Encoding Max-CSP into Partial Max-SAT[*]

Josep Argelich
DIEI, UdL
Lleida, Spain

Alba Cabiscol
DIEI, UdL
Lleida, Spain

Inês Lynce
IST, INESC-ID
Lisboa, Portugal

Felip Manyà
DIEI, UdL
Lleida, Spain

Abstract

We define a number of original encodings that map Max-CSP instances into Partial Max-SAT instances. Our encodings rely on the well-known direct and support encodings from CSP into SAT. Then, we report on an experimental investigation that was conducted to compare the performance profile of our encodings on random binary Max-CSP instances. Moreover, we define a new variant of the support encoding from CSP into SAT which produces fewer clauses than the standard support encoding.

1 Introduction

In the last few years there has been a growing interest in studying logical and algorithmic aspects of Max-SAT formalisms. The most recent and relevant results can be summarized as follows: (i) there exist Max-SAT solvers like MaxSatz [12, 13] and MiniMaxSat [9] which solve instances that are beyond the reach of the solvers existing just five years ago; (ii) sound resolution refinements have been incorporated into Max-SAT solvers [11, 14], as well as good quality underestimations of the lower bound [12, 13], (iii) a resolution-style calculus for Max-SAT has been proven to be complete [4, 5], (iv) formalisms like Partial Max-SAT have been investigated for solving problems with soft constraints [1, 9, 2], and (v) two evaluations of Max-SAT solvers have been performed for the first time as a co-located event of the 2006 and 2007 editions of the International Conference on Theory and Applications of Satisfiability Testing.

In this paper, we define a number of original encodings that map Max-CSP instances into Partial Max-SAT instances. Our encodings rely on the well-known direct and support encodings from CSP into SAT. Then, we report on an experimental investigation that was conducted to com-pare the performance profile of our encodings on random binary Max-CSP instances. Interestingly, we also define a new variant of the support encoding from CSP into SAT, called *minimal support encoding*, which produces fewer clauses than the standard support encoding.

The objective of our research is to show that different Max-SAT encodings for a same optimization problem may produce substantial differences on performance, as well as to identify features of the encodings that lead to better performance profiles.

The structure of the paper is as follows. Section 2 contains preliminary definitions about Max-SAT and CSP. Section 3 surveys the support and direct encodings from CSP into SAT, and defines the minimal support encoding. Section 4 defines a number of original encodings from Max-CSP into Partial Max-SAT. Section 5 reports and analyses the experimental investigation. Section 6 presents the conclusions and future research directions.

2 Preliminaries

2.1 Max-SAT definitions

In propositional logic a variable x_i may take values 0 (for false) or 1 (for true). A literal l_i is a variable x_i or its negation $\bar{x}_i$. A clause is a disjunction of literals, and a CNF formula is a multiset of clauses.

An assignment of truth values to the propositional variables satisfies a literal x_i if x_i takes the value 1 and satisfies a literal $\bar{x}_i$ if x_i takes the value 0, satisfies a clause if it satisfies at least one literal of the clause, and satisfies a CNF formula if it satisfies all the clauses of the formula.

The Max-SAT problem for a CNF formula ϕ is the problem of finding an assignment of values to propositional variables that maximizes the number of satisfied clauses. In this sequel we often use the term Max-SAT meaning Min-UNSAT. This is because, with respect to exact computations, finding an assignment that minimizes the number of unsatisfied clauses is equivalent to finding an assignment that maximizes the number of satisfied clauses.

We also consider the extension of Max-SAT known as Partial Max-SAT because it is more well-suited for representing and solving NP-hard problems. A Partial Max-SAT instance is a CNF formula in which some clauses are *relaxable* or *soft* and the rest are *non-relaxable* or *hard*. Solving a Partial Max-SAT instance amounts to finding an assignment that satisfies all the hard clauses and the maximum number of soft clauses. Hard clauses are represented between square brackets, and soft clauses are represented between round brackets.

2.2 Max-CSP definitions

Definition 1. A *Constraint Satisfaction Problem (CSP)* instance is defined as a triple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{X} = \{X_1, \ldots, X_n\}$ is a set of variables, $\mathcal{D} = \{d(X_1), \ldots, d(X_n)\}$ is a set of finite domains containing the values the variables may take, and $\mathcal{C} = \{C_1, \ldots, C_m\}$ is a set of constraints. Each constraint $C_i = \langle S_i, R_i \rangle$ is defined as a relation R_i over a subset of variables $S_i = \{X_{i_1}, \ldots, X_{i_k}\}$, called the *constraint scope*. The relation R_i may be represented extensionally as a subset of the Cartesian product $d(X_{i_1}) \times \cdots \times d(X_{i_k})$.

Definition 2. An *assignment* v for a CSP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is a mapping that assigns to every variable $X_i \in \mathcal{X}$ an element $v(X_i) \in d(X_i)$. An assignment v satisfies a constraint $\langle \{X_{i_1}, \ldots, X_{i_k}\}, R_i \rangle \in C$ iff $\langle v(X_{i_1}), \ldots, v(X_{i_k}) \rangle \in R_i$.

Definition 3. The *Constraint Satisfaction Problem (CSP)* for a CSP instance P consists in deciding whether there exists an assignment that satisfies P.

The *Max-CSP problem* for a CSP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is the problem of finding an assignment that minimizes (maximizes) the number of violated (satisfied) constraints.

In the sequel we assume that all CSP are unary and binary; i.e., the scope of all the constraints has at most cardinality two.

3 Encoding CSP into SAT

Mappings of binary CSPs into SAT is an area of research that has been investigated by several authors [3, 7, 10, 15, 6, 8]. They have proposed a number of encodings with different performance profiles and achieving different degrees of local propagation on SAT solvers. Among them, the most well-known are the direct encoding and the support encoding. They have proposed a number of encodings having different performance profiles and achieving different degrees of local propagation on SAT solvers. Among them, the most well-known are the direct encoding and the support encoding. In the rest of this section, we first define the direct encoding and the support encoding, and then define a new encoding from CSP into SAT called *minimal support encoding*.

3.1 Direct encoding and support encoding

In the *direct encoding*, we associate a Boolean variable x_{ij} with each value j that can be assigned to the CSP variable X_i. Assuming that X_i has a domain of size m, the direct encoding contains clauses that ensure that each CSP variable X_i is given a value: for each i, $x_{i1} \vee \cdots \vee x_{im}$ (called *at-least-one* clauses), and contains clauses that rule out any binary nogoods. For example, if $X_1 = 2$ and $X_3 = 1$ is not allowed, then the clause $\overline{x}_{12} \vee \overline{x}_{31}$ (called *conflict* clause) is added. We consider the version of the direct encoding that adds clauses that ensure that each CSP variable X_i takes no more than one value: for each i, j, k with $j < k$, $\overline{x}_{ij} \vee \overline{x}_{ik}$ (called *at-most-one* clauses). These clauses are redundant, but are considered in the literature in order to maintain a one-to-one mapping between CSP models and SAT models.

In the *support encoding*, the idea is to encode into clauses the *support* for a value instead of encoding conflicts. The support for a value j of a CSP variable X_i across a constraint is the set of values of the other variable in the constraint which allow $X_i = j$. If $v_1, v_2, \ldots, v_k$ are the supporting values of variable X_l for $X_i = j$, we add the clause $\overline{x}_{ij} \vee x_{lv_1} \vee x_{lv_2} \vee \cdots \vee x_{lv_k}$ (called *support* clause). There is one support clause for each pair of variables X_i, X_l involved in a constraint, and for each value in the domain of X_i. Unlike conflict clauses, a clause in each direction is used in the literature, one for the pair X_i, X_l and one for X_l, X_i. The support clauses on their own do not provide a correct encoding of CSPs into SAT. To complete an encoding using support clauses we need to add the at-least-one and at-most-one clauses for each CSP variable to ensure that each CSP variable takes exactly one value of its domain.

Example 1. The direct encoding of the CSP $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle = \langle \{X, Y\}, \{d(X) = \{1, 2, 3\}, d(Y) = \{1, 2, 3\}\}, \{X \leq Y\} \rangle$ contains the following clauses:

at-least-one	$x_1 \vee x_2 \vee x_3$	$y_1 \vee y_2 \vee y_3$	
at-most-one	$\overline{x}_1 \vee \overline{x}_2$	$\overline{x}_1 \vee \overline{x}_3$	$\overline{x}_2 \vee \overline{x}_3$
	$\overline{y}_1 \vee \overline{y}_2$	$\overline{y}_1 \vee \overline{y}_3$	$\overline{y}_2 \vee \overline{y}_3$
conflict	$\overline{x}_2 \vee \overline{y}_1$	$\overline{x}_3 \vee \overline{y}_1$	$\overline{x}_3 \vee \overline{y}_2$

and the support encoding for that CSP contains the at-least-one clauses, the at-most-one clauses, and the following support clauses:

support	$\overline{x}_2 \vee y_2 \vee y_3$	$\overline{y}_1 \vee x_1$
	$\overline{x}_3 \vee y_3$	$\overline{y}_2 \vee x_1 \vee x_2$

The support clause for x_1 is missing because it is subsumed by $y_1 \vee y_2 \vee y_3$, and the support clause for y_3 is missing because it is subsumed by $x_1 \vee x_2 \vee x_3$.

3.2   Minimal support encoding

Our first contribution in this paper is to give a new version of the support encoding, which we call *minimal support encoding*. Our definition follows from the observation that the support encoding contains redundant clauses. More precisely, given a binary constraint C_k with scope $\{X, Y\}$, it is enough to add the support clauses either for the values of X or for the values of Y; it is not necessary to add a clause in each direction. Despite the number of papers dealing with the support encodings, this fact has gone unnoticed so far.

Definition 4. The *minimal support encoding* is like the support encoding except for the fact that, for every constraint C_k with scope $\{X, Y\}$, we only add either the support clauses for all the domain values of the CSP variable X or the support clauses for all the domain values of the CSP variable Y.

Example 2. A minimal support encoding for the CSP instance from Example 1 contains the following clauses:

at-least-one	$x_1 \vee x_2 \vee x_3$	$y_1 \vee y_2 \vee y_3$	
at-most-one	$\overline{x}_1 \vee \overline{x}_2$	$\overline{x}_1 \vee \overline{x}_3$	$\overline{x}_2 \vee \overline{x}_3$
	$\overline{y}_1 \vee \overline{y}_2$	$\overline{y}_1 \vee \overline{y}_3$	$\overline{y}_2 \vee \overline{y}_3$
support	$\overline{x}_2 \vee y_2 \vee y_3$	$\overline{x}_3 \vee y_3$	

Proposition 5. *The* minimal support encoding *is correct.*

PROOF: We assume, without loss of generality, that we add the support clauses for all the domain values of the CSP variable X for every constraint C_k with scope $\{X, Y\}$. Given a CSP assignment, we construct its corresponding Boolean assignment by setting the variable x_i to true if the CSP assignment assigns the value i to X; otherwise, we set the variable x_i to false. Given a Boolean assignment that satisfies the minimal support encoding of a CSP, we construct its corresponding CSP assignment by assigning to the CSP variable X the value i if x_i is true. Note that there is exactly one x_i for each CSP variable X which is true because the minimal support encoding contains the at-least-one and at-most-one clauses. So, it is a valid CSP assignment.

We prove first that if a CSP assignment satisfies all the constraints of a CSP instance, then its corresponding Boolean assignment satisfies its minimal encoding. Since a CSP assignment assigns exactly one value to each CSP variable, the Boolean assignment satisfies the at-least-one and at-most-one clauses. For every constraint C_k with scope $\{X, Y\}$, the CSP assignment assigns a value i to X and a

value j to Y. Since $(X = i, Y = j)$ is an allowed combination, among the clauses encoding that constraint, there is a clause of the form $\overline{x}_i \vee y_j \vee \cdots$ which is satisfied by the Boolean encoding because y_j is true. The remaining clauses are also satisfied by the Boolean assignment because they are of the form $\overline{x}_l \vee \cdots$, where $l \neq i$, and the Boolean assignment assigns the value false to all variables x_l with $l \neq i$.

We prove now that if a Boolean assignment satisfies the minimal support encoding of a CSP instance P, then its corresponding CSP assignment satisfies P. Assume that the CSP assignment does not satisfy P. Therefore, there exists a constraint C_k of P with scope $\{X, Y\}$ which is violated because the CSP assignment assigns a value i to X and a value j to Y which corresponds to a forbidden combination. In this case, there is exactly one support clause of the form $\overline{x}_i \vee y_{j_1} \vee \cdots \vee y_{j_k}$ among the support clauses encoding C_k which is not satisfied by the Boolean assignment because x_i is true and $y_{j_1} \neq y_j, \ldots, y_{j_k} \neq y_j$. The rest of support clauses encoding C_k are satisfied by the Boolean assignment because it assigns the value false to all variables x_l with $l \neq i$. ∎

4   Encoding Max-CSP into Partial Max-SAT

4.1   Direct encoding for Partial Max-SAT

Given a CSP instance P, our goal is to define a version of the direct encoding that produces a Partial Max-SAT instance ϕ such that the minimum number of constraints of P that are violated by a CSP assignment is exactly the same as the minimum number of clauses of ϕ that are falsified by a Boolean assignment.

Definition 6. The *direct encoding* of a Max-CSP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is the Partial Max-SAT instance that contains as hard clauses the corresponding at-least-one and at-most-one clauses for every CSP variable in $\mathcal{X}$, and contains a soft clause $\overline{x}_i \vee \overline{y}_j$ for every nogood $(X = i, Y = j)$ of every constraint of $\mathcal{C}$ with scope $\{X, Y\}$.

Example 3. The Partial Max-SAT direct encoding for the Max-CSP problem of the CSP instance from Example 1 is as follows:

at-least-one	$[x_1 \vee x_2 \vee x_3]$	$[y_1 \vee y_2 \vee y_3]$	
at-most-one	$[\overline{x}_1 \vee \overline{x}_2]$	$[\overline{x}_1 \vee \overline{x}_3]$	$[\overline{x}_2 \vee \overline{x}_3]$
	$[\overline{y}_1 \vee \overline{y}_2]$	$[\overline{y}_1 \vee \overline{y}_3]$	$[\overline{y}_2 \vee \overline{y}_3]$
conflict	$(\overline{x}_2 \vee \overline{y}_1)$	$(\overline{x}_3 \vee \overline{y}_1)$	$(\overline{x}_3 \vee \overline{y}_2)$

Proposition 7. *Solving a Max-CSP instance is equivalent to solving the Partial Max-SAT problem of its direct encoding.*

PROOF: The hard clauses ensure that there is a one-to-one mapping between the set of CSP assignments and the set of feasible assignments of the Partial Max-SAT instance. Moreover, at most one of the conflict clauses that encode a certain constraint can be falsified by a feasible assignment. If the CSP assignment satisfies a constraint, then the corresponding Boolean assignment also satisfies the conflict clauses that encode that constraint because there is no clause forbidding allowed values. If the CSP assignment violates a constraint, then the corresponding Boolean assignment does not satisfy the conflict clause that encodes the forbidden values of the two variables involved in the constraint, and satisfies the remaining clauses. ∎

4.2 Support encoding for Partial Max-SAT

The support encoding for Partial Max-SAT may be defined by adapting the minimal support encoding from CSP into SAT:

Definition 8. The *minimal support encoding* of a Max-CSP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is the Partial Max-SAT instance that contains as hard clauses the corresponding at-least-one and at-most-one clauses for every CSP variable in $\mathcal{X}$, and contains as soft clauses the support clauses of the minimal support encoding from CSP into SAT.

Example 4. A minimal Partial Max-SAT support encoding for the Max-CSP problem of the CSP instance from Example 1 contains the following clauses:

at-least-one	$[x_1 \vee x_2 \vee x_3]$	$[y_1 \vee y_2 \vee y_3]$	
at-most-one	$[\overline{x}_1 \vee \overline{x}_2]$	$[\overline{x}_1 \vee \overline{x}_3]$	$[\overline{x}_2 \vee \overline{x}_3]$
	$[\overline{y}_1 \vee \overline{y}_2]$	$[\overline{y}_1 \vee \overline{y}_3]$	$[\overline{y}_2 \vee \overline{y}_3]$
support	$(\overline{x}_2 \vee y_2 \vee y_3)$	$(\overline{x}_3 \vee y_3)$	

Proposition 9. *Solving a Max-CSP instance is equivalent to solving the Partial Max-SAT problem of its minimal support encoding.*

PROOF: Proposition 5 proves that there is one unsatisfied clause for every violated constraint. Since the minimal support encoding is correct, and the hard clauses ensure a one-to-one mapping between Max-CSP and feasible Partial Max-SAT assignments, the Max-CSP optimal solutions are exactly the same as the Partial Max-SAT optimal solutions. ∎

We now define how to adapt to Partial Max-SAT the support encoding from CSP into SAT.

Definition 10. The *support encoding* of a Max-CSP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is the Partial Max-SAT instance that contains as hard clauses the corresponding at-least-one and at-most-one clauses for every CSP variable in $\mathcal{X}$, and contains, for every constraint $C_k \in \mathcal{C}$ with scope $\{X, Y\}$, a soft clause of the form $S_{X=j} \vee c_k$ for every support clause $S_{X=j}$ encoding the support for the value j of the CSP variable X, where c_k is an auxiliary variable , and contains a soft clause of the form $S_{Y=m} \vee \overline{c}_k$ for every support clause $S_{Y=m}$ encoding the support for the value m of the CSP variable Y.

Observe that we introduce an auxiliary variable for every constraint. This is due to the fact that there are two unsatisfied soft clauses for every violated constraint of the Max-CSP instance if we do not introduce auxiliary variables. It is particularly important to have one unsatisfied clause for every violated constraints when mapping weighted Max-CSP instances into weighted Max-SAT instances.[1] In this case, all the clauses encoding a certain constraint have as weight the weight associated to that constraint. When a constraint is violated with weight w, this guarantees that there is exactly one unsatisfied clause with weight w.

Example 5. The Partial Max-SAT support encoding for the Max-CSP problem of the CSP instance from Example 1 is as follows:

at-least-one	$[x_1 \vee x_2 \vee x_3]$	$[y_1 \vee y_2 \vee y_3]$	
at-most-one	$[\overline{x}_1 \vee \overline{x}_2]$	$[\overline{x}_1 \vee \overline{x}_3]$	$[\overline{x}_2 \vee \overline{x}_3]$
	$[\overline{y}_1 \vee \overline{y}_2]$	$[\overline{y}_1 \vee \overline{y}_3]$	$[\overline{y}_2 \vee \overline{y}_3]$
support	$(\overline{x}_2 \vee y_2 \vee y_3 \vee c_1)$	$(\overline{y}_1 \vee x_1 \vee \overline{c}_1)$	
	$(\overline{x}_3 \vee y_3 \vee c_1)$	$(\overline{y}_2 \vee x_1 \vee x_2 \vee \overline{c}_1)$	

Proposition 11. *Solving a Max-CSP instance is equivalent to solving the Partial Max-SAT problem of its support encoding.*

PROOF: By introducing auxiliary variables we ensure that the optimal solutions of Max-CSP are exactly the same as the optimal solutions of Partial Max-SAT. The auxiliary variables allow to violate exactly one clause for every violated constraint. ∎

5 Experimental results

We conducted an empirical evaluation to assess the impact of the defined encodings on the performance of two of the best performing Partial Max-SAT solvers: MiniMaxSat [9] and PMS [2]. The evaluation was performed on a cluster with 160 2 GHz AMD Opteron 248 Processors with 1 GB of memory.

As benchmarks we considered binary CSPs, which were obtained with a generator of uniform random binary CSPs[2]

---

[1]In weighted Max-CSP (Max-SAT), each constraint (clause) has a weight and the goal is to minimize the sum of the weights of the violated constraints (falsified clauses).
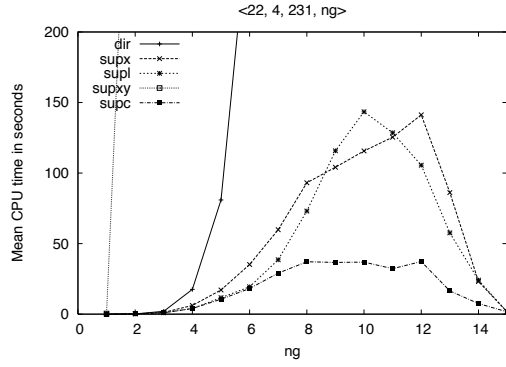
[2]http://www.lirmm.fr/~bessiere/generator.html

Figure 1. Experimental results for MiniMaxSat
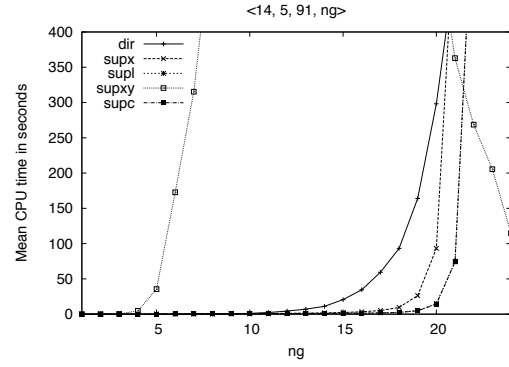

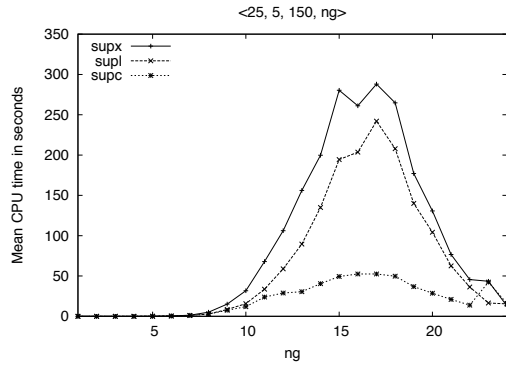**Figure 3. Experimental results for PMS**

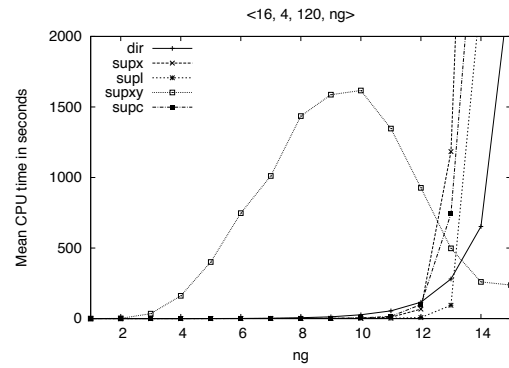**Figure 2. Experimental results for MiniMaxSat**

Figure 4. Experimental results for PMS

—designed and implemented by Frost, Bessière, Dechter and Regin— that implements the so-called model B: in the class $\langle n, d, p_1, p_2 \rangle$ with n variables of domain size d, we choose a random subset of exactly $p_1 n(n-1)/2$ constraints (rounded to the nearest integer), each with exactly $p_2 d^2$ conflicts (rounded to the nearest integer); p_1 may be thought of as the *density* of the problem and p_2 as the *tightness* of constraints. The difficulty of the instances depends on the selected values for n, d, p_1 and p_2. We selected values that allowed to solve the instances in a reasonable amount of time in each solver.

We used the following encodings: the direct encoding (`dir`), the support encoding (`supxy`), and three variants of the minimal support encoding (`supx`, `supl`, `supc`). The encoding `supx` refers to the minimal support encoding of a binary CSP containing only the support clauses for the CSP variable X and not for the variable Y for every constraint with scope $\{X, Y\}$; we do not show results for the encoding containing only support clauses for the CSP variable Y because its behaviour is very close to `supx` for

the solved random instances. The encoding `supl` refers to the minimal support encoding containing, for each constraint, the support clauses for the variable that produces a smaller total number of literals. The encoding `supc` refers to the minimal support encoding containing, for each constraint, the support clauses for the variable that produces smaller size clauses; we give a score of 16 to unit clauses, a score of 4 to binary clauses and a score of 1 to ternary clauses, and choose the variable with higher sum of scores. For instance, given the CSP instance $\langle X, D, C \rangle$, where $X = \{X, Y\}, d(X) = d(Y) = \{1, 2, 3, 4\}, C = \{C_{XY}\} = \{\{(1, 2), (1, 3), (1, 4)\}\}$, `supc` prefers three binary support clauses $x_1 \vee \overline{y}_2, x_1 \vee \overline{y}_3, x_1 \vee \overline{y}_4$ rather than the quaternary support clause $\overline{x}_1 \vee y_2 \vee y_3 \vee y_4$, while `supl` prefers $\overline{x}_1 \vee y_2 \vee y_3 \vee y_4$.

In the first experiment we solved 100 Max-CSP instances with MiniMaxSat for each data point; the instances had 22 variables, domains of 4 elements, 231 constraints and variable tightness (we vary the number of nogoods (ng)). We compared all the defined encodings of Max-CSP into Partial Max-SAT. The results are shown in Figure 1.

In the second experiment we solved 100 Max-CSP in-

stances with MiniMaxSat for each data point; the instances had 25 variables, domains of 5 elements, 150 constraints and variable tightness. We compared all the defined encodings of Max-CSP into Partial Max-SAT. The results are shown in Figure 2. We omit the results for the encodings `dir` and `supxy` because they are not competitive.

In the third experiment we solved 100 Max-CSP instances with PMS for each data point; the instances had 14 variables, domains of 5 elements, 91 constraints and variable tightness. We compared all the defined encodings of Max-CSP into Partial Max-SAT. The results are shown in Figure 3.

In the fourth experiment we solved 100 Max-CSP instances with PMS for each data point; the instances had 16 variables, domains of 4 elements, 120 constraints and variable tightness. We compared all the defined encodings of Max-CSP into Partial Max-SAT. The results are shown in Figure 4.

We observe that support encodings from Max-CSP into Partial Max-SAT, which have been introduced for the first time in this paper, outperform the direct encoding for both solvers. In MiniMaxSat, the best performing encoding is the minimal support encoding. Among the different versions of the minimal support encoding, we observe that `supc` is up to 6 times faster than the other two encodings (`supx` and `supl`). In PMS, the best encoding for high values of tightness is the support encoding while the best encodings for lower values are `supc` and `supl` in Figure 4, and `supl` in Figure 3.

6 Conclusions

We have defined the minimal support encoding, which is a new encoding from CSP into SAT, and a number of original encodings (`dir`, `supxy`, `supx`, `supl`, `supc`) that map Max-CSP instances into partial Max-SAT instances, and have provided experimental evidence that different Max-SAT encodings for a given optimization problem may produce substantial differences on the performance of a solver. Since our mappings produce one unsatisfied clause for every violated constraints, they can be easily extended to mappings from weighted Max-CSP instances into weighted Max-SAT instances; all the clauses encoding a certain constraint should have as weight the weight associated to that constraint.

To the best of our knowledge, this is the first paper that addresses the question of how to encode Max-CSP into Max-SAT, and analyzes the impact of modelling on the performance of Max-SAT solvers. Future research directions include analyzing the degree of local consistency achieved by each encoding, comparing with benchmarks other than random binary Max-CSP instances, and generalizing our results to n-ary constraints and many-valued Max-SAT.

References

[1] J. Argelich and F. Manyà. Exact Max-SAT solvers for over-constrained problems. *Journal of Heuristics*, 12(4–5):375–392, 2006.

[2] J. Argelich and F. Manyà. Partial Max-SAT solvers with clause learning. In *the 10th International Conference on Theory and Applications of Satisfiability Testing, SAT-2007*, pages 28–40. LNCS 4501, 2007.

[3] C. Bessière, E. Hebrard, and T. Walsh. Local consistencies in SAT. In *the 6th International Conference on the Theory and Applications of Satisfiability Testing, SAT-2003*, pages 299–314. LNCS 2919, 2003.

[4] M. L. Bonet, J. Levy, and F. Manyà. A complete calculus for Max-SAT. In *the 9th International Conference on Theory and Applications of Satisfiability Testing, SAT-2006*, pages 240–251. LNCS 4121, 2006.

[5] M. L. Bonet, J. Levy, and F. Manyà. Resolution for Max-SAT. *Artificial Intelligence*, 171(8–9):240–251, 2007.

[6] M. Gavanelli. The log-support encoding of CSP into SAT. In *the 13th International Conference on Principles and Practice of Constraint Programming, CP-2007*, pages 815–822. LNCS 4741, 2007.

[7] R. Génisson and P. Jégou. Davis and Putnam were already checking forward. In *the 12th European Conference on Artificial Intelligence (ECAI)*, pages 180–184, 1996.

[8] I. P. Gent. Arc consistency in SAT. In *the 15th European Conference on Artificial Intelligence (ECAI), Lyon, France*, pages 121–125, 2002.

[9] F. Heras, J. Larrosa, and A. Oliveras. MiniMaxSat: A new weighted Max-SAT solver. In *the 10th International Conference on Theory and Applications of Satisfiability Testing, SAT-2007*, 2007.

[10] S. Kasif. On the parallel complexity of discrete relaxation in constraint satisfaction networks. *Artificial Intelligence*, 45:275–286, 1990.

[11] J. Larrosa, F. Heras, and S. de Givry. A logical approach to efficient max-sat solving. *Artificial Intelligence*, 172(2–3):204–233, 2008.

[12] C. M. Li, F. Manyà, and J. Planes. Exploiting unit propagation to compute lower bounds in branch and bound Max-SAT solvers. In *the 11th International Conference on Principles and Practice of Constraint Programming, CP-2005*, pages 403–414. LNCS 3709, 2005.

[13] C. M. Li, F. Manyà, and J. Planes. Detecting disjoint inconsistent subformulas for computing lower bounds for Max-SAT. In *the 21st National Conference on Artificial Intelligence, AAAI-2006*, pages 86–91, 2006.

[14] C. M. Li, F. Manyà, and J. Planes. New inference rules for Max-SAT. *Journal of Artificial Intelligence Research*, 30:321–359, 2007.

[15] T. Walsh. SAT v CSP. In *the 6th International Conference on Principles of Constraint Programming, CP-2000*, pages 441–456. LNCS 1894, 2000.