Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 47 (2026), No. 2, pp. 399–427
DOI : 10.47974/JIOS-1392

Two paradigms for combining optimization and satisfiability : Maximum satisfiability and optimum satisfiability problems

Anna Konovalenko [†]
Lars Magnus Hvattum [*]
Sebastián Urrutia [§]
Faculty of Logistics
Molde University College
P. O. Box 2110
N-6402 Molde
Norway

Abstract

Maximum satisfiability (MaxSAT) and optimum satisfiability (OptSAT) problems are two optimization versions of the NP-complete Boolean satisfiability problem. In the literature, these versions have been described and tackled separately by using specialized solvers for each problem. This paper investigates a connection between MaxSAT and OptSAT where each problem can be reduced to the other. We consider several heuristic solvers and different sets of benchmark instances for each problem and investigate whether one class of solvers can tackle instances of the other problem with competitive results. Through computational experiments, we conclude that specialized solvers for one of the problems cannot compete with specialized solvers for the other problem and point out potential improvements of the solvers that are needed to successfully address a wider range of problem instances.

Subject Classification: *(2010) 90C10 Integer programming, 90C27 Combinatorial optimization.*

Keywords: *Boolean optimization problem, Minimum cost satisfiability, Binary integer programming, Heuristic.*

1. Introduction

The Boolean satisfiability problem (SAT) is the problem of determining whether there exists an assignment of truth values to a set of Boolean

[†] *E-mail:* **anna.konovalenko@himolde.no**
[*] *E-mail:* **hvattum@himolde.no (Corresponding Author)**
[§] *E-mail:* **sebastian.urrutia@himolde.no**

variables that satisfies a given Boolean formula. A closely related optimization problem is the maximum satisfiability problem (MaxSAT) which is obtained by considering a Boolean formula consisting of clauses that must not necessarily be satisfied, but where the goal is to determine truth values so as to maximize the sum of weights for satisfied clauses. Another closely related optimization problem is the optimum satisfiability problem (OptSAT), in which weights are attached directly to variables and the sum of the weights of true-valued variables is maximized while all clauses are satisfied.

In the scientific literature, MaxSAT and OptSAT have been addressed as two separate, non-overlapping problems. They are indeed different, in that their objective functions are defined differently. However, the two problems share significant similarities, in particular due to their satisfiability-based structure.

We investigate the relationship between MaxSAT and OptSAT by providing reductions from each problem to the other. Then, we consider several specialized solvers for each problem, as well as different sets of benchmark instances for the two problems, respectively. Through computational experiments we investigate whether one class of solvers can effectively tackle instances of both types of problems, or whether the benchmark instances do require specialized solvers to obtain competitive results. The latter would indicate that the two optimization variants of SAT are sufficiently different to warrant specialized solvers, but may also give some hints as to how current-day specialized solvers can be improved to handle more general instances.

SAT is a well-known decision problem that has a wide range of applications in logic, computer science, engineering, graph theory, and operational research. Previous studies have shown that many practical problems can be formulated and solved as SAT. This includes studies from various fields such as electronic design automation [28, 37], network security management [14], scheduling [3, 11], robotics [18] and database design [12]. SAT is NP-complete [4] and there have been significant conceptual advances in heuristic SAT solvers over the past decades.

Researchers have also studied a variety of special cases and generalizations of SAT. In this work, we consider two different optimization versions of SAT. The first one, MaxSAT, has been considered when solving real-life optimization problems in electrical engineering [36] and computer science [33]. As a result, this problem has been of ever-growing interest for scientists in these fields. In 2006, this led to the establishment of an annual

MaxSAT solvers' evaluation, where participants have demonstrated consistent improvements in the solvers' performances [1].

The second problem, OptSAT, can be used to express large classes of binary optimization models, such as set covering, graph stability, and set partitioning problems. It was first introduced by Davoine et al. [6], where the problem was referred to as the Boolean optimization problem. Solvers with improved performances were subsequently presented by Hvattum et al. [16], Hvattum et al. [17] and da Silva et al. [5].

In a parallel line of research, a variant known as the minimum cost satisfiability problem (MinCostSAT) has been studied. This variant is identical to OptSAT, with the exception that it is formulated as a minimization problem rather than a maximization problem. Li [25] presented MinCostSAT as a general problem with special cases including set covering and binate covering problems [13]. The author also investigated the performance of specialized MinCostSAT solvers, which were later studied and updated by Fu and Malik [8] and Li et al. [24].

OptSAT and its MinCostSAT variant are both special cases of the more general pseudo-Boolean optimization problem (PBO). In the PBO problem weights are also attached directly to the variables, but clauses are converted into a special type of constraints with binary variables, known as cardinality constraints [34]. Much research has been performed in this field, including algorithms that can tackle both MaxSAT and PBO problems [26]. Furthermore, the PBO problem is a special case of the binary integer programming problem, which is a special case of mixed integer programming problems. Nadel [31], in turn, considered the optimization of a generic objective function subject to a satisfiable propositional formula, a version that incidentally was also abbreviated as OptSAT. Rather than climbing this hierarchy of generalizations, we focus specifically on the formulations of OptSAT and MaxSAT and the state-of-the-art heuristic solvers for these two problem classes.

The remainder of this paper is structured as follows. In Section 2, the problem formulations are presented. Section 3 describes problem reductions from MaxSAT to OptSAT as well as from OptSAT to MaxSAT. In Section 4, we describe the experimental setup. Results of computational experiments are reported in Section 5 and conclusions are given in Section 6.

2. Problem formulations

2.1 *Preliminaries*

SAT is a canonical decision problem, which aims to determine whether or not there exists a set of truth values such that a given logical expression $\phi(x_1,\ldots,x_n)$ is satisfied. The expression is defined in propositional, first-order logic, based on Boolean variables.

A Boolean variable x_j or its negation $\overline{x_j}$ is called a literal. A disjunction of literals is known as a clause C_i. The Boolean expression $\phi(x_1,\ldots,x_n)$ is in conjunctive normal form (CNF) if it is a conjunction of one or more clauses. Based on laws of logical equivalences, every logical expression can be converted into an equivalent formula that is in CNF. This allows $\phi(x_1,\ldots,x_n)$ to be expressed in the form:

$$\phi = C_1 \wedge \ldots \wedge C_m,$$

where each clause C_i is a disjunction of some Boolean variables x_j, $j \in A_i \subset \{1,\ldots,n\}$, and some negated variables $\overline{x_j}$, $j \in B_i \subset \{1,\ldots,n\}$:

$$C_i = (\bigvee_{j \in A_i} x_j) \vee (\bigvee_{j \in B_i} \overline{x_j}).$$

2.2 *The weighted partial maximum satisfiability problem*

The weighted partial maximum satisfiability problem, hereafter simply referred to as MaxSAT, considers two types of clauses, hard and soft. Hard clauses C_i, for $i \in \{1,\ldots,m\}$, must be satisfied, while the other clauses C_i, for $i \in \{m+1,\ldots,m+q\}$, are soft and associated with a positive reward w_i, when being satisfied. The objective of the problem is to find an assignment that maximizes the sum of weights of satisfied soft clauses, while satisfying all hard clauses. That is, one must satisfy:

$$\phi(x_1,\ldots,x_n) = C_1 \wedge \ldots \wedge C_m = true.$$

The objective of MaxSAT can be stated as

$$\max z = \sum_{i=m+1}^{m+q} f(C_i),$$

where $f : \{C_{m+1},\ldots,C_{m+q}\} \mapsto \mathbb{R}^+$ is a function defined as

$$f(C_i) = \begin{cases} w_i, & \text{if } C_i = true, \\ 0, & \text{if } C_i = false. \end{cases}$$

Applying the following transformations, MaxSAT can be converted to an integer linear programming problem (ILP). Each soft clause C_i is associated with an auxiliary variable $y_i \in \{0,1\}$ such that $y_i = 1$ if clause C_i is satisfied and $y_i = 0$, otherwise. A clause can be represented as a separate constraint by replacing Boolean variables with binary variables (*true* = 1, *false* = 0), disjunction with +, and a negated variable $\overline{x_j}$ with $(1 - x_j)$. This results in the following model:

$$
\begin{aligned}
&\max z = \sum_{i=m+1}^{m+q} w_i y_i, \\
&\sum_{j \in A_i} x_j + \sum_{j \in B_i} (1 - x_j) \geq 1, && i \in \{1, \ldots, m\}, \\
&\sum_{j \in A_i} x_j + \sum_{j \in B_i} (1 - x_j) \geq y_i, && i \in \{m+1, \ldots, m+q\}, \\
&x_j \in \{0,1\}, && j \in \{1, \ldots, n\}, \\
&y_i \in \{0,1\}, && i \in \{m+1, \ldots, m+q\}.
\end{aligned}
$$

2.3 *The optimum satisfiability problem*

OptSAT is also constrained by a Boolean expression in CNF:

$$\phi(x_1, \ldots, x_n) = C_1 \wedge \ldots \wedge C_m = true.$$

Each original variable x_j is associated with a positive value v_j, which is awarded when the variable takes the value *true*. The objective of OptSAT can be written as:

$$\max z = \sum_{j=1}^{n} g(x_j),$$

where $g : \{x_1, \ldots, x_n\} \mapsto \mathbb{R}^+$ is a function such that

$$g(x_j) = \begin{cases} v_j, & \text{if } x_j = true, \\ 0, & \text{if } x_j = false. \end{cases}$$

The ILP formulation of the OptSAT has a similar structure to that of MaxSAT, except that no auxiliary variables are needed for the clauses. It can be defined as follows:

$$\begin{aligned} \max z &= \sum_{j=1}^{n} v_j x_j, \\ \sum_{j \in A_i} x_j + \sum_{j \in B_i} (1 - x_j) &\geq 1, && i \in \{1, \ldots, m\}, \\ x_j &\in \{0,1\}, && j \in \{1, \ldots, n\}. \end{aligned}$$

The solution to OptSAT is the set of truth assignments for the variables that yields the highest objective function value z, while satisfying the Boolean expression ϕ. The MinCostSAT variant instead minimizes a weighted sum of the true variables. This can be converted to the presented form in two steps. First, the objective function is negated. Second, variables x_j for which $g(x_j)$ now becomes negative, are substituted by $x'_j = \overline{x}_j$, while setting $g(x'_j) = -g(x_j)$. The convertion in the ILP formulation can be seen in the following example.

Consider the MinCostSAT with the objective function $\min z = \sum_{j=1}^{n} v_j x_j$. Then, the maximization problem is $\max -z = -\sum_{j=1}^{n} v_j x_j$. We can set $x_j = 1 - x'_j$, which leads to the following transformations: $\max -z = -\sum_{j=1}^{n} v_j x_j = -\sum_{j=1}^{n} v_j (1 - x'_j) = -\sum_{j=1}^{n} v_j + \sum_{j=1}^{n} v_j x'_j$. This shifts the objective function value by a constant amount $\sum_{j=1}^{n} v_j$, which must then be taken into account when recovering the original objective function value of MinCostSAT.

3. Reductions

This section describes how to transform a MaxSAT instance into an OptSAT instance, and vice versa. Transformations between different variants of satisfiability problems are well known [7], and we present the details here for the sake of completeness.

3.1 *From MaxSAT to OptSAT*

Generally, MaxSAT and OptSAT are similar to each other, except that weights in MaxSAT are attached to the clauses rather than to the variables. A MaxSAT instance can be converted to OptSAT by introducing an additional variable x_{n+i} for each soft clause C_{m+i}, $i \in \{1, \ldots, q\}$ to store information about weights. Hard clauses C_i, $i \in \{1, \ldots, m\}$ remain unchanged.

A MaxSAT instance with hard clauses $\phi(x_1, \ldots, x_n) = C_1 \wedge \ldots \wedge C_m$ can be converted to an OptSAT instance by first extending ϕ to $\phi(x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+q})$ in the following way:

$$\phi = C_1 \wedge \ldots \wedge C_m \wedge (C_{m+1} \vee \overline{x_{n+1}}) \wedge \ldots \wedge (C_{m+q} \vee \overline{x_{n+q}}) = true.$$

Next, the objective function coefficients associated with the original variables x_j, $j \in \{1,\ldots,n\}$, are set to $v_j = 0$ while $v_{n+i} = w_{m+i}$, $i \in \{1,\ldots,q\}$, for the additional variables x_{n+i}.

Proposition 1: Consider an instance of MaxSAT and a corresponding instance of OptSAT, obtained by the transformation given above. Consider optimal solutions for the two instances, respectively. The sum of the weights of satisfied clauses in the optimal MaxSAT solution is identical to the sum of weights of true variables in the optimal OptSAT solution.

Proof: Consider an optimal solution to the MaxSAT instance. We can create a corresponding solution to the OptSAT instance by taking the same truth values for variables $x_1,\ldots,x_n$. The truth values of the additional variables, associated to each of the soft clauses, are set to *true* for the satisfied clauses and *false* for the unsatisfied clauses. Then, the modified ϕ for the OptSAT instance is satisfied, and the objective function value of the OptSAT solution is equal to the sum of weights for the satisfied clauses in the MaxSAT instance.

Next, consider an optimal solution of the OptSAT instance. We can create a corresponding solution to the MaxSAT instance by using the same truth values for variables $x_1,\ldots,x_n$. Due to the optimality of the OptSat instance, the soft clauses satisfied in the MaxSAT instance are exactly those where the corresponding variable x_{n+i} for $i \in \{1,\ldots,q\}$ is true in the OptSAT solution. Therefore, the sum of weights of satisfied clauses in the MaxSAT instance is equal to the sum of weights of true variables in the OptSAT instance.

Finally, if an optimal MaxSAT solution does not exist, meaning that the MaxSAT instance is infeasible, then neither does an optimal OptSAT solution exist, as the hard clauses are identical in both instances. The same argument holds in the other direction. Correspondingly, if an optimal OptSAT solution does not exist, then it must be due to the hard constraints from the original MaxSAT instance, since otherwise a solution with $x_{n+i} = false$ for $i \in \{1,\ldots,q\}$, should exist.

3.2 *From OptSAT to MaxSAT*

For transforming an OptSAT instance to a MaxSAT instance we propose adding a new soft unit clause for each of the original variables. In

particular, a soft clause C_{m+j} is introduced for each variable x_j, $j \in \{1,\ldots,n\}$, such that $C_{m+j} = x_j$. Hard clauses are given by the original clauses C_i, $i \in 1,\ldots,m$, from the OptSAT instance:

$$\phi(x_1,\ldots,x_n) = C_1 \wedge \ldots \wedge C_m = true.$$

The original clauses from the OptSAT instance do not influence the objective function, as all of them are defined as hard clauses. However, the weight v_j of the original variable x_j is represented as objective function coefficients $w_{m+j} = v_j$, $j \in \{1,\ldots,n\}$, associated with the soft unit clauses C_{m+j}.

Proposition 2: Consider an instance of OptSAT and a corresponding instance of MaxSAT, obtained by the transformation given above. Consider optimal solutions for the two instances, respectively. The sum of weights of true variables in the optimal OptSAT solution is identical to sum of the weights of satisfied clauses in the optimal MaxSAT solution.

Proof: Consider an optimal solution to the OptSAT instance. We can create a corresponding solution to the MaxSAT instance by taking the same truth values for the variables $x_1,\ldots,x_n$. All the clauses in the original OptSAT instance are then satisfied, which means all the hard clauses in the MaxSAT instance are satisfied. The soft clause C_{m+j} in MaxSAT is satisfied iff $x_j = true$. Thus the objective function value of the MaxSAT solution is equal to the objective function value of the OptSAT solution.

Next, consider an optimal solution of the MaxSAT instance. We can again create a corresponding solution to the OptSAT instance by using the same truth values for all variables $x_1,\ldots,x_n$. Since all hard clauses in MaxSAT must be satisfied, this solution is feasible for the OptSAT instance. Furthermore, they have the same objective function values, since a soft clause C_{m+j} in the MaxSAT instance is satisfied iff $x_j = true$.

Finally, if an optimal OptSAT solution does not exist, meaning that the OptSAT instance is infeasible, then the hard clauses in the MaxSAT instance cannot be satisfied together either. Conversely, if no optimal MaxSAT solution exists, then the hard clauses in the MaxSAT cannot be satisfied, and neither can the original clauses in the OptSAT Instance.

4. Experimental setup

Our aim is to investigate the ability of specialized MaxSAT and specialized OptSAT solvers to handle instances originating from the two

different optimization variants of SAT, respectively. We present the best heuristic solvers for both problems and their results for instances of each problem. At the same time, a general-purpose mixed-integer programming solver is executed using the ILP formulation for each instance, thereby providing bounds to assess the hardness of solving the instances and to benchmark the specialized solvers. This section describes the solvers tested, the instances used, and how tests are conducted.

4.1 *Solvers*

MaxSAT solvers are taken from the 15th annual edition of a competition known as the MaxSAT Evaluation, organized in 2021. The three best solvers competing in the class of heuristic solvers are used in our computational study: TT-Open WBO-Inc-20 [29], SATLike-c(w) [23], and Open-WBO-Inc-complete [21].

The solver TT-Open WBO-Inc-20 is based on the Open-WBO solver, which was introduced in 2014 as a modular open-source exact solver [35]. The Open-WBO solver relies on successive calls to an underlying SAT solver [32] and is special compared to other solvers due to an opportunity to easily replace the underlying SAT solver. There are two MaxSAT algorithms in the Open-WBO solver: an unsatisfiability-based algorithm, based on the iterated use of a SAT solver to identify unsatisfiable subformulas [9], and a linear search algorithm (LSU), an algorithm which adds a new relaxation variable to each soft clause and solves the resulting formula with a SAT solver [2]. We note how this handling of the problem resembles the conversion of a MaxSAT instance into an OptSAT instance, as outlined above. Thus, one could expect that this solver is also very effective at solving OptSAT instances.

After the performance of the Open-WBO solver as one of the best solvers, the solver was transformed to a quick heuristic solver Open-WBO-Inc [20]. Lately, the Open-WBO-Inc solver was improved by adding more advanced heuristics for variable selection and tuning variable values [30] as well as optimizing the generic solution [31]. The methods were integrated into the TT-Open WBO-Inc and the TT-Open WBO-Inc-20 solvers, respectively.

The second solver SATLike-c(w) combines two major solvers: the local search solver SATLike [22] and the previously introduced TT-Open-WBO-Inc solver. An underlying SAT solver is executed to find a feasible solution which further is taken as an initial solution for the SATLike solver. SATLike keeps working until it fails to improve the current solution within

a given time limit. Afterwards, TT-Open-WBO-Inc is executed to improve the current solution.

The third MaxSAT solver is Open-WBO-Inc-complete, which is a version of the Open-WBO-Inc-BMO solver [19]. The Open-WBO-Inc-BMO solver uses linear search on clauses, which are clustered by weights [20]. Based on bounded multilevel optimization (BMO) on each cluster [27], it performs a sequence of calls to a SAT solver and updates an upper bound on the number of unsatisfied soft clauses. Once the upper bound cannot be improved for a selected cluster, the next cluster is taken until all clusters are viewed. In its turn, the Open-WBO-Inc-complete solver runs the Open-WBO-Inc-BMO solver first and if an optimal solution is found in terms of the BMO assumption, it switches to the LSU algorithm in order to search for a better bound.

There are two OptSAT solvers used in our study. The TS-ACW is a heuristic solver for the OptSAT problem based on an advanced tabu search [10] with adaptive clause weights. The search is designed to be able to move in infeasible space by penalizing an infeasibility measure. The TS-ACW beat the previously developed methods for OptSAT, both in terms of solution quality and solution time [17].

The second OptSAT solver Evol-TS [5] combines the well-performing tabu search with an evolutionary heuristic. It starts by creating an initial reference set of solutions. Then, it iteratively selects a random subset of solutions to be combined. The variables with common values in the combined solutions are fixed, and the TS-ACW is used to search for improved solutions over the remaining free variables. If the new solution is different from all solutions in the reference set, it is added to the reference set.

Finally, the commercial MIP-solver Gurobi is used to solve the ILP-formulations of both the MaxSAT and the OptSAT problems.

4.2 *Instances*

Instances for the MaxSAT problem are taken from the annual MaxSAT Evaluation of solvers (https://maxsat-evaluations.github.io/2020). Out of 28 different classes of instances used to assess the performance of heuristic solvers, we manually selected eight. From these, a total of 50 instances with different sizes and structures are used in our computational experiments. The instances are summarized in Appendix A.

There are three sources of OptSAT instances. The earliest instances were provided by Davoine et al. [6]. Instances are combined in classes,

where each class consists of five instances. Classes 1-49 are random instances, classes 50-54 are based on graph stability problems, while classes 55-63 are based on set covering problems. Instances in classes 64-69 were generated later by Silva et al. [5] to check the performance of combining tabu search and evolutionary heuristics. The third source of instances for the OptSAT is Fu and Malik [8], who presented instances in the form of MinCostSAT. These instances are in turn representing three different types of problems: the bounded model checking problem, a graph coloring problem, and artificial intelligent planning problems [15]. Weights for all variables in the MinCostSAT instances are equal to one. In total, 50 instances of the OptSAT problem are selected and their details are provided in Appendix A.

4.3 *Setup*

All the experiments are conducted on a MacBook Air, Apple M1 Chip with 8-Core CPU, 8-Core GPU and 16-core Neural Engine, using the operating system macOS Big Sur, version 11.2.3. Solvers were compiled using g++ (gxx c++ 4.2.1) with Apple clang version 12.0.5 and the optimization flag -O3. Gurobi version 9.1.2 (mac64) is used as the mixed-integer programming solver.

Two different time limits are considered for all the solvers, namely 60 seconds and 600 seconds. These time limits are also traditionally used in the MaxSAT Evaluation competition. The MaxSAT solvers and the OptSAT solvers were run five times per instance, with default parameters and different random seeds, respectively.

5. Computational results

Computational results for MaxSAT instances are shown in Table 1 and Table 2 and for OptSAT instances in Table 3 and Table 4.

For each of the instances, we give the cost of solutions for the three different MaxSAT solvers in columns 4-6. The results in the tables represent the average objective function values over five runs. The objective function value is the sum of weights of satisfied soft clauses. If a MaxSAT solver fails to find a feasible solution on an instance, in all of the five runs, the result is reported as NA.

The results of the two OptSAT solvers are given in columns 7 and 8. When an OptSAT solver finds a feasible solution for all five runs for an instance, the average objective function value is reported. In the case that

an OptSAT solver fails to find a feasible solution, we instead report the average number of unsatisfied clauses, and this is indicated with an asterisk after the number. Finally, for some instances the OptSAT solvers may run out of memory and thus not produce any results. If this happens to all five runs, the result is given as NA, and otherwise the number of successful runs is given in parentheses.

For most of the instances, the MaxSAT solvers show the same result in all five runs. For a few large instances, the obtained values vary slightly. The OptSAT solvers' values varies at most 4 % from the average for different random seeds.

For the mixed-integer programming solver, we report both the primal bound (L.B.) and the dual bound (U.B.) obtained at the end of the run. The primal bound corresponds to a feasible solution found during the search, while the dual bound corresponds to the value of the best open node in the branch-and-bound search tree. Some instances cannot be run using this solver due to memory limitations, in which case the result is given as NA. The best overall results for each instance are highlighted in a bold font.

Table 1

Comparison of solvers on MaxSAT instances, 60 seconds.

	Gurobi		MaxSAT Solvers			OptSAT Solvers	
			TT-Open WBO-Inc-20	SATLike-c(w)	Open-WBO-Inc-complete		
	L.B.	U.B.				TS-ACW	Evol-TS
MA1	17507225	17516939	17505973	**17507601**	17506899	5*	30*(1)
MA2	**21079364**	21091716	21078680	21079230	21078091	6*	24*(2)
MA3	**8940006**	89460545	8936045	8939403	8936600	4*	86*(3)
MA4	11707529	11718019	11701995	**11707913**	11702166	5*	63*(3)
MA5	14172952	14184101	14172638	**14175429**	14171934	7*	80*(4)
MB1	502640	505267	503511	**503850**	503434	502754	502361
MB2	901665	909328	903663	**904444**	903634	900851	900023
MB3	1507610	1526670	1512217	**1516299**	1513070	1508610	1506310
MB4	2406500	2443250	2415938	**2422587**	2415186	2407437	2405531
MB5	3060410	3064080	3063254	**3063613**	3063489	3062097	3061933
MC1	NA	528689	**525570**	**525570**	506995	1652*	5335*(2)
MC2	NA	379105	NA	NA	NA	2205*	11907*(4)

Contd...

MC3	NA	195321	**193562**	193561	192997	921*	4039*
MC4	NA	23554	**23247**	**23247**	**23247**	896*	5765*(4)
MC5	NA	484833	NA	NA	NA	2247*	11255*(2)
MC6	NA	96716	**86715**	81944	83249	379*	329*(2)
MC7	NA	9353	**7723**	7675	7675	1416*	8017*(2)
MD1	NA	NA	**505317**	**505317**	**505317**	505267	504930
MD2	NA	NA	**505937**	**505937**	**505937**	505852	505585
MD3	NA	NA	**506562**	**506562**	**506562**	506488	506199
MD4	NA	NA	**507413**	**507413**	**507413**	507405	507377
MD5	NA	NA	502225	502226	**502227**	502221	502182
ME1	**4653**	4697	4550	**4653**	4504	4609	4566
ME2	**5157**	5253	5049	**5157**	5068	5106	5061
ME3	**4573**	4608	4510	**4573**	4435	4540	4497
ME4	15732	17024	15599	**15785**	15542	15527	15472
ME5	16238	17279	15867	**16317**	15897	16026	16024
ME6	**5967**	6154	5914	**5967**	5784	5897	5832
ME7	5701	5892	5580	**5712**	5565	5656	5619
ME8	**5060**	5189	4960	**5060**	4960	5013	4984
ME9	**5009**	5009	4853	**5009**	4826	4959	4894
MF1	NA	NA	**9599494**	**9599494**	**9599494**	725*	2271*(2)
MF2	NA	NA	**1214080**	**1214080**	**1214080**	1204*	890*(4)
MF3	NA	NA	449840	**450135**	449730	917*	2255*(3)
MF4	NA	NA	886860	**887215**	886905	1834*	7103*(1)
MF5	NA	NA	452630	**452635**	452610	1061*	2462*(2)
MF6	NA	NA	884080	884470	**885240**	2239*	NA
MF7	NA	NA	**1484230**	1483920	1483535	2642*	NA
MG1	175	352	**249**	**249**	**249**	197	102
MG2	290	487	**373**	372	371	299	224
MG3	521	687	**590**	**590**	**590**	485	406
MG4	575	798	**711**	709	**711**	575	511
MG5	702	951	848	**849**	848	702	570
MG6	701	938	**842**	**842**	**842**	701	601
MG7	808	1069	**963**	**963**	**963**	808	692
MH1	NA	NA	493474	**502996**	470618	NA	NA
MH2	NA	NA	**22121**	21990	21776	623*	25112*(4)
MH3	NA	NA	330885	**332632**	316660	NA	NA
MH4	NA	NA	30661	**30750**	30702	9*	119116*
MH5	NA	NA	53350	**53556**	53702	NA	NA

Table 2
Comparison of solvers on MaxSAT instances, 600 seconds.

	Gurobi		MaxSAT Solvers			OptSAT Solvers	
			TT-Open WBO-Inc-20	SATLike-c(w)	Open-WBO-Inc-complete		
	L.B.	U.B.				TS-ACW	Evol-TS
MA1	**17507795**	17509523	17505973	17507601	17506899	4*	30*(1)
MA2	**21079364**	21086011	21078680	21079230	21078091	6*	21*(2)
MA3	**8940536**	8941396	8936045	8939403	8936600	3*	76*(3)
MA4	**11708315**	11709471	11701995	11707913	11702166	5*	26*(3)
MA5	**14175895**	14177309	14172638	14175429	14171934	5*	84*(4)
MB1	503600	505257	**503850**	**503850**	503434	503079	502688
MB2	902713	909328	903663	**904444**	903634	900851	900394
MB3	1511420	1526670	1512217	**1516299**	1513070	1508610	1506465
MB4	2411280	2443250	2415938	**2422587**	2415186	2407617	2406281
MB5	3061720	3064080	3063378	**3063613**	3063489	3062143	3061933
MC1	NA	528112	**525570**	**525570**	506995	1633*	1403*(2)
MC2	NA	379101	NA	NA	NA	2127*	11777*(4)
MC3	NA	195238	**193562**	193561	193241	912*	755*
MC4	11133	23491	**23247**	**23247**	**23247**	864*	5683*(4)
MC5	NA	484830	NA	NA	NA	2190*	11138*(2)
MC6	NA	96662	**86715**	81944	84770	349*	322*(2)
MC7	NA	9271	**7722**	**7722**	7700	1371*	7899*(2)
MD1	505308	505344	**505317**	**505317**	**505317**	505267	505312
MD2	505931	505962	**505937**	**505937**	**505937**	505855	505926
MD3	506547	506590	**506562**	**506562**	**506562**	506488	506553
MD4	507407	507430	507413	507413	**507414**	507405	507415
MD5	502221	502243	502228	**502286**	502227	502221	502182
ME1	**4653**	4653	4550	**4653**	4506	4624	4566
ME2	**5157**	5157	5049	**5157**	5068	5107	5061
ME3	**4573**	4573	4510	**4573**	4495	4549	4497
ME4	15732	16703	15599	**15785**	15542	15578	15472
ME5	16295	16998	15867	**16317**	15907	16054	16024
ME6	**5967**	6019	5914	**5967**	5784	5907	5832
ME7	**5712**	5712	5580	**5712**	5565	5658	5619

Contd...

ME8	**5060**	5060	4960	**5060**	4960	5020	4984
ME9	**5009**	5009	4853	**5009**	4826	4961	4894
MF1	NA	NA	**9599494**	**9599494**	**9599494**	708*	528*(2)
MF2	NA	NA	**1214080**	**1214080**	**1214080**	1321*	769*(4)
MF3	NA	NA	449840	**450135**	449730	873*	747*(3)
MF4	NA	NA	886860	**887215**	886905	1690*	7103*(1)
MF5	NA	NA	452630	**452635**	452610	956*	865*(2)
MF6	NA	NA	884080	884470	**885240**	623*	NA
MF7	NA	NA	**1484230**	1483920	1483535	1093*	NA
MG1	183	327	**249**	**249**	**249**	198	186
MG2	304	477	**373**	**373**	**373**	300	294
MG3	521	687	**590**	**590**	**590**	528	492
MG4	647	798	**711**	**711**	**711**	575	537
MG5	786	951	**849**	**849**	**849**	702	627
MG6	779	938	**842**	**842**	**842**	701	628
MG7	891	1069	**963**	**963**	**963**	808	722
MH1	NA	NA	493183	**502996**	501462	NA	NA
MH2	NA	NA	22151	22148	**22157**	502*	12716*(4)
MH3	NA	NA	330885	**332632**	330672	NA	NA
MH4	NA	NA	30684	30753	**30773**	6*	116329*
MH5	NA	NA	53765	**53870**	53750	NA	NA

As shown in Table 1 and Table 2, MaxSAT instances are best solved using the group of MaxSAT solvers. For all 50 instances, they provide better results than the OptSAT solvers. The best performance is by the SATLike-c(w) solver, which returns the best solution found for the majority of instances. Gurobi presents lower bounds equal to the solutions of the SATLike-c(w) solver for instances with a relatively small number of variables and clauses. OptSAT solvers find admissible solutions for moderate-size instances but do not find feasible solutions to large instances. In the past studies using the OptSAT solvers, the instances used have not involved more than up to 3,000 variables and 15,000 clauses. However, large MaxSAT instances have more than 100,000 variables and one million clauses. Adding to this, when converting an instance of MaxSAT to OptSAT, the number of variables increases by the number of soft clauses, and the number of literals in each soft clause increases by one.

Table 3
Comparison of solvers on OptSAT instances, 60 seconds.

	Gurobi		MaxSAT Solvers			OptSAT Solvers	
	L.B.	U.B.	TT-Open WBO-Inc-20	SATLike-c(w)	Open-WBO-Inc-complete	TS-ACW	Evol-TS
OA1	**20518**	21056	19975	20481	20066	**20518**	**20518**
OA2	20582	21302	20208	20546	20203	**20619**	20613
OA3	20373	21048	20057	20410	20050	**20441**	**20441**
OA4	20848	21482	20610	20823	20645	**20864**	**20864**
OA5	20550	21217	20113	20607	20113	**20614**	**20614**
OB1	**5729**	5729	**5729**	**5729**	**5729**	**5729**	**5729**
OB2	**5615**	5615	5601	**5615**	5610	**5615**	**5615**
OB3	**5150**	5150	**5150**	**5150**	**5150**	**5150**	**5150**
OB4	**5595**	5595	5565	**5595**	5577	**5595**	**5595**
OB5	**5613**	5613	**5613**	**5613**	**5613**	**5613**	**5613**
OC1	144159	147292	143786	144563	143699	**144708**	**144708**
OC2	145973	149571	145101	146188	145190	**146357**	**146357**
OC3	142356	145450	141357	142459	141403	**142624**	**142624**
OC4	144019	147211	142990	144012	142976	**144188**	144132
OC5	137015	139927	135718	137144	135832	**137173**	137146
OD1	134664	209332	119804	140432	119510	142021	**142092**
OD2	135036	212942	118000	137222	113205	141845	**141984**
OD3	135585	208422	124083	138524	122967	**142141**	142074
OD4	140393	214991	123391	139790	119894	143083	**143119**
OD5	134942	208613	122589	140188	122271	**142289**	**142289**
OE1	142001	146046	141155	142093	141178	**142220**	142217
OE2	142759	146752	142056	142880	141987	**142988**	**142988**
OE3	141735	145545	140559	141580	140756	**141791**	141778

Contd...

OE4	137759	141393	136987	137640	136779	**137892**	**137892**
OE5	139917	143874	139199	140031	138840	**140193**	140190
OF1	184176	237699	167276	185066	163319	**186153**	186116
OF2	183939	235063	164999	182982	164723	**186894**	186879
OF3	182263	236660	160739	183214	162605	185225	**185275**
OF4	187547	237434	169535	187743	167795	**189238**	189054
OF5	183948	235924	165503	185207	161446	186754	**186902**
OG1	1575200	2223120	1464559	1652967	1463023	1664253	**1672519**
OG2	1609300	2224770	1460533	1655348	1463692	1661876	**1673550**
OG3	1603460	2219150	1459776	1668414	1466699	1672658	**1683733**
OG4	1590310	2220050	1473187	1659679	1477845	1672074	**1681865**
OG5	1606100	2202530	1476200	1638021	1469312	1657047	**1665911**
OH1	6166660	8959580	5967820	**6716667**	5880825	6561532	6596940
OH2	6060050	8936920	5740598	6629182	5716279	6503989	**6634287**
OH3	6127890	8890140	5784650	**6612473**	5671659	6455277	6492690
OH4	6092730	8924040	5893007	**6694655**	5823968	6551069	6580875
OH5	6167160	8983810	5837822	**6618794**	5772874	6452793	6554137
OI1	**130**	130	**130**	**130**	**130**	1*	76*(2)
OI2	**156**	156	**156**	**156**	**156**	3*	93*(2)
OI3	**181**	191	**181**	**181**	**181**	4*	119*(4)
OI4	**207**	227	**207**	**207**	**207**	4*	104*(4)
OI5	216	245	**219**	**219**	**219**	8*	137*(4)
OJ1	**705**	705	**705**	**705**	**705**	1*	1099*
OJ2	**979**	979	**979**	**979**	**979**	2*	1894*(2)
OJ3	**4440**	4446	4437	4438	4438	15*	4930*(3)
OJ4	**2751**	2751	**2751**	**2751**	**2751**	33*	11014*(2)
OJ5	**5894**	5894	**5894**	**5894**	**5894**	57*	19746*(3)

Table 4
Comparison of solvers on OptSAT instances, 600 seconds.

	Gurobi		MaxSAT Solvers			OptSAT Solvers	
			TT-Open WBO-Inc-20	SATLike-c(w)	Open-WBO-Inc-complete		
	L.B.	U.B.				TS-ACW	Evol-TS
OA1	**20518**	20857	19975	20481	20066	**20518**	**20518**
OA2	20609	21130	20208	20546	20203	**20619**	20613
OA3	20425	20859	20057	20410	20050	**20441**	**20441**
OA4	20848	21297	20610	20823	20645	**20864**	**20864**
OA5	20598	21039	20113	20607	20113	**20614**	**20614**
OB1	**5729**	5729	**5729**	**5729**	**5729**	**5729**	**5729**
OB2	**5615**	5615	**5615**	**5615**	**5615**	**5615**	**5615**
OB3	**5150**	5150	**5150**	**5150**	**5150**	**5150**	**5150**
OB4	**5595**	5595	**5595**	**5595**	**5595**	**5595**	**5595**
OB5	**5613**	5613	**5613**	**5613**	**5613**	**5613**	**5613**
OC1	144676	146776	143786	144563	143699	**144708**	**144708**
OC2	146094	149177	145101	146188	145190	**146357**	**146357**
OC3	142356	145134	141357	142459	141403	**142624**	**142624**
OC4	144019	146813	142990	144012	142976	**144196**	144132
OC5	137015	139549	135718	137144	135832	**137180**	**137180**
OD1	136684	206538	119804	140432	119510	**142144**	142092
OD2	137835	210246	118000	137222	113205	141901	**141984**
OD3	135799	205519	124083	138524	122967	**142348**	142137
OD4	141757	212390	123391	139790	119894	**143173**	143163
OD5	136448	205964	122589	140188	122271	**142289**	**142289**
OE1	142001	145672	141155	142093	141178	**142220**	**142220**
OE2	142759	146390	142056	142880	141987	**142988**	**142988**
OE3	141735	145162	140559	141580	140756	**141798**	141784
OE4	137759	141039	136987	137640	136779	**137892**	**137892**
OE5	139917	143531	139199	140031	138840	**140206**	140190
OF1	184176	232476	167276	185066	163319	**186407**	186158
OF2	183939	230940	164999	182982	164723	**186965**	186886

Contd...

OF3	182528	232316	160739	183214	162605	**185510**	184857
OF4	187810	232952	169535	187743	167795	**189372**	189084
OF5	183955	231568	165503	185207	161446	**186966**	186824
OG1	1651982	2221946	1464559	1652967	1463023	1667627	**1675820**
OG2	1646419	2221625	1460533	1655348	1463692	1661876	**1676305**
OG3	1657972	2217402	1459776	1668414	1466699	1674125	**1685346**
OG4	1665218	2220054	1473187	1659679	1477845	1673147	**1685724**
OG5	1631455	2200403	1476200	1638021	1469312	1658534	**1669612**
OH1	6481754	8939279	5967820	**6716667**	5880825	6597452	6608456
OH2	6457859	8912344	5740598	6629182	5716279	6630962	**6678527**
OH3	6368438	8873074	5784650	**6612473**	5671659	6479011	6531128
OH4	6487984	8903377	5893007	**6694655**	5823968	6563980	6608834
OH5	6494549	8977461	5837822	6618794	5772874	6626184	**6673492**
OI1	**130**	130	**130**	**130**	**130**	1*	68*(2)
OI2	**156**	156	**156**	**156**	**156**	2*	90*(2)
OI3	**181**	191	**181**	**181**	**181**	4*	117*(4)
OI4	**207**	227	**207**	**207**	**207**	4*	104*(4)
OI5	216	245	**219**	**219**	**219**	6*	134*(4)
OJ1	**705**	705	**705**	**705**	**705**	1*	1066*
OJ2	**979**	979	**979**	**979**	**979**	1*	1804*(2)
OJ3	**4440**	4446	4436	4438	4438	15*	4730*(3)
OJ4	**2751**	2751	**2751**	**2751**	**2751**	25*	10988*(2)
OJ5	**5894**	5894	**5894**	**5894**	**5894**	55*	19746*(3)

Table 3 and Table 4 show that the results for OptSAT instances depend on the nature of the instances. The OptSAT instances which were provided by Davoine et al. [6] and da Silva et al. [5] are best tackled by the group of OptSAT solvers. For the majority of such instances, MaxSAT solvers are not able to outperform the OptSAT solvers. This is perhaps surprising, given that each of the three MaxSAT solvers tested internally converts the representation of the problem solved to a form that resembles OptSAT, and therefore should be effective at finding solutions for such instances. However, a main characteristic of these instances is that it is not difficult to find feasible solutions. In addition, the instances are not particularly large. They are nevertheless not easy, as evidenced by the performance of the

Gurobi solver and the MaxSAT solvers: these solvers may be excellent at finding a feasible solution, but struggles to subsequently find better feasible solutions. On the other hand, the OptSAT solvers were created to have an ability of efficiently navigating between feasible and infeasible solutions, and thus provide superior performance in terms of identifying near-optimal solutions in such solution landscapes.

On the set of MinCostSAT instances (labelled OI and OJ), however, the group of MaxSAT solvers outperforms the OptSAT solvers. These instances have a challenging feasibility part, and MaxSAT solvers are better at coping with this. The mixed-integer programming solver Gurobi is able to find solutions of good quality for MinCostSAT instances (in 6 out of 10 cases, an optimal solution is found within 60 seconds), while the OptSAT solvers end up with infeasible solutions for the given time limits and hardware.

Table 5

Total number of best results for each solver for time limits of 60/600 seconds.

	Gurobi	MaxSAT Solvers			OptSAT Solvers	
		TT-Open WBO-Inc-20	SATLike-c(w)	Open-WBO -Inc-complete	TS-ACW	Evol-TS
MaxSAT Instances	8/12	19/19	37/36	14/17	0/0	0/0
OptSAT Instances	15/15	12/14	18/17	12/14	25/29	26/25
Total	20/22	31/33	57/58	26/31	25/29	26/25

Table 5 summarizes the results by showing, for each solver, the number of instances for which the solver provided the best average solution. Each cell has two values, representing a time limit of 60 and 600 seconds, respectively. The computational experiments show that OptSAT solvers cannot compete with MaxSAT solvers on the MaxSAT instances, while in its turn, MaxSAT solvers give worse results than the OptSAT solvers for most of the OptSAT instances. The exception to this is the set of MinCostSAT instances, which the MaxSAT solvers tackle well. It is also observed that these observations hold for both of the time limits tested: none of the solution methods are able to improve their performances significantly when going from 60 seconds to 600 seconds of total runtime.

6. Conclusions

MaxSAT and OptSAT are similar in structure, yet appear to be sufficiently different to warrant separate treatment. In other words, the current situation is that specialized solvers for one of the problems cannot compete with specialized solvers for the other problem when instances of this later problem are tackled. This may in particular be related to the type of instances that are commonly used within each of the modelling paradigms: OptSAT instances are often characterized by propositional formulas that are relatively easy to satisfy, but where finding the correct mix of truth values to maximize the objective function is hard. MaxSAT instances typically involve formulas that are hard to satisfy, and any feasible solution is therefore also close in quality to any optimal solution. These instances also tend to be very large, and an effort must be made to create solvers that deal efficiently with the corresponding large sets of data.

For state-of-the-art OptSAT-solvers, more focus seems to be needed on dealing with feasibility. In particular, for instances derived from the MinCostSAT problem, where feasibility is harder to obtain, the MaxSAT solvers are performing better than the OptSAT solvers. On the other hand, when feasibility is relatively easy to obtain, the MaxSAT solvers seem to have a potential for improvement by focusing more on the optimization aspect. That is, on the original OptSAT instances, the tested MaxSAT solvers find feasible solutions, but the solutions found are typically far from the optimal objective function values. This indicates that both state-of-the-art MaxSAT solvers and OptSAT solvers have potential for improvement, allowing them to deal successfully with a larger variety of instances.

A. Description of instances

In the following, n is the number of variables, m is the number of hard clauses, and q is the number of soft clauses. The number of literals in each clause is indicated by $|A_i \cup B_i|$, either for hard clauses when $i \in \{1,\ldots,m\}$ or for soft clauses when $i \in \{m+1,\ldots,m+q\}$. Table 6 lists the MaxSAT instances, and Table 7 lists the OptSAT instances.

Table 6
Description of the MaxSAT instances

Family	ID	Full name	n	m	q	$\lvert A_i \cup B_i \rvert$ (hard)	$\lvert A_i \cup B_i \rvert$ (soft)
mpe	MA1	random-net-30-6 _network-4.net	86	29	8403	2-4	1-6
	MA2	random-net-40-5 _network-7.net	116	38	10276	2-4	1-6
	MA3	random-net-40-5 _network-10.net	105	37	5204	2-4	1-6
	MA4	random-net-50-5 _network-9.net	144	48	5884	2-4	1-6
	MA5	random-net-50-5 _network-10.net	147	49	7198	2-4	1-6
ramsey	MB1	ram_k3_n13.ra1	78	0	1001	0	3-6
	MB2	ram_k3_n15.ra1	105	0	1820	0	3-6
	MB3	ram_k3_n17.ra1	136	0	3060	0	3-6
	MB4	ram_k3_n19.ra1	171	0	4845	0	3-6
	MB5	ram_k4_n18.ra1	153	0	6120	0	6
pseudo Boolean	MC1	normalized-mps-v2-20-10-cracpb1.opb	17336	59694	572	1-14	1
	MC2	normalized-mps-v2-20-10-l152lav.opb	24570	97199	1989	1-16	1
	MC3	normalized-mps-v2-20-10-lp4l.opb	10516	35820	1086	1-10	1
	MC4	normalized-mps-v2-20-10-mod008.opb	10309	62038	319	1-93	1
	MC5	normalized-mps-v2-20-10-mod010.opb	24776	79461	2655	1-4	1
	MC6	normalized-mps-v2-20-10-p0548.opb	5519	19214	416	1-14	1
	MC7	normalized-mps-v2-20-10-sentoy.opb	13262	89480	60	1-16	1
set covering	MD1	scpnrg1_weighted	10000	1000	10000	153-254	1
	MD2	scpnrg3_weighted	10000	1000	10000	154-258	1

Contd...

	MD3	scpnrg5_weighted	10000	1000	10000	155-256	1
	MD4	scpnrh2_weighted	10000	1000	10000	437-580	1
	MD5	scpnrh4_weighted	10000	1000	10000	437-578	1
max cut	ME1	brock200_4.clq	40	0	1028	0	2
	ME2	brock400_3.clq	40	0	1136	0	2
	ME3	brock800_3.clq	40	0	1014	0	2
	ME4	hamming6-2.clq	64	0	3648	0	2
	ME5	johnson8-4-4.clq	70	0	3710	0	2
	ME6	p_hat500-3.clq	42	0	1310	0	2
	ME7	p_hat700-3.clq	42	0	1260	0	2
	ME8	san200_0.7_1.clq	40	0	1116	0	2
	ME9	sanr200_0.7.clq	40	0	1092	0	2
min width	MF1	milan_200_12_1k_10s_1t_12	11558	16913	1092	2-3	1
	MF2	mitdbsample_100_43_1k_5s_2t_7	18945	26945	1548	2-3	1
	MF3	mitdbsample_200_26_1k_2s_2t_4	12794	16265	390	2-3	1
	MF4	mitdbsample_300_26_1k_6s_2t_6	23914	30968	728	2-3	1
	MF5	mitdbsample_300_43_1k_3s_2t_3	14542	17770	430	2-3	1
	MF6	mitdbsample_300_26_1k_6s_1t_8	29608	39948	1170	2-3	1
	MF7	mitdbsample_300_32_1k_6s_1t_8	34736	46955	1440	2-3	1
af synthesis	MG1	af-synthesis_stb_50_20_8	11550	75792	70	2-1157	1
	MG2	af-synthesis_stb_50_40_9	13091	113424	90	2-1741	1
	MG3	af-synthesis_stb_50_80_3	16151	188142	130	2-1361	1
	MG4	af-synthesis_stb_50_120_3	19266	261486	170	2-1361	1
	MG5	af-synthesis_stb_50_140_0	20765	301980	190	2-1093	1
	MG6	af-synthesis_stb_50_140_7	20518	315806	190	2-1661	1
	MG7	af-synthesis_stb_50_160_4	22316	338824	210	2-1507	1

Contd...

maxSAT Queries	MH1	adult_train_0_ CNF_5_20	141264	1392408	30024	2-13	1
	MH2	compas_train_ 0_CNF_5_5	24404	142929	6584	2-8	1
	MH3	credit_train_8 _CNF_5_15	132691	1943413	27551	2-19	1
	MH4	toms_test_1_ CNF_3_10	11282	555003	5549	2-97	1
	MH5	twitter_test_4 _CNF_3_10	807569	817102	9533	2-78	1

Table 7
Description of the OptSAT instances.

Family	ID	Full Name	n	m	$\lvert A_i \cup B_i \rvert$
class27	OA1	rn200m1000t10s0c25num0	200	1000	10
	OA2	rn200m1000t10s0c25num1	200	1000	10
	OA3	rn200m1000t10s0c25num2	200	1000	10
	OA4	rn200m1000t10s0c25num3	200	1000	10
	OA5	rn200m1000t10s0c25num4	200	1000	10
class33	OB1	rn100m400t20s10c50num0	100	400	10-30
	OB2	rn100m400t20s10c50num1	100	400	10-30
	OB3	rn100m400t20s10c50num2	100	400	10-30
	OB4	rn100m400t20s10c50num3	100	400	10-30
	OB5	rn100m400t20s10c50num4	100	400	10-30
class49	OC1	rn500m2500t25s0c75num0	500	2500	25
	OC2	rn500m2500t25s0c75num1	500	2500	25
	OC3	rn500m2500t25s0c75num2	500	2500	25
	OC4	rn500m2500t25s0c75num3	500	2500	25
	OC5	rn500m2500t25s0c75num4	500	2500	25
class54	OD1	qn1000m10000t2s0c0num0	1000	10000	2
	OD2	qn1000m10000t2s0c0num1	1000	10000	2
	OD3	qn1000m10000t2s0c0num2	1000	10000	2
	OD4	qn1000m10000t2s0c0num3	1000	10000	2
	OD5	qn1000m10000t2s0c0num4	1000	10000	2
class63	OE1	rn500m2500t25s0c0num0	500	2500	25
	OE2	rn500m2500t25s0c0num1	500	2500	25

Contd...

	OE3	rn500m2500t25s0c0num2	500	2500	25
	OE4	rn500m2500t25s0c0num3	500	2500	25
	OE5	rn500m2500t25s0c0num4	500	2500	25
class64	OF1	lmhn500m2500num1	500	2500	3
	OF2	lmhn500m2500num2	500	2500	3
	OF3	lmhn500m2500num3	500	2500	3
	OF4	lmhn500m2500num4	500	2500	3
	OF5	lmhn500m2500num5	500	2500	3
class66	OG1	llmhn1500m7500num1	1500	7500	3
	OG2	llmhn1500m7500num2	1500	7500	3
	OG3	llmhn1500m7500num3	1500	7500	3
	OG4	llmhn1500m7500num4	1500	7500	3
	OG5	llmhn1500m7500num5	1500	7500	3
class69	OH1	lmhn3000m15000num1	3000	15000	3
	OH2	lmhn3000m15000num2	3000	15000	3
	OH3	lmhn3000m15000num3	3000	15000	3
	OH4	lmhn3000m15000num4	3000	15000	3
	OH5	lmhn3000m15000num5	3000	15000	3
Coloring	OI1	3col120_5.cnf	240	1026	2-4
	OI2	3col140_5.cnf	280	1196	2-4
	OI3	3col160_5.cnf	320	1366	2-4
	OI4	3col180_5.cnf	360	1536	2-4
	OI5	3col200_5.cnf	400	1706	2-4
Planning	OJ1	logistics.b	843	7301	2-14
	OJ2	logistics.c	1141	10719	2-14
	OJ3	logistics.d	4713	21991	2-16
	OJ4	bw_large.c.cnf	3016	50457	2-16
	OJ5	bw_large.d.cnf	6325	131973	2-20

References

[1] J. Argelich, C. M. Li, F. Manyà, and J. Planes, "The first and second Max-SAT evaluations," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 4, pp. 251–278 (2008).

[2] D. L. Berre and A. Parrain, "The SAT4J library, release 2.2," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 7, pp. 59–64 (2010).

[3] J. Coelho and M. Vanhoucke, "Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers," *European Journal of Operational Research,* vol. 213, pp. 73–82, Aug. (2011), doi: 10.1016/j.ejor.2011.03.019.

[4] S. A. Cook, "The complexity of theorem-proving procedures," in Proc. Third ACM Symp. Theory Comput., pp. 151–158 (1971).

[5] R. da Silva, L. M. Hvattum, and F. Glover, "Combining solutions of the optimum satisfiability problem using evolutionary tunneling," MENDEL, vol. 26, pp. 23–29, May (2020), doi: 10.13164/mendel.2020.1.023.

[6] T. Davoine, P. L. Hammer, and B. Vizvári, "A heuristic for Boolean optimization problems," *Journal of Heuristics,* vol. 9, pp. 229–247 (2003).

[7] L. Fang and M. S. Hsiao, "A fast approximation algorithm for MINONE SAT and its application on MAX-SAT solving," in Adv. Techniques Logic Synthesis, Optimizations and Applications, pp. 149–170, Springer (2011.

[8] Z. Fu and S. Malik, "Solving the minimum-cost satisfiability problem using SAT-based branch-and-bound search," in Proc. 2006 IEEE/ACM Int. Conf. Comput. Aided Design, pp. 852–859 (2006).

[9] Z. Fu and S. Malik, "On solving the partial MAX-SAT problem," in Theory and Applications of Satisfiability Testing - SAT 2006, A. Biere and C. P. Gomes, Eds., Berlin, Heidelberg: Springer, pp. 252–265 (2006).

[10] F. Glover and M. Laguna, Tabu Search, Kluwer Academic Publisher, Boston, Dordrecht, London (1997).

[11] P. Großmann, S. Hölldobler, M. Norbert, K. Nachtigall, J. Opitz, and P. Steinke, "Solving periodic event scheduling problems with SAT," in Adv. Research Appl. Artif. Intell., J. He, D. Wei, A. Moonis, and W. Xindong, Eds., Berlin, Heidelberg: Springer, pp. 166–175 (2012).

[12] S. Guo, W. Sun, and M. A. Weiss, "Solving satisfiability and implication problems in database systems," *ACM Trans. Database Systems,* vol. 21, no. 2, pp. 270–293 (1996), doi: 10.1145/232616.232692.

[13] G. D. Hachtel and F. Somenzi, Logic Synthesis and Verification Algorithms, 1st ed. Kluwer Academic Publishers, USA (2000), ISBN 0792397460.

[14] J. Homer and X. Ou, "SAT-solving approaches to context-aware enterprise network security management," *IEEE Journal on Selected Ar-*

eas in Communications, vol. 27, pp. 315–322, May (2009), doi: 10.1109/JSAC.2009.090407.

[15] H. H. Hoos and T. Stützle, "SATLIB: An online resource for research on SAT," in SAT2000, I. P. Gent, H. van Maaren, and T. Walsh, Eds., IOS Press, pp. 283–292 (2000).

[16] L. M. Hvattum, A. Løkketangen, and F. Glover, "Adaptive memory search for Boolean optimization problems," *Discrete Applied Mathematics,* vol. 142, pp. 99–109 (2004).

[17] L. M. Hvattum, A. Løkketangen, and F. Glover, "New heuristics and adaptive memory procedures for Boolean optimization problems," in Integer Programming: Theory and Practice, J. Karlof, Ed., CRC Press, Boca Raton, FL, pp. 1–18 (2006).

[18] F. Imeson and S. L. Smith, "A language for robot path planning in discrete environments: the TSP with Boolean satisfiability constraints," in Proc. IEEE Int. Conf. Robotics Automation (ICRA), pp. 5772–5777 (2014), doi: 10.1109/ICRA.2014.6907707.

[19] S. Joshi, P. Kumar, V. Manquinho, R. Martins, A. Nadel, and S. Rao, "Open-WBO-Inc in MaxSAT evaluation 2018," in MaxSAT Evaluation 2018: Solver and Benchmark Descriptions, F. Bacchus, J. Berg, M. Järvisalo, and R. Martins, Eds., vol. B-2018-2, Department of Computer Science, University of Helsinki, Finland, pp. 16–17 (2018).

[20] S. Joshi, P. Kumar, S. Rao, and R. Martins, "Open-WBO-Inc: Approximation strategies for incomplete weighted MaxSAT," *Journal on Satisfiability, Boolean Modeling and Computation,* vol. 11, pp. 73–97, Sept. (2019), doi: 10.3233/SAT190118.

[21] S. Joshi, P. Kumar, S. Rao, and R. Martins, "Open-WBO-Inc in MaxSAT evaluation 2020," in MaxSAT Evaluation 2020: Solver and Benchmark Descriptions, F. Bacchus, J. Berg, M. Järvisalo, and R. Martins, Eds., vol. B-2020-2, Department of Computer Science, University of Helsinki, Finland, pp. 24–25 (2020).

[22] Z. Lei and S. Cai, "Solving (weighted) partial MaxSAT by dynamic local search for SAT," in Proc. Twenty-Seventh Int. Joint Conf. Artificial Intelligence (IJCAI-18), pp. 1346–1352 (2018), doi: 10.24963/ijcai.2018/187.

[23] Z. Lei and S. Cai, "SATLike-c(w): solver description," in MaxSAT Evaluation 2020: Solver and Benchmark Descriptions, F. Bacchus, J. Berg, M. Järvisalo, and R. Martins, Eds., vol. B-2020-2, Department of Computer Science, University of Helsinki, Finland, p. 15 (2020).

[24] C. M. Li, Z. Zhu, F. Manyà, and L. Simon, "Minimum satisfiability and its applications," in Proc. IJCAI'11, Barcelona, Catalonia, Spain, pp. 605–610 (2011), AAAI Press, ISBN 9781577355137.

[25] X. Y. Li, Optimization Algorithms for the Minimum-Cost Satisfiability Problem, Ph.D. dissertation, North Carolina State University, 2004, AAI3154233.

[26] V. Manquinho, J. Marques-Silva, and J. Planes, "Algorithms for weighted Boolean optimization," (2009), ISBN 978-3-642-02776-5, doi: 10.1007/978-3-642-02777-2_45.

[27] J. Marques-Silva, J. Argelich, A. Graça, and I. Lynce, "Boolean lexicographic optimization: algorithms & applications," *Annals of Mathematics and Artificial Intelligence*, vol. 62, pp. 317–343 (2011).

[28] J. P. Marques-Silva and K. A. Sakallah, "Boolean satisfiability in electronic design automation," in Proc. 37th Annual Design Automation Conf. (DAC '00), New York, NY, USA, pp. 675–680 (2000), Association for Computing Machinery, ISBN 1581131879, doi: 10.1145/337292.337611.

[29] A. Nadel, "TT-Open-WBO-Inc-20: an anytime MaxSAT solver entering MSE'20," in MaxSAT Evaluation 2020: Solver and Benchmark Descriptions, F. Bacchus, J. Berg, M. Järvisalo, and R. Martins, Eds., vol. B-2020-2, Department of Computer Science, University of Helsinki, Finland, pp. 32–33 (2020).

[30] A. Nadel, "Polarity and variable selection heuristics for SAT-based anytime MaxSAT," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 12, pp. 17–22 (2020).

[31] A. Nadel, "On optimizing a generic function in SAT," in Proc. 2020 Formal Methods in Computer Aided Design (FMCAD), pp. 205–213 (2020), doi: 10.34727/2020/isbn.978-3-85448-042-6_28.

[32] E. Niklas and S. Niklas, "An extensible SAT-solver," in Theory and Applications of Satisfiability Testing, E. Giunchiglia and A. Tacchella, Eds., Springer Berlin Heidelberg, pp. 502–518 (2004), ISBN 978-3-540-24605-3.

[33] H. Pierre and J. Brigitte, "Algorithms for the maximum satisfiability problem," *Computing*, vol. 44, no. 4, pp. 279–303, Apr. (1990), ISSN 0010-485X, doi: 10.1007/BF02241270.

[34] O. Roussel and V. Manquinho, Pseudo-Boolean and Cardinality Constraints, IOS Press, Netherlands (2020), ISBN 1586039296.

[35] M. Ruben, M. Vasco, and L. Inês, "Open-WBO: a modular MaxSAT solver," in Theory and Applications of Satisfiability Testing - SAT 2014, S. Carsten and E. Uwe, Eds., Springer International Publishing, pp. 438–445 (2014), ISBN 978-3-319-09284-3.

[36] H. Xu, R. A. Rutenbar, and K. Sakallah, "Sub-SAT: A formulation for relaxed Boolean satisfiability with applications in routing," in Proc. 2002 Int. Symposium on Physical Design (ISPD '02), New York, NY, USA, pp. 182–187 (2002), Association for Computing Machinery, ISBN 1581134606.

[37] M. Zapletina, D. Zhukov, and S. V. Gavrilov, "Boolean satisfiability methods for modern computer-aided design problems in microelectronics," Proceedings of Universities. ELECTRONICS, vol. 25, pp. 525–538, Dec. (2020), doi: 10.24151/1561-5405-2020-25-6-525-538.

Received November, 2022