum falsifiability, the paper issues the challenge of developing efficient native algorithms for MaxFalse including native core-guided algorithms.

The work described in the paper opens a significant number of research directions. Concrete examples include additional native algorithms for computing MFSes and for MaxFalse, as well as improved versions of the proposed algorithms, among others.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve earlier versions of the paper. This work is partially supported by SFI PI grant BEACON (09/IN.1/I2618), FCT grant POLARIS (PTDC/EIA-CCO/123051/2010) and national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

References

[1] E.A. Akkoyunlu, The enumeration of maximal cliques of large graphs, *SIAM J. Comput.* **2**(1) (1973), 1–6.
[2] D.V. Andrade, M.G.C. Resende and R.F. Werneck, Fast local search for the maximum independent set problem, *J. Heuristics* **18**(4) (2012), 525–547.
[3] E. Angel, E. Bampis and L. Gourvès, On the minimum hitting set of bundles problem, *Theor. Comput. Sci.* **410**(45) (2009), 4534–4542.
[4] C. Ansótegui, M.L. Bonet and J. Levy, A new algorithm for weighted partial MaxSAT, in: *AAAI Conference on Artificial Intelligence*, 2010.
[5] C. Ansotegui, C.M. Li, F. Manya and Z. Zhu, A SAT-based approach to MinSAT, in: *International Conference of the Catalan Association for Artificial Intelligence*, 2012, pp. 185–189.
[6] J. Argelich, C.-M. Li, F. Manya and Z. Zhu, MinSAT versus MaxSAT for optimization problems, in: *International Conference on Principles and Practice of Constraint Programming*, 2013.
[7] G. Audemard and L. Simon, Predicting learnt clauses quality in modern sat solvers, in: *International Joint Conference on Artificial Intelligence*, 2009, pp. 399–404.
[8] A. Avidor and U. Zwick, Approximating MIN k-SAT, in: *International Symposium on Algorithms and Computation*, 2002, pp. 465–475.
[9] A. Avidor and U. Zwick, Approximating MIN 2-SAT and MIN 3-SAT, *Theory Comput. Syst.* **38**(3) (2005), 329–345.
[10] F. Bacchus, J. Davies, M. Tsimpoukelli and G. Katsirelos, Relaxation search: A simple way of managing optional clauses, in: *AAAI Conference on Artificial Intelligence*, 2014, pp. 835–841.

[11] J. Bailey and P.J. Stuckey, Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization, in: *International Symposium on Practical Aspects of Declarative Languages*, 2005, pp. 174–186.

[12] R. Battiti and M. Protasi, Reactive local search for the maximum clique problem, *Algorithmica* **29**(4) (2001), 610–637.

[13] E. Birnbaum and E.L. Lozinskii, Consistent subsets of inconsistent systems: Structure and behaviour, *J. Exp. Theor. Artif. Intell.* **15**(1) (2003), 25–46.

[14] I.M. Bomze, M. Budinich, P.M. Pardalos and M. Pelillo, The maximum clique problem, in: *Handbook of Combinatorial Optimization*, Springer, 1999, pp. 1–74.

[15] C. Bourke, K. Deng, S.D. Scott, R.E. Schapire and N.V. Vinodchandran, On reoptimizing multi-class classifiers, *Machine Learning* **71**(2,3) (2008), 219–242.

[16] T. Brihaye, V. Bruyère, L. Doyen, M. Ducobu and J.-F. Raskin, Antichain-based QBF solving, in: *International Symposium on Automated Technology for Verification and Analysis*, 2011, pp. 183–197.

[17] A. Butman, D. Hermelin, M. Lewenstein and D. Rawitz, Optimization problems in multiple-interval graphs, *ACM Transactions on Algorithms* **6**(2) (2010), Article No. 40.

[18] S. Cai, K. Su and Q. Chen, EWLS: A new local search for minimum vertex cover, in: *AAAI Conference on Artificial Intelligence*, M. Fox and D. Poole, eds, AAAI Press, 2010.

[19] S. Cai, K. Su, C. Luo and A. Sattar, NuMVC: An efficient local search algorithm for minimum vertex cover, *J. Artif. Intell. Res. (JAIR)* **46** (2013), 687–716.

[20] S. Cai, K. Su and A. Sattar, Local search with edge weighting and configuration checking heuristics for minimum vertex cover, *Artif. Intell.* **175**(9,10) (2011), 1672–1696.

[21] S. Cai, K. Su and A. Sattar, Two new local search strategies for minimum vertex cover, in: *AAAI Conference on Artificial Intelligence*, J. Hoffmann and B. Selman, eds, AAAI Press, 2012.

[22] B. Chamaret, S. Josselin, P. Kuonen, M. Pizarroso, B. Salas-Manzanedo, S. Ubeda and D. Wagner, Radio network optimization with maximum independent set search, in: *Vehicular Technology Conference, 1997, IEEE 47th*, Vol. 2, May 1997, 1997, pp. 770–774.

[23] T. Chen, V. Filkov and S. Skiena, Identifying gene regulatory networks from experimental data, *Parallel Computing* **27**(1,2) (2001), 141–162.

[24] J. Davies and F. Bacchus, Solving MAXSAT by solving a sequence of simpler SAT instances, in: *International Conference on Principles and Practice of Constraint Programming*, 2011, pp. 225–239.

[25] J. Davies and F. Bacchus, Exploiting the power of MIP solvers in MAXSAT, in: *International Conference on Theory and Applications of Satisfiability Testing*, 2013, pp. 166–181.

[26] J. Davies and F. Bacchus, Postponing optimization to speed up MAXSAT solving, in: *International Conference on Principles and Practice of Constraint Programming*, 2013, pp. 247–262.

[27] E. Di Rosa, E. Giunchiglia and M. Maratea, Solving satisfiability problems with preferences, *Constraints* **15**(4) (2010), 485–515.

[28] N. Eén and N. Sörensson, An extensible SAT-solver, in: *International Conference on Theory and Applications of Satisfiability Testing*, 2003, pp. 502–518.

[29] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, 1979.

[30] J. Gate and I.A. Stewart, Frameworks for logically classifying polynomial-time optimisation problems, in: *International Computer Science Symposium in Russia*, 2010, pp. 120–131.

[31] F. Gavril, Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph, *SIAM J. Comput.* **1**(2) (1972), 180–187.

[32] E. Giunchiglia and M. Maratea, Solving optimization problems with DLL, in: *European Conference on Artificial Intelligence*, 2006, pp. 377–381.

[33] A. Goldstein, P. Kolman and J. Zheng, Minimum common string partition problem: Hardness and approximations, *Electr. J. Comb.* **12** (2005), R50.

[34] R. Hassin, J. Monnot and D. Segev, Approximation algorithms and hardness results for labeled connectivity problems, *J. Comb. Optim.* **14**(4) (2007), 437–453.

[35] F. Heras, A. Morgado, J. Planes and J. Marques-Silva, Iterative SAT solving for minimum satisfiability, in: *International Conference on Tools with Artificial Intelligence*, 2012, pp. 922–927.

[36] A. Ignatiev, M. Janota and J. Marques-Silva, Quantified maximum satisfiability: A core-guided approach, in: *International Conference on Theory and Applications of Satisfiability Testing*, 2013, pp. 250–266.

[37] A. Ignatiev, A. Morgado and J. Marques-Silva, On reducing maximum independent set to minimum satisfiability, in: *International Conference on Theory and Applications of Satisfiability Testing*, 2014, pp. 103–120.

[38] A. Ignatiev, A. Morgado, J. Planes and J. Marques-Silva, Maximal falsifiability: Definitions, algorithms, and applications, in: *International Conferences on Logic for Programming, Artificial Intelligence and Reasoning*, 2013, pp. 439–456.

[39] Y. Interian, G. Corvera, B. Selman and R. Williams, Finding small unsatisfiable cores to prove unsatisfiability of QBFs, in: *International Symposium on Artificial Intelligence and Mathematics*, 2006.

[40] K. Jain, J. Padhye, V.N. Padmanabhan and L. Qiu, Impact of interference on multi-hop wireless network performance, *Wireless Networks* **11**(4) (2005), 471–487.

[41] D.S. Johnson, C.H. Papadimitriou and M. Yannakakis, On generating all maximal independent sets, *Inf. Process. Lett.* **27**(3) (1988), 119–123.

[42] D. Joseph, J. Meidanis and P. Tiwari, Determining DNA sequence similarity using maximum independent set algorithms for interval graphs, in: *Scandinavian Workshop on Algorithm Theory*, Vol. 621, 1992, pp. 326–337.

[43] R.M. Karp, Reducibility among combinatorial problems, in: *Symposium on the Complexity of Computer Computations*, 1972, pp. 85–103.

[44] R.M. Karp and A. Wigderson, A fast parallel algorithm for the maximal independent set problem, *J. ACM* **32**(4) (1985), 762–773.

[45] R. Kohli, R. Krishnamurti and K. Jedidi, Subset-conjunctive rules for breast cancer diagnosis, *Discrete Applied Mathematics* **154**(7) (2006), 1100–1112.

[46] R. Kohli, R. Krishnamurti and P. Mirchandani, The minimum satisfiability problem, *SIAM J. Discrete Math.* **7**(2) (1994), 275–283.

[47] A. Kugel, Natural Max-SAT encoding of Min-SAT, in: *International Conference on Learning and Intelligent Optimization*, 2012, pp. 431–436.

[48] E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Generating all maximal independent sets: NP-hardness and polynomial-time algorithms, *SIAM J. Comput.* **9**(3) (1980), 558–565.

[49] C.M. Li and F. Manya, MaxSAT, hard and soft constraints, in: *Frontiers in Artificial Intelligence and Applications*, A. Biere, M. Heule, H. van Maaren and T. Walsh, eds, Handbook of Satisfiability, Vol. 185, IOS Press, 2009, pp. 613–631.

[50] C.M. Li, F. Manya, Z. Quan and Z. Zhu, Exact MinSAT solving, in: *International Conference on Theory and Applications of Satisfiability Testing*, 2010, pp. 363–368.

[51] C.M. Li and Z. Quan, Combining graph structure exploitation and propositional reasoning for the maximum clique problem, in: *International Conference on Tools with Artificial Intelligence*, 2010, pp. 344–351.

[52] C.M. Li, Z. Zhu, F. Manya and L. Simon, Minimum satisfiability and its applications, in: *International Joint Conference on Artificial Intelligence*, 2011, pp. 605–610.

[53] C.M. Li, Z. Zhu, F. Manya and L. Simon, Optimizing with minimum satisfiability, *Artif. Intell.* **190** (2012), 32–44.

[54] M.H. Liffiton and A. Malik, Enumerating infeasibility: Finding multiple muses quickly, in: *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, 2013, pp. 160–175.

[55] M.H. Liffiton, M.N. Mneimneh, I. Lynce, Z.S. Andraus, J. Marques-Silva and K.A. Sakallah, A branch and bound algorithm for extracting smallest minimal unsatisfiable subformulas, *Constraints* **14**(4) (2009), 415–442.

[56] M.H. Liffiton and K.A. Sakallah, Algorithms for computing minimal unsatisfiable subsets of constraints, *J. Autom. Reasoning* **40**(1) (2008), 1–33.

[57] M.V. Marathe and S.S. Ravi, On approximation algorithms for the minimum satisfiability problem, *Inf. Process. Lett.* **58**(1) (1996), 23–29.

[58] J. Marques-Silva, F. Heras, M. Janota, A. Previti and A. Belov, On computing minimal correction subsets, in: *International Joint Conference on Artificial Intelligence*, 2013, pp. 615–622.

[59] R. Martins, S. Joshi, V.M. Manquinho and I. Lynce, Incremental cardinality constraints for MaxSAT, in: *International Conference on Principles and Practice of Constraint Programming*, 2014, pp. 531–548.

[60] C. Mencía, A. Previti and J. Marques-Silva, Literal-based MCS extraction, in: *International Joint Conference on Artificial Intelligence*, 2015, pp. 1973–1979.

[61] 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) (2013), 478–534.

[62] A. Morgado, M.H. Liffiton and J. Marques-Silva, MaxSAT-based MCS enumeration, in: *International Haifa Verification Conference*, 2012, pp. 86–101.

[63] N. Narodytska and F. Bacchus, Maximum satisfiability using core-guided MaxSAT resolution, in: *AAAI Conference on Artificial Intelligence*, 2014, pp. 2717–2723.

[64] A. Nöhrer, A. Biere and A. Egyed, Managing SAT inconsistencies with HUMUS, in: *Workshop on Variability Modelling of Software-Intensive Systems*, 2012, pp. 83–91.

[65] P.R.J. Östergård, A fast algorithm for the maximum clique problem, *Discrete Applied Mathematics* **120**(1–3) (2002), 197–207.

[66] P.M. Pardalos and J. Xue, The maximum clique problem, *Journal of Global Optimization* **4**(3) (1994), 301–328.

[67] A. Previti and J. Marques-Silva, Partial MUS enumeration, in: *AAAI Conference on Artificial Intelligence*, 2013, pp. 818–825.

[68] W.J. Pullan, Approximating the maximum vertex/edge weighted clique using local search, *J. Heuristics* **14**(2) (2008), 117–134.

[69] W.J. Pullan and H.H. Hoos, Dynamic local search for the maximum clique problem, *J. Artif. Intell. Res. (JAIR)* **25** (2006), 159–185.

[70] R. Ramaswami and K.N. Sivarajan, Routing and wavelength assignment in all-optical networks, *IEEE/ACM Trans. Netw.* **3**(5) (1995), 489–500.

[71] R. Reiter, A theory of diagnosis from first principles, *Artif. Intell.* **32**(1) (1987), 57–95.

[72] M.G.C. Resende, T.A. Feo and S.H. Smith, Algorithm 787: Fortran subroutines for approximate solution of maximum independent set problems using GRASP, *ACM Trans. Math. Softw.* **24**(4) (1998), 386–394.

[73] J.M. Robson, Algorithms for maximum independent sets, *J. Algorithms* **7**(3) (1986), 425–440.

[74] R.E. Tarjan and A.E. Trojanowski, Finding a maximum independent set, *SIAM J. Comput.* **6**(3) (1977), 537–546.

[75] S. Tsukiyama, M. Ide, H. Ariyoshi and I. Shirakawa, A new algorithm for generating all the maximal independent sets, *SIAM J. Comput.* **6**(3) (1977), 505–517.

[76] Z. Zhu, C.M. Li, F. Manya and J. Argelich, A new encoding from MinSAT into MaxSAT, in: *International Conference on Principles and Practice of Constraint Programming*, 2012, pp. 455–463.