On Reducing Maximum Independent Set to Minimum Satisfiability

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

[1] IST/INESC-ID, Lisbon, Portugal
[2] University College Dublin, Ireland
[3] ISDCT SB RAS, Irkutsk, Russia
{aign,ajrm}@sat.inesc-id.pt, jpms@ucd.ie

Abstract. Maximum Independent Set (MIS) is a well-known NP-hard graph problem, tightly related with other well known NP-hard graph problems, namely Minimum Vertex Cover (MVC) and Maximum Clique (MaxClq). This paper introduces a novel reduction of MIS into Minimum Satisfiability (MinSAT), thus, providing an alternative approach for solving MIS. The reduction naturally maps the vertices of a graph into clauses, without requiring the inclusion of hard clauses. Moreover, it is shown that the proposed reduction uses fewer variables and clauses than the existing alternative of mapping MIS into Maximum Satisfiability (MaxSAT). The paper develops a number of optimizations to the basic reduction, which significantly reduce the total number of variables used. The experimental evaluation considered the reductions described in the paper as well as existing state-of-the-art approaches. The results show that the proposed approaches based on MinSAT are competitive with existing approaches.

1 Introduction

Maximum Independent Set (MIS) is a well-known NP-hard graph problem, tightly related with other well known NP-hard graph problems, namely Minimum Vertex Cover (MVC) and Maximum Clique (MaxClq) [13]. These NP-hard graph problems find a wide range of practical applications, having been extensively studied in a number of settings over the last few decades (e.g. see [1, 4, 6–10, 12, 18, 23–25, 30, 35, 36, 42–46, 48, 49, 51, 56] and references therein). A large number of solutions have been developed for these NP-hard graph problems including complete algorithms, e.g. branch-and-bound search, but also incomplete algorithms, e.g. local search, genetic algorithms. These works also include recent algorithms for the MaxClq problem [35], as well as reductions of MaxClq to Maximum Satisfiability (MaxSAT) [36].

In contrast, and although work in Minimum Satisfiability (MinSAT) [27, 40] can be traced to the mid 90s, to our best knowledge, no *natural* applications have been described in the literature for MinSAT. For example, earlier work on MinSAT mainly targeted the development of inapproximability results for other combinatorial optimization problems. Admittedly, it is well-known how to reduce the different variants of MaxSAT to MinSAT (e.g. [19, 37]). For example, by flipping the polarity of literals when soft clauses are all unit, one obtains a MinSAT instance instead of a MaxSAT instance [37]. However, since algorithms for MaxSAT and MinSAT share many insights, these mappings are not expected to provide significant breakthroughs. Nevertheless, there has been significant recent research activity on algorithms for the MinSAT problem, and

C. Sinz and U. Egly (Eds.): SAT 2014, LNCS 8561, pp. 103–120, 2014.

its variants [19, 34, 37, 38], and so it is of interest to find combinatorial optimization problems that can be modeled as variants of MinSAT. A preliminary example towards achieving this goal is the encoding of WMaxCSP into weighted partial MinSAT, recently proposed in [3].

This paper represents another step towards identifying combinatorial problems that can be modeled as variants of MinSAT, in our case of (weighted) MinSAT. More concretely, this paper establishes a relationship between MIS (and other related NP-hard graph problems) and MinSAT, by showing how to reduce MIS to MinSAT. The reduction of MIS (and so of related graph problems) to MinSAT is significant due to the fact that it provides many concrete practical applications of MinSAT, something that to our best knowledge was not known. Besides the basic reduction (and associated proof), the paper also develops a number of optimizations which are shown to be crucially effective when solving MIS problem instances in practice.

The paper is organized as follows. Section 2 introduces the notation and definitions used throughout the paper. Section 3 develops the basic reduction of MIS to MinSAT. Section 4 develops optimizations to the basic reduction which in practice yield significant reductions in the number of used variables. Preliminary experimental results are analyzed in Section 5. Finally, Section 6 concludes the paper.

2 Preliminaries

This section briefly introduces the definitions used throughout the paper. Additional standard definitions can be found elsewhere (e.g. [5]). 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. Where appropriate, a CNF formula will also be understood as a conjunction of disjunctions of literals, where each disjunction represents a *clause* and a *literal* is a Boolean variable or its complement. Boolean variables are represented by $\{x, x_1, x_2, \ldots\}$, and literals by $\{l, l_1, l_2, \ldots\}$. The set of all variables of formula $\mathcal{F}$ is denoted by $\texttt{var}(\mathcal{F})$. The clauses of a formula are represented by $\{c, c_1, c_2, \ldots\}$. Two literals are said to be complementary, if one of the literals corresponds to a variable x, while the other corresponds to the negation of the variable, that is $\neg x$. An assignment is a mapping $\mathcal{A} : \texttt{var}(\mathcal{F}) \mapsto \{0, 1\}$. A clause is satisfied by an assignment if one of its literals is assigned value 1. A model of $\mathcal{F}$ is an assignment that satisfies all clauses in $\mathcal{F}$.

The standard definitions of MinSAT and MaxSAT are assumed (e.g. [32, 38]). In the context of MinSAT and 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. A weight can be associated with each clause, such that hard clauses have a special weight $\top$. Hence, the weight function is a mapping $w : \mathcal{H} \cup \mathcal{R} \to \{\top\} \cup \mathbb{N}$, such that $\forall_{c \in \mathcal{H}}\, w(c) = \top$ and $\sum_{c \in \mathcal{R}} w(c) < \top$. If no weight function is specified, it is assumed that $\forall_{c \in \mathcal{R}}\, w(c) = 1$.

The MinSAT and MaxSAT problems are defined as follows. The *Minimum/Maximum Satisfiability* (MinSAT/MaxSAT) problem consists in computing a subset of soft clauses $\mathcal{S} \subseteq \mathcal{R}$, that minimizes/maximizes the sum of the weights of the clauses in $\mathcal{S}$, such that $\mathcal{S} \cup \mathcal{H}$ is satisfiable while falsifying $R \setminus S$. Closely related to MinSAT, is the *Maximum Falsifiability* (MaxFalse) problem, which corresponds to computing the complement of a solution of the MinSAT problem (see [22]).

The paper also considers a number of optimization problems in graphs. Consider an undirected graph $\mathcal{G} = (V, E)$, where $r = |V|$ and $s = |E|$. An *Independent Set* (IS) is a set $I \subseteq V$ such that $\forall_{u,v \in I},\ (u, v) \notin E$. A *Vertex Cover* (VC) 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 the complemented graph $\mathcal{G}^C$ (e.g. [13]). The *Maximum Independent Set* (MIS) problem consists in computing an IS of largest size. The 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 also represents a solution for *Minimum Vertex Cover* (MVC) of a graph $\mathcal{G}$, as well as a *Maximum Clique* (MaxClq) of the complemented graph $\mathcal{G}^C$.

2.1 Related Work

As mentioned before, a solution for the MIS problem can be used to compute a solution for the MVC problem or the MaxClq problem (and vice-versa). Approaches to the considered problems can be divided in two main categories, either exact algorithms or heuristic methods. Heuristic algorithms try to obtain a solution quickly but do not guarantee the optimality of the solution returned. Local search has been extensively used as a way to obtain a heuristic solution to the problems (e.g. see [1, 4, 7–10, 44, 45] and references therein).

An approximation to the MIS problem can also be obtained by a greedy algorithm [14] that at each step selects a vertex to belong to the MIS and removes all other vertices that share an edge with the selected vertex. In [14, 39, 57], the greedy heuristic approach was explored as a way to obtain lower bounds in a branch-and-bound procedure for binate covering problems.

In contrast to the heuristic methods, exact algorithms guarantee the optimality of the solution returned. Many exact algorithms can be found in the literature, most of which are based on the branch-and-bound technique (e.g. [11, 15, 28, 35, 36, 42, 47, 50, 52–54]). An additional approach to solve MaxClq is characterized by encoding the MaxClq problem into MaxSAT, and use an off-the-shelf MaxSAT solver on the encoded instance. Nevertheless, such approach is not competitive with current state-of-the-art exact MaxClq solvers [36] (also confirmed by our experimental results, see Section 5).

Recent years have seen a growing interest in the development of algorithms and techniques for the MinSAT problem. Existing works can be categorized in two main areas of research: either by reducing the MinSAT problem into a MaxSAT problem (e.g. [20, 29, 34, 58]); or by proposing a dedicated MinSAT solver (e.g. [2, 3, 20, 37, 38]).

3 Reducing MIS to MinSAT

Reductions of MaxClq to MaxSAT are well-known (e.g. [35]). Since a solution to a MIS problem can be obtained by solving the MaxClq problem of the complemented graph, then MaxSAT algorithms can be used for computing solutions to MIS. Additionally, MaxSAT can be reduced into MinSAT using auxiliary variables [55], thus making it possible to solve MIS through MinSAT.

This section proposes a natural reduction of the MIS problem directly into MinSAT that does not require the addition of hard clauses. The proposed reduction, referred to

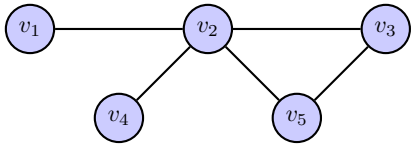


Fig. 1. Example graph

as *basic*, was recently mentioned in the context of Maximum Falsifiability [22]. Maximum Falsifiability (MaxFalse) was introduced in [22] as the problem of computing the complement of a MinSAT solution.

Consider an undirected graph $\mathcal{G} = (V, E)$, and a CNF formula $\mathcal{F}$. The idea of the basic reduction into MinSAT/MaxFalse is to associate with each vertex $v_i \in V$, a clause $c_i \in \mathcal{F}$. If c_i is falsified in the MinSAT/MaxFalse solution of $\mathcal{F}$, then the corresponding vertex v_i is included in the solution of the MIS of $\mathcal{G}$.

Each pair of clauses c_i, $c_j \in \mathcal{F}$, whose associated vertices have an edge between them in the graph ($(v_i, v_j) \in E$), cannot be allowed to be falsified simultaneously (due to the independence of vertices). As such, for each edge $(v_i, v_j) \in E$, a new variable x is created, and the literal x is added to c_i, while $\neg x$ is added to c_j. Any assignment to the variable x will force at least one of the clauses c_i or c_j to be satisfied.

Example 1. Consider the graph $\mathcal{G} = (V, E)$, with $V = \{v_1, v_2, v_3, v_4, v_5\}$ and $E = \{(v_1, v_2), (v_2, v_3), (v_2, v_4), (v_2, v_5), (v_3, v_5)\}$, as shown in Figure 1.

The basic reduction creates a new CNF formula $\mathcal{F}$. Each vertex v_i is represented by a clause c_i, and each edge of the graph (v_i, v_j) introduces a new variable $x_{i,j}$. Formula $\mathcal{F}$ is formed by the clauses in Equation 1.

$$\begin{aligned} c_1 &= x_{1,2} \\ c_2 &= \neg x_{1,2} \vee x_{2,3} \vee x_{2,4} \vee x_{2,5} \\ c_3 &= \neg x_{2,3} \vee x_{3,5} \\ c_4 &= \neg x_{2,4} \\ c_5 &= \neg x_{2,5} \vee \neg x_{3,5} \end{aligned} \tag{1}$$

The following proposition proves the correctness of the basic reduction.

Proposition 1. *Given a graph $\mathcal{G} = (V, E)$. Let $\mathcal{F}$ be a CNF formula obtained from $\mathcal{G}$ by the above basic reduction.*

(1) Any MinSAT/MaxFalse solution of $\mathcal{F}$ represents an MIS of $\mathcal{G}$.

(2) Any MIS solution of $\mathcal{G}$ represents a MinSAT/MaxFalse solution of $\mathcal{F}$.

Proof. Here we consider just MaxFalse (for MinSAT the complement can be used).

(1) Consider a MaxFalse solution of $\mathcal{F}$, and let $\mathcal{F}' \subseteq \mathcal{F}$ be the set of clauses that are falsified by the MaxFalse solution. Then there is an assignment $\mathcal{A}$ that falsifies all clauses in $\mathcal{F}'$. Let $V' \subseteq V$ be a set of vertices that are associated to the clauses in $\mathcal{F}'$.

First we prove that the vertices in V' are independent. Assume by contradiction that the vertices in V' are not independent. Then there is (at least) one edge between two of its vertices, which means there is a variable x such that x belongs to one of the associated clauses in $\mathcal{F}'$ and $\neg x$ to another. Those two clauses cannot be simultaneously falsified, which is a contradiction since $\mathcal{A}$ falsifies all clauses in $\mathcal{F}'$.

Now we prove that V' is an MIS of $\mathcal{G}$. Assume by contradiction that V' is not maximum. Then there is a set $V'' \subseteq V$ such that V'' is an MIS and $|V''| > |V'|$. Let $\mathcal{F}'' \subseteq \mathcal{F}$ be the set of clauses associated to V''. Since the vertices in V'' are independent (no edges between them), the clauses in $\mathcal{F}''$ share no variables, and can be simultaneously falsified (consider an assignment that falsifies all literals in $\mathcal{F}''$). But $|\mathcal{F}''| = |V''| > |V'| = |\mathcal{F}'|$, which is a contradiction since $\mathcal{F}'$ is a MaxFalse/MinSAT solution.

(2) Consider $V' \subseteq V$ an MIS of $\mathcal{G}$ and let $\mathcal{F}' \subseteq \mathcal{F}$ be the set of clauses associated to the vertices in V'. The clauses in $\mathcal{F}'$ are simultaneously falsifiable, because the vertices in V' are independent, which means the clauses in $\mathcal{F}'$ share no variables, so the assignment that falsifies all literals in the $\mathcal{F}'$ is able to falsify all clauses in $\mathcal{F}'$.

Finally, we prove that $\mathcal{F}'$ is a MaxFalse solution. Consider by contradiction that $\mathcal{F}'$ is not a MaxFalse solution. Since the clauses in $\mathcal{F}'$ are simultaneously falsifiable, then there is a set $\mathcal{F}'' \subseteq \mathcal{F}$ such that $\mathcal{F}''$ is a MaxFalse solution and $|\mathcal{F}''| > |\mathcal{F}'|$. Let $V'' \subseteq V$ be the set of vertices associated to the clause in $\mathcal{F}''$. Since the clauses in $\mathcal{F}''$ are simultaneously falsifiable then the vertices in V'' are independent (otherwise there would be two different clauses in $\mathcal{F}''$ containing complementary literals, and the clauses in $\mathcal{F}''$ could not be simultaneously falsifiable). But $|V''| = |\mathcal{F}''| > |\mathcal{F}'| = |V'|$, which is a contradiction since V' is a MIS of $\mathcal{G}$. □

Observe that, given a graph with r vertices and s edges, the basic reduction of MIS into MinSAT/MaxFalse always introduces s variables and r clauses[1]. As such, the basic reduction represents a polynomial-time reduction from MIS to MinSAT/MaxFalse[2].

Although the proposed reduction is not efficient in terms of the number of used variables, it is still more compact than the known reductions from MaxClq to MaxSAT (e.g see [35, 36]). To the best of our knowledge, there are two MaxSAT encodings of MaxClq described in the literature, both have to deal with not only soft clauses but also with hard clauses (partial MaxSAT instances are constructed). Given a graph $\mathcal{G} = (V, E)$, $|V| = r$, $|E| = s$, the first encoding produces r variables and r soft clauses, as well as $\frac{r \cdot (r-1)}{2} - s$ hard clauses (one hard clause for each edge of the complemented graph $\mathcal{G}^C$). The improved version of this encoding reduces the number of soft clauses to k, where $k \leq r$ is the number of disjoint independent sets of $\mathcal{G}$ computed heuristically. The reader is referred to [35, 36] for details.

Additionally observe that, the basic reduction does not always produce a formula with the minimal number of variables. That is, there are graphs for which a CNF formula with r clauses and less than s variables can be obtained, whose MinSAT/MaxFalse solution represents an MIS solution of the original graph. Section 4 introduces several techniques that reduce the number of variables.

[1] Note that the number of edges s is usually much higher than the number of vertices r and can be potentially close to $\frac{r \cdot (r-1)}{2}$.

[2] Observe that since MIS is an NP-hard problem, then the basic reduction provides an (alternative) natural proof of NP-hardness of both MinSAT and MaxFalse. The original proof of NP-hardness of MinSAT was demonstrated by reducing MaxSAT into MinSAT (see [27]). However, note that 2-MIS — the maximum independent set problem for a graph with a vertex degree bounded by 2 — can be trivially solved in polynomial time [17]. Therefore, the basic reduction does not cover the case of NP-hardness of 2-MinSAT/2-MaxFalse, even though they are also known to be NP-hard (see [27]).

4 Improvements to the Basic Reduction

Consider a graph $\mathcal{G} = (V, E)$, $|V| = r$, $|E| = s$. As it was shown in Section 3, the number of variables introduced by the basic reduction of the MIS problem for $\mathcal{G}$ to MinSAT/MaxFalse is equal to the number of edges of the graph $\mathcal{G}$, which is bounded by $\frac{r \cdot (r-1)}{2}$. This means that in the case of dense graphs with a large number of vertices the basic reduction generates CNF formulas with a large number of variables, which does not allow one to efficiently use this reduction in practice. This section describes 3 techniques for producing CNF formulas with a number of variables smaller by orders of magnitude compared to the basic reduction. In some cases decreasing the number of variables also leads to formula simplification by decreasing the number of clauses. Additionally, we also make the conjecture that given a graph $\mathcal{G}$, finding a formula $\mathcal{F}$ with the minimum number of variables cannot be done in polynomial time.

4.1 Greedy Approach

In contrast to introducing a new variable for each edge of graph $\mathcal{G}$, the greedy approach is able to use one variable for several edges. The greedy approach hinges on the idea that all edges incident to a vertex of $\mathcal{G}$ can be represented by one Boolean variable. As a result, the number of variables used to encode a graph into a formula is bounded by the number r of vertices of $\mathcal{G}$.

The pseudocode of the greedy algorithm is shown in Algorithm 1. For any graph $\mathcal{G} = (V, E)$ it constructs a set of clauses $\mathcal{F}$, each clause of which corresponds to a vertex of $\mathcal{G}$ and several edges of $\mathcal{G}$ can be encoded by the same pair of literals. Initially each clause of formula $\mathcal{F}$ is an empty set of literals (see lines 2–5). Since the graphs we consider are not directed, we assume that both (v_i, v_j) and (v_j, v_i) denote the same edge between vertices v_i and v_j of the graph. The idea of the algorithm is that each edge incident to a vertex v_i can be encoded by the same pair of literals x_i and $\neg x_i$. Therefore, at each iteration of the main loop, Algorithm 1 picks a vertex v_i of $\mathcal{G}$ with the maximum degree (line 7) and introduces a positive literal x_i to clause c_i (line 8). After that for each vertex v_j that has a connection to v_i, clause c_j gets a literal $\neg x_i$ (see line 10). All the considered edges (all the ones incident to v_i) are removed from graph $\mathcal{G}$. The loop continues until there are no edges in the graph that are not yet encoded.

Example 2. Consider the graph represented in Figure 1. Figure 2 illustrates how the greedy reduction works. First, it picks v_2 as a vertex with the maximum degree and adds a literal x_2 into c_2, while literal $\neg x_2$ is added into clauses c_1, c_3, c_4, and c_5. Then the corresponding edges are removed from the graph. The only edge in the graph that is not yet considered is the edge (v_3, v_5). The next vertex with the maximum degree is v_3. The algorithm adds literal x_3 into clause c_3, while $\neg x_3$ is added into c_5. The resulting set of clauses is shown in (2) below.

$$\begin{aligned} c_1 &= \neg x_2 \\ c_2 &= x_2 \\ c_3 &= \neg x_2 \vee x_3 \\ c_4 &= \neg x_2 \\ c_5 &= \neg x_2 \vee \neg x_3 \end{aligned} \tag{2}$$

Algorithm 1. Greedy reduction algorithm

```
1  Function Greedy(G = (V, E))
2      F ← ∅
3      foreach v_i ∈ V do
4          c_i ← ∅
5          F ← F ∪ c_i
6      while E ≠ ∅ do
7          v_i ← vertex in V with maximum degree
8          c_i ← c_i ∪ {x_i}
9          foreach v_j ∈ V s.t. (v_i, v_j) ∈ E do
10             c_j ← c_j ∪ {¬x_i}
11             E ← E \ {(v_i, v_j)}
12     return F
```

(a) Pick v_2 and encode its connections

(b) Pick v_3 and encode its connections

Fig. 2. Example on how the greedy approach works

Proposition 2. *Given a graph $\mathcal{G}$ with r vertices, the complexity of the greedy reduction algorithm for graph $\mathcal{G}$ is $\mathcal{O}(r^2)$.*

Proof. Observe that the algorithm has to traverse all the edges of graph $\mathcal{G}$, and each is traversed once. The trivial worst case scenario is when graph $\mathcal{G}$ is a clique — in this case graph $\mathcal{G}$ has r^2 edges. □

Proposition 3. *Given a graph $\mathcal{G} = (V, E)$, let $\mathcal{F} = Greedy(\mathcal{G})$. $V' \subseteq V$ is an MIS of $\mathcal{G}$ iff the set $\mathcal{F}' \subseteq \mathcal{F}$ of clauses associated to the vertices in V' is a MaxFalse solution (the complement of a MinSAT solution) of $\mathcal{F}$.*

For the sake of succinctness and due to lack of space, we do not provide a proof of Proposition 3. However, the correctness of the greedy approach can be shown using an argumentation analogous to the proof of Proposition 1.

Also note that the proposed greedy algorithm is known to be not optimal in terms of the number of used variables. A simple example of a graph, for which the greedy approach is not optimal is shown in Figure 3. For this graph it is enough to introduce 4 variables, while the greedy algorithm introduces 5 variables. The resulting set of clauses produced by the greedy algorithm and the *optimal encoding* are shown in Figure 4.

Nevertheless, the greedy algorithm can be seen as a significant improvement over the basic approach. Given a graph $\mathcal{G} = (V, E)$, where $|V| = r$, the basic reduction can

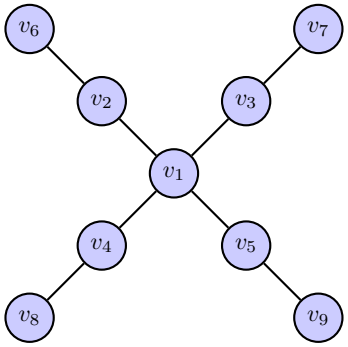


Fig. 3. Counterexample showing non-optimality of the greedy algorithm

$c_1 = x_1$
$c_2 = \neg x_1 \vee x_2$
$c_3 = \neg x_1 \vee x_3$
$c_4 = \neg x_1 \vee x_4$
$c_5 = \neg x_1 \vee x_5$
$c_6 = \neg x_2$
$c_7 = \neg x_3$
$c_8 = \neg x_4$
$c_9 = \neg x_5$

(a) Greedy encoding

$c_1 = \neg x_2 \vee \neg x_3 \vee \neg x_4 \vee \neg x_5$
$c_2 = x_2$
$c_3 = x_3$
$c_4 = x_4$
$c_5 = x_5$
$c_6 = \neg x_2$
$c_7 = \neg x_3$
$c_8 = \neg x_4$
$c_9 = \neg x_5$

(b) Optimal encoding

Fig. 4. Sets of clauses produced for the graph shown in Figure 3

potentially introduce $\mathcal{O}(r^2)$ variables while the number of variables used by the greedy algorithm is bounded by r.

4.2 Optimizations

This section provides a description of two additional heuristic techniques for minimizing the number of variables. Both techniques can be applied to the formulas produced by the considered basic and greedy reduction algorithms.

Variable Compatibility. Although CNF formulas produced by the greedy reduction are much more compact (in terms of the number of used variables) than the ones produced by the basic reduction, in some cases it might be still possible to reduce the number of variables even more. The following technique is referred to as *variable compatibility*. The idea of the variable compatibility method originates from the approaches used in the automata theory for the finite-state machine minimization, which makes use of the so-called compatibility graphs and merger tables [21, 26].

While the original method of simplifying finite-state machines (e.g. see [26]) operates with the so-called *compatible states*, here we use a notion of *compatible variables*. Variables decided to be compatible can replace each other and, thus, reduce the total number of variables. Assume that for a given graph $\mathcal{G} = (V, E)$, where $|V| = r$, $|E| = s$, an MIS to MaxFalse reduction (either basic or greedy) produces a CNF formula $\mathcal{F}$ over a set X, $|X| = k$, variables. Note that for the case of the basic reduction $k \leq s$, for the greedy reduction, $k \leq r$.

The variable compatibility method consists in constructing and filling a $k \times k$ table, rows and columns of which are labeled by variables of X. Filling a cell of the table with coordinates (x_i, x_j) means that variables x_i and x_j are not compatible. Initially, all the cells of the table are empty (all variables are possibly compatible). The following rules can be used in order to conclude that two variables are not compatible.

1. A clause of $\mathcal{F}$ cannot be a *tautology*, i.e. it cannot contain literals x_i and $\neg x_i$ simultaneously. For example, given a clause $\neg x_1 \vee x_2$, variables x_1 and x_2 cannot replace each other. Otherwise, the clause is a tautology.
2. The structure of the clauses must enforce *no new connection* between vertices of the original graph $\mathcal{G}$. For example, given two clauses $\neg x_1 \vee x_2$ and $\neg x_3$, variables x_2 and x_3 cannot replace each other. Otherwise, there is an edge between the corresponding vertices in the original graph.

Example 3. Consider the graph represented in Figure 1. Although the clauses produced by the basic reduction for this graph are already shown in (1), for the sake of simplicity we represent it again with each variable having exactly one index (instead of two).

$$\begin{aligned} c_1 &= x_1 \\ c_2 &= \neg x_1 \vee x_2 \vee x_3 \vee x_4 \\ c_3 &= \neg x_2 \vee x_5 \\ c_4 &= \neg x_3 \\ c_5 &= \neg x_4 \vee \neg x_5 \end{aligned} \tag{3}$$

In order to determine the classes of *compatible variables*, the following compatibility table can be constructed:

	x_1	x_2	x_3	x_4	x_5
x_1	–				
x_2	$*_2 *_{1,3}$	–			
x_3	$*_2 *_{1,4}$		–		
x_4	$*_2 *_{1,5}$			–	
x_5	$*_{1,5}$	$*_3$	$*_{3,4}$		–

If two variables are not compatible because of violating rule 1, then the corresponding cell of the table is marked by $*$. If they are not compatible because of rule 2, the cell is marked by symbol $*$. The subscripts denote the clauses involved.

Note that rule 1 can be applied to each clause that contains both positive and negative literals. The second rule can be applied to a pair of clauses if there is *no edge* between the corresponding nodes in the original graph $\mathcal{G}$. For checking that, one can construct a graph $\mathcal{G}^C$, which is complement to $\mathcal{G}$ and use edges of $\mathcal{G}^C$ to check the validity of rule 2. In our example the edges of $\mathcal{G}^C$ are represented by the pairs of clauses: (c_1, c_3), (c_1, c_4), (c_1, c_5), (c_3, c_4), and (c_4, c_5).

After applying both rules and filling the table one can see that there are 3 compatibility classes: (x_2, x_3, x_4), (x_3, x_4), (x_4, x_5). Here, one can heuristically choose a compatibility class to use. Using the first compatibility class (x_2, x_3, x_4) enables us to use 3 variables instead of the original 5: x_1, x_2, and x_5 (variables x_3 and x_4 are replaced by x_2). Thus, as a result, the clauses after their modification are shown in (4).

$$\begin{aligned} c_1 &= x_1 \\ c_2 &= \neg x_1 \vee x_2 \\ c_3 &= \neg x_2 \vee x_5 \\ c_4 &= \neg x_2 \\ c_5 &= \neg x_2 \vee \neg x_5 \end{aligned} \tag{4}$$

An immediate observation is that time complexity of the variable compatibility heuristic is formed by the time required to check both rules 1 and 2. In order to check rule 1, in the worst case one has to traverse pairs of literals in each clause of $\mathcal{F}$. For checking rule 2, in the worst case one has to traverse pairs of literals in all pairs of clauses. Thus, the complexity of variable compatibility is $\mathcal{O}(k^2 \cdot r^2)$, where k and r are numbers of variables and clauses in the original formula $\mathcal{F}$, respectively.

Also observe that the example above illustrates the non-optimality of the variable compatibility technique. It was shown in Section 4.1 that it is enough to use only 2 variables for encoding graph $\mathcal{G}$ from Figure 1, while variable compatibility leaves 3 variables in formula $\mathcal{F}$ after doing the simplification.

Literal Compatibility. An immediate observation is that a possible improvement of variable compatibility, which was shown above to be not optimal, can be *literal compatibility*. The idea is that instead of computing compatibility classes for variables, one can try to determine classes of *compatible literals*. The two rules to apply are almost the same and can be seen as a generalization of the ones presented in Section 4.2:

1. *(no tautology)*, e.g. given a clause $\neg x_1 \vee x_2$, literals $\neg x_1$ and $\neg x_2$ are not compatible. Note that if literals x_1 and x_j are not compatible, then literals $\neg x_1$ and $\neg x_2$ are not compatible as well.
2. *(no new connection)*, e.g. given a pair of clauses: $\neg x_1 \vee x_2$ and $\neg x_3$, literals $\neg\neg x_1 = x_1$ and $\neg x_3$, $\neg x_2$ and $\neg x_3$ are not compatible.

Example 4. Consider the graph shown in Figure 1. In order to improve the basic reduction, one can construct the following compatibility table.

	x_1	$\neg x_1$	x_2	$\neg x_2$	x_3	$\neg x_3$	x_4	$\neg x_4$	x_5	$\neg x_5$
x_1	–	–								
$\neg x_1$	–	–								
x_2	$*_2 *_{1,3}$		–	–						
$\neg x_2$		$*_2 *_{1,3}$	–	–						
x_3	$*_2 *_{1,4}$			$*_2 *_{3,4}$	–	–				
$\neg x_3$		$*_2 *_{1,4}$	$*_2 *_{3,4}$		–	–				
x_4	$*_2 *_{1,5}$			$*_2$		$*_2 *_{4,5}$	–	–		
$\neg x_4$		$*_2 *_{1,5}$	$*_2$		$*_2 *_{4,5}$		–	–		
x_5	$*_{1,5}$	$*_{1,3}$	$*_3$		$*_{3,4}$	$*_{4,5}$		$*_5$	–	–
$\neg x_5$	$*_{1,3}$	$*_{1,5}$		$*_3$	$*_{4,5}$	$*_{3,4}$	$*_5$		–	–

The compatibility classes for literals x and $\neg x$ should be symmetric (e.g. see compatibility classes $(x_1, \neg x_2, \neg x_3, \neg x_4)$ and $(\neg x_1, x_2, x_3, x_4)$). Thus, we consider only classes for positive literals. So, according to the result table there are the following classes: $(x_1, \neg x_2, \neg x_3, \neg x_4)$, $(x_2, x_3, x_4, \neg x_5)$, (x_3, x_4), and (x_4, x_5). Using the first compatibility class, one can get 2 variables instead of the original 5. As a result, the clauses after their modification (using the first compatibility class) are:

$$\begin{aligned} c_1 &= x_1 \\ c_2 &= \neg x_1 \\ c_3 &= x_1 \vee x_5 \\ c_4 &= x_1 \\ c_5 &= x_1 \vee \neg x_5 \end{aligned} \tag{5}$$

Observe that time complexity of the literal compatibility heuristic is asymptotically the same ($\mathcal{O}(k^2 \cdot r^2)$) as the time complexity of variable compatibility. The only difference is that instead of k variables, for the case of literal compatibility $2k$ literals are considered.

Note that although literal compatibility is supposed to be more compact than variable compatibility, it is still not optimal in terms of the number of used variables. As an example, one can consider a graph shown in Figure 3 and use the literal compatibility technique to reduce the set of clauses produced by the greedy reduction (see Figure 4a). In this case, literal compatibility is not able to remove any variable and leaves the formula as it is, containing 5 variables. Recall that the optimal encoding for the graph presented in Figure 3 is shown in Figure 4b and contains 4 variables.

4.3 Further CNF Formula Simplification

All the techniques described in the previous sections can reduce the number of variables used when reducing MIS to MinSAT/MaxFalse by orders of magnitude (see Section 4.1). However, one can simplify the resulting formula even more. Here we give a brief explanation of how we can make the formula simpler after we have finished *removing* variables.

Recall that each clause of the formula being produced represents a vertex of the original graph. In many practical cases the number of removed variables is so large that some of the clauses that originally represent different vertices of the graph and, thus, have different literals, start duplicating each other. With a view to simplify the formula, one can keep just one version of each clause while removing all the duplications and making the formula weighted. Although it is not always the case, there are situations when formulas being produced get simplified by orders of magnitude. As an example, Table 1 shows the number of variables and clauses in the formulas produced for the *c-fat* family of DIMACS MaxClq[3] instances by the basic reduction, and the greedy reduction with and without variable compatibility optimization. Additionally it also reports the running time of the MinSatz solver for all the considered instances. Note that MinSatz could not solve 4 instances produced by the basic reduction while it was able to immediately report the answer for all the instances encoded by the greedy approach using variable compatibility.

[3] `ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/clique/`

Table 1. Example of formula sizes for the basic reduction with and without variable compatibility

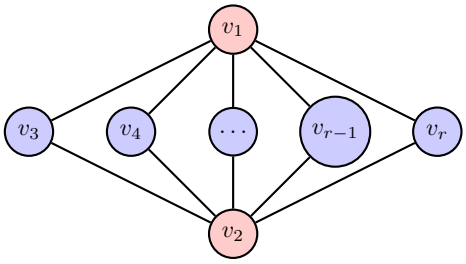
		basic			greedy			greedy+vc		
		# of vars	# of clauses	r.t.	# of vars	# of clauses	r.t.	# of vars	# of clauses	r.t.
Instance	c-fat200-1	18366	200	0.4	188	200	0.05	35	37	0
	c-fat200-2	16665	200	0.75	176	200	0.07	16	18	0
	c-fat200-5	11427	200	0.96	142	200	0.07	5	7	0
	c-fat500-1	120291	500	—	486	500	0.63	78	80	0
	c-fat500-10	78123	500	—	374	500	0.53	6	8	0
	c-fat500-2	115611	500	—	474	500	0.51	38	40	0
	c-fat500-5	101559	500	—	436	500	0.37	14	16	0

4.4 Complexity of Reducing the Number of Variables

Previous sections 4.1 and 4.2 investigated the ways to simplify CNF formulas produced by the MIS reduction to MinSAT/MaxFalse in terms of the number of used variables. Note that the number of clauses of the result CNF formulas were considered to be fixed. We also showed that in this sense all the considered improvements to the reduction are not optimal in general, i.e. the number of used variables in general is greater or equal to the minimum number of variables introduced by a *potentially optimal* encoding.

Let us formulate a problem of finding an optimal encoding as follows: given a graph $\mathcal{G}$ on r vertices construct a CNF formula $\mathcal{F}$ with exactly r clauses and a *smallest possible* number of variables. Let us refer to this problem as *MinRed*. An important question is whether there is polynomial-time algorithm for solving MinRed. Although we do not know the answer to this question, our conjecture is that there is no polynomial time algorithm for solving MinRed. A support of this conjecture is that the MinRed problem seemingly is at least as hard as the Minimum Vertex Cover problem (MVC). The basic idea of the conjecture is both MinRed and MVC being minimization problems, MVC approximates MinRed. This means that given a graph $\mathcal{G} = (V, E)$, $|V| = r$, $|E| = s$, the size μ' of its MinRed solution is *always* lower or equal to the size μ'' of any vertex cover of $\mathcal{G}$ (including its MVC).

Indeed, any vertex cover of size μ'' of the graph $\mathcal{G}$ can be used to construct a CNF formula containing exactly r clauses and μ'' variables. To do so, one can apply an algorithm similar to the greedy reduction algorithm described in Section 4.1. Such an algorithm introduces a new variable x_i for each vertex v_i in the MVC of $\mathcal{G}$ and adds literal x_i to the corresponding clause c_i of $\mathcal{F}$ while adding its complementary literal $\neg x_i$ into all clauses c_j corresponding to the vertices v_j connected to v_i. Therefore, size μ'' of the MVC can be seen as an upper bound on the minimum possible number of variables in $\mathcal{F}$, i.e. $\mu' \leq \mu''$. Moreover, there are cases where the minimum number of variables in $\mathcal{F}$ is strictly less than the minimum vertex cover of $\mathcal{G}$. An example of such a situation is shown in Figure 5. For this graph the MVC solution has size 2 (it includes vertices v_1 and v_2). However, the minimum number of variables in formula $\mathcal{F}$ corresponding to $\mathcal{G}$ (the MinRed solution for $\mathcal{G}$) is 1. It should be noted again that although MVC's upper-bounding MinRed gives an intuition that it must be at least as hard to find a solution for MinRed as to find its upper bound (a solution for MVC), we acknowledge that we do not know whether this is indeed true and there is no polytime algorithm for MinRed nor we have a proof of this fact.



(a) Graph $\mathcal{G}$ and its MVC

$$
\begin{aligned}
c_1 &= x_1 \\
c_2 &= x_1 \\
c_3 &= \neg x_1 \\
c_4 &= \neg x_1 \\
&\cdots \\
c_{r-1} &= \neg x_1 \\
c_r &= \neg x_1
\end{aligned}
$$

(b) The optimal formula $\mathcal{F}$ for $\mathcal{G}$

Fig. 5. Example of a graph $\mathcal{G}$, for which the size of the MVC of $\mathcal{G}$ is greater than the minimum number of variables in the corresponding CNF formula $\mathcal{F}$

5 Experimental Results

This section presents the results obtained in the experimental evaluation with the proposed approaches. The experiments were performed on an Intel Xeon 5160 3GHz, with 4GB of memory, and running Fedora Linux operating system.

Although the MIS problem (and the other known NP-hard graph problems) is well studied, to the best of our knowledge there are not many native MIS instances available. Therefore, the classes of benchmarks used in the evaluation are described below.

1. We considered several known sets of native *crafted* MaxClq benchmarks, namely DIMACS MaxClq, FRB, and additional MaxClq instances studied in [35, 36, 38]. All the mentioned benchmark sets comprise the *Crafted MaxClq* set of benchmarks used in our experimental evaluation. The total number of instances in the Crafted MaxClq benchmark set is 117.
2. Another considered benchmark set includes native MIS instances. These instances are obtained from the Binate Covering Problem benchmarks (BCP) since it is known that solving MIS can be seen as an approximation for BCP (e.g. see [14,57]).

The total number of instances considered is 233.

The experimental evaluation is aimed at showing that the proposed reduction from MIS to MinSAT/MaxFalse can be seen as an efficient way to solve the MIS problem. In order to do so and since there are tight relationships between MIS and MaxClq, MaxSAT and MinSAT/MaxFalse, we used several approaches to MIS/MaxClq and, thus, the corresponding classes of dedicated solvers.

1. For both benchmark sets a native MaxClq solver called *MaxCLQ* was used. It is known to be one of the best native tools for MaxClq[4] (for a comprehensive comparison of different state-of-the-art tools for MaxClq see [31, 35, 36, 38]). In order to enable MaxCLQ to deal with MIS instances, they were trivially transformed into MaxClq by complementing the graphs.
2. MinSAT/MaxFalse instances were solved by *MinSatz*, which is a known branch-and-bound MinSAT solver (see [38]). In order to produce MinSAT/MaxFalse instances, we used the greedy reduction with variable compatibility and clause duplicates removal.

[4] Although it was reported in [31] that IncMaxCLQ was the best native solver for MaxClq, we were not able to run it in our experimental evaluation.

Table 2. Number of solved instances for different approaches

—	CLQ	MinSz	MaxSz-d	MaxSz-mf	MiFuMaX-d	MiFuMaX-mf	Direct MaxSAT VBS	MaxFalse VBS	VBS
Crafted Clq	66	59	43	36	19	30	45	74	76
BCP	63	65	61	55	56	53	66	72	76
Total	129	124	104	91	75	83	111	146	152

3. Alternatively, we also transformed the MinSAT instances into MaxSAT with the use of Tseitin variables (see [55]). The following MaxSAT solvers were applied to the obtained MaxSAT instances: MaxSatz [33] and MiFuMaX [41]. MiFuMaX was chosen since it is based on the widely used Fu&Malik's core-guided algorithm for MaxSAT (e.g. see [16]). The corresponding solvers in the evaluation are called *MaxSatz-mf* and *MiFuMaX-mf*, respectively.
4. Finally, we considered MaxSAT instances that encode the MaxClq problem *directly* (without doing the MIS-to-MaxFalse transformations). The algorithm is an improved MaxClq-to-MaxSAT encoding (see [36]) and uses enumeration of disjoint independent sets. For these instances MaxSatz and MiFuMaX solvers were also used (*MaxSatz-dir* and *MiFuMaX-dir* in the evaluation, respectively).

Figure 6 shows a cactus plot illustrating the performance of the considered solvers on the total set of all instances in both Crafted MaxClq and BCP benchmark sets. The best performance overall is shown by MaxCLQ, which is able to solve 129 instances out of 233. MinSatz comes second with 124 instances solved, which is 4% less than MaxCLQ's result. MaxSAT solvers dealing with both direct MaxClq-to-MaxSAT encodings and MaxSAT instances obtained from the MIS-to-MaxFalse encodings perform significantly worse (see Figure 6).

Note that the *virtual best solver* (VBS) among the MaxSAT solvers dealing with the direct MaxClq-to-MaxSAT encoding can solve 111 instances while the VBS incorporating all the approaches based on the MinSAT/MaxFalse encodings of MIS can solve 146 instances. The VBS among all the considered approaches is able to solve 152 instances, which is only 6 instances more than the MinSAT/MaxFalse approach. Moreover, the VBS of all the approaches based on the MinSAT/MaxFalse encodings is able solve 15 instances (6.4% out of all 233 instances) that none of the other considered approaches can solve. Table 2 shows a detailed information about the number of instances solved by different approaches to MIS/MaxClq.

The experimental results indicate that the direct MaxSAT approach to the MIS and MaxClq problems has the worst performance among the considered approaches. A possible reason of the MinSAT approach being so much better than MaxSAT is that the MinSAT encoding is more compact in terms of the number of introduced variables and clauses, which is a result of the techniques proposed in the paper. Furthermore, the advantage of the MinSAT encoding is also explained by the fact that it does not contain any hard clauses while the known MaxSAT encodings do use a large number of hard clauses. Although the best performance is shown by MaxCLQ, which is a native MaxClq algorithm, MinSatz is very close to MaxCLQ solving 4% fewer instances. It is also interesting that the MinSAT/MaxFalse approach can solve 6.4% instances that cannot be solved by other approaches (i.e. native MaxCLQ and direct MaxSAT). Moreover, although there are not many papers on MinSAT solving (especially if compared to MaxSAT), considering the virtual best solvers shows that the proposed MinSAT/MaxFalse approach to the MIS problem is a promising way to deal with the MIS/MaxClq problems.

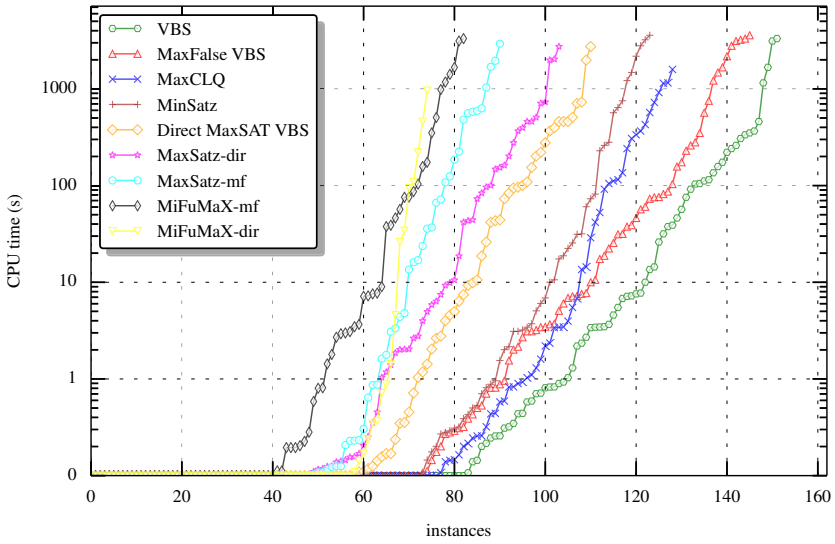


Fig. 6. Cactus plot showing the performance of the considered approaches

6 Conclusions

A number of new algorithms for the MinSAT problem have been proposed in recent years [19, 34, 37, 38]. Nevertheless, besides being used in problem reductions, MinSAT has seldom been used for modeling and solving combinatorial optimization problems, the exception being [3]. This paper represents another step towards identifying combinatorial optimization problems which can be solved with MinSAT. The paper proposes a reduction from MIS (and so from related NP-hard graph problems) to MinSAT. The paper also develops a number of heuristics to reduce the obtained MinSAT formulas. In practice, the proposed techniques are very effective, and allow compacting the original MinSAT formulas to a fraction of their original size. The experimental results show that the obtained MinSAT formulas, solved with a standard MinSAT solver, allow obtaining results that are comparable to native solvers for MaxClq instances, and outperform those solvers on actual MIS instances. Moreover, the use of MinSAT comprehensively outperforms approaches based on using a reduction to MaxSAT. In addition, the results of the VBS solvers suggest that portfolios of solvers could significantly outperform a standalone solver. Overall, the experimental results are promising, and motivate the development of more efficient MinSAT solvers.

Future research work will focus on further improvements to the proposed MinSAT models. Another area of research is to implement portfolios of solvers for NP-hard graph problems, by exploiting some of the reductions proposed in recent years (including the ones in this paper). Finally, another area of research is to develop more efficient MinSAT solvers, e.g. similar to what has been done in the MaxSAT area in recent years.

Acknowledgments. This work is partially supported by SFI PI grant BEACON (09-/IN.1/I2618), FCT grant POLARIS (PTDC/EIA-CCO/123051/2010), and INESC-ID's multiannual PIDDAC funding PEst-OE/EEI/LA0021/2013.

References

1. Andrade, D.V., Resende, M.G.C., Werneck, R.F.F.: Fast local search for the maximum independent set problem. J. Heuristics 18(4), 525–547 (2012)
2. Ansotegui, C., Li, C.M., Manya, F., Zhu, Z.: A SAT-based approach to MinSAT. In: Escrig, M.T., Toledo, F.J., Golobardes, E. (eds.) CCIA 2002. LNCS (LNAI), vol. 2504, pp. 185–189. Springer, Heidelberg (2002)
3. Argelich, J., Li, C.-M., Manyà, F., Zhu, Z.: MinSAT versus MaxSAT for optimization problems. In: Schulte, C. (ed.) CP 2013. LNCS, vol. 8124, pp. 133–142. Springer, Heidelberg (2013)
4. Battiti, R., Protasi, M.: Reactive local search for the maximum clique problem. Algorithmica 29(4), 610–637 (2001)
5. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (2009)
6. Bomze, I.M., Budinich, M., Pardalos, P.M., Pelillo, M.: The maximum clique problem. In: Handbook of Combinatorial Optimization, pp. 1–74. Springer (1999)
7. Cai, S., Su, K., Chen, Q.: EWLS: A new local search for minimum vertex cover. In: Fox, M., Poole, D. (eds.) AAAI. AAAI Press (2010)
8. Cai, S., Su, K., Luo, C., Sattar, A.: NuMVC: An efficient local search algorithm for minimum vertex cover. J. Artif. Intell. Res. (JAIR) 46, 687–716 (2013)
9. Cai, S., Su, K., Sattar, A.: Local search with edge weighting and configuration checking heuristics for minimum vertex cover. Artif. Intell. 175(9-10), 1672–1696 (2011)
10. Cai, S., Su, K., Sattar, A.: Two new local search strategies for minimum vertex cover. In: Hoffmann, J., Selman, B. (eds.) AAAI. AAAI Press (2012)
11. Carraghan, R., Pardalos, P.M.: An exact algorithm for the maximum clique problem. Operations Research Letters 9(6), 375–382 (1990)
12. Chamaret, B., Josselin, S., Kuonen, P., Pizarroso, M., Salas-Manzanedo, B., Ubeda, S., Wagner, D.: Radio network optimization with maximum independent set search. In: IEEE 47th Vehicular Technology Conference, 1997, vol. 2, pp. 770–774 (May 1997)
13. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms. The MIT press (2009)
14. Coudert, O.: On solving covering problems. In: DAC, pp. 197–202 (1996)
15. Fahle, T.: Simple and fast: Improving a branch-and-bound algorithm for maximum clique. In: Möhring, R., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 485–498. Springer, Heidelberg (2002)
16. Fu, Z., Malik, S.: On solving the partial MAX-SAT problem. In: Biere, A., Gomes, C.P. (eds.) SAT 2006. LNCS, vol. 4121, pp. 252–265. Springer, Heidelberg (2006)
17. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)
18. Gavril, F.: Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. SIAM J. Comput. 1(2), 180–187 (1972)
19. Heras, F., Morgado, A., Planes, J., Marques-Silva, J.: Iterative SAT solving for minimum satisfiability. In: ICTAI, pp. 922–927 (2012)
20. Heras, F., Morgado, A., Planes, J., Marques-Silva, J.: Iterative sat solving for minimum satisfiability. In: ICTAI, vol. 1, pp. 922–927. IEEE (2012)
21. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to automata theory, languages, and computation - international edition, 2nd edn. Addison-Wesley (2003)
22. Ignatiev, A., Morgado, A., Planes, J., Marques-Silva, J.: Maximal falsifiability: Definitions, algorithms, and applications. In: LPAR, pp. 439–456 (2013)
23. Jain, K., Padhye, J., Padmanabhan, V.N., Qiu, L.: Impact of interference on multi-hop wireless network performance. Wireless Networks 11(4), 471–487 (2005)

24. Johnson, D.S., Papadimitriou, C.H., Yannakakis, M.: On generating all maximal independent sets. Inf. Process. Lett. 27(3), 119–123 (1988)
25. Joseph, D., Meidanis, J., Tiwari, P.: Determining dna sequence similarity using maximum independent set algorithms for interval graphs. In: Nurmi, O., Ukkonen, E. (eds.) SWAT 1992. LNCS, vol. 621, pp. 326–337. Springer, Heidelberg (1992)
26. Kohavi, Z.: Switching and Finite Automata Theory. Tata McGraw-Hill (1978)
27. Kohli, R., Krishnamurti, R., Mirchandani, P.: The minimum satisfiability problem. SIAM J. Discrete Math. 7(2), 275–283 (1994)
28. Konc, J., Janezic, D.: An improved branch and bound algorithm for the maximum clique problem. In: MATCH, vol. 58, pp. 560–590 (2007)
29. Kügel, A.: Natural Max-SAT encoding of Min-SAT. In: Hamadi, Y., Schoenauer, M. (eds.) LION 2012. LNCS, vol. 7219, pp. 431–436. Springer, Heidelberg (2012)
30. Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R.: Generating all maximal independent sets: NP-hardness and polynomial-time algorithms. SIAM J. Comput. 9(3), 558–565 (1980)
31. Li, C.M., Fang, Z., Xu, K.: Combining maxsat reasoning and incremental upper bound for the maximum clique problem. In: ICTAI, pp. 939–946 (2013)
32. Li, C.M., Manya, F.: MaxSAT, hard and soft constraints. In: Biere, et al. (eds.) [5], pp. 613–631
33. Li, C.M., Manyà, F., Planes, J.: New inference rules for max-sat. J. Artif. Intell. Res.(JAIR) 30, 321–359 (2007)
34. Li, C.M., Manyà, F., Quan, Z., Zhu, Z.: Exact MinSAT solving. In: Strichman, O., Szeider, S. (eds.) SAT 2010. LNCS, vol. 6175, pp. 363–368. Springer, Heidelberg (2010)
35. Li, C.M., Quan, Z.: Combining graph structure exploitation and propositional reasoning for the maximum clique problem. In: ICTAI, pp. 344–351 (2010)
36. Li, C.M., Quan, Z.: An efficient branch-and-bound algorithm based on maxsat for the maximum clique problem. In: AAAI, vol. 10, pp. 128–133 (2010)
37. Li, C.M., Zhu, Z., Manya, F., Simon, L.: Minimum satisfiability and its applications. In: IJCAI, pp. 605–610 (2011)
38. Li, C.M., Zhu, Z., Manya, F., Simon, L.: Optimizing with minimum satisfiability. Artif. Intell. 190, 32–44 (2012)
39. Manquinho, V.M., Silva, J.P.M.: Satisfiability-based algorithms for boolean optimization. Ann. Math. Artif. Intell. 40(3-4), 353–372 (2004)
40. Marathe, M.V., Ravi, S.S.: On approximation algorithms for the minimum satisfiability problem. Inf. Process. Lett. 58(1), 23–29 (1996)
41. MiFuMaX — a Literate MaxSAT Solver, `http://sat.inesc-id.pt/~mikolas/sw/mifumax/book.pdf` (accessed: January 31, 2014)
42. Östergård, P.R.J.: A fast algorithm for the maximum clique problem. Discrete Applied Mathematics 120(1-3), 197–207 (2002)
43. Pardalos, P.M., Xue, J.: The maximum clique problem. Journal of Global Optimization 4(3), 301–328 (1994)
44. Pullan, W.J.: Approximating the maximum vertex/edge weighted clique using local search. J. Heuristics 14(2), 117–134 (2008)
45. Pullan, W.J., Hoos, H.H.: Dynamic local search for the maximum clique problem. J. Artif. Intell. Res. (JAIR) 25, 159–185 (2006)
46. Ramaswami, R., Sivarajan, K.N.: Routing and wavelength assignment in all-optical networks. IEEE/ACM Trans. Netw. 3(5), 489–500 (1995)
47. Régin, J.-C.: Using constraint programming to solve the maximum clique problem. In: Rossi, F. (ed.) CP 2003. LNCS, vol. 2833, pp. 634–648. Springer, Heidelberg (2003)
48. Resende, M.G.C., Feo, T.A., Smith, S.H.: Algorithm 787: Fortran subroutines for approximate solution of maximum independent set problems using GRASP. ACM Trans. Math. Softw. 24(4), 386–394 (1998)

49. Robson, J.M.: Algorithms for maximum independent sets. J. Algorithms 7(3), 425–440 (1986)
50. San Segundo, P., Rodríguez-Losada, D., Jiménez, A.: An exact bit-parallel algorithm for the maximum clique problem. Computers & Operations Research 38(2), 571–581 (2011)
51. Tarjan, R.E., Trojanowski, A.E.: Finding a maximum independent set. SIAM J. Comput. 6(3), 537–546 (1977)
52. Tomita, E., Kameda, T.: An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. Journal of Global Optimization 37(1), 95–111 (2007)
53. Tomita, E., Seki, T.: An efficient branch-and-bound algorithm for finding a maximum clique. In: Calude, C.S., Dinneen, M.J., Vajnovszki, V. (eds.) DMTCS 2003. LNCS, vol. 2731, pp. 278–289. Springer, Heidelberg (2003)
54. Tomita, E., Sutani, Y., Higashi, T., Takahashi, S., Wakatsuki, M.: A simple and faster branch-and-bound algorithm for finding a maximum clique. In: Rahman, M. S., Fujita, S. (eds.) WALCOM 2010. LNCS, vol. 5942, pp. 102–112. Springer, Heidelberg (2010)
55. Tseitin, G.S.: On the complexity of derivation in propositional calculus. Studies in Constructive Mathematics and Mathematical Logic 2(115-125), 10–13 (1968)
56. Tsukiyama, S., Ide, M., Ariyoshi, H., Shirakawa, I.: A new algorithm for generating all the maximal independent sets. SIAM J. Comput. 6(3), 505–517 (1977)
57. Villa, T., Kam, T., Brayton, R.K., Sangiovanni-Vincentelli, A.L.: Explicit and implicit algorithms for binate covering problems. IEEE Trans. on CAD of Integrated Circuits and Systems 16(7), 677–691 (1997)
58. Zhu, Z., Li, C.-M., Manyà, F., Argelich, J.: A new encoding from MinSAT into MaxSAT. In: Milano, M. (ed.) CP 2012. LNCS, vol. 7514, pp. 455–463. Springer, Heidelberg (2012)