2017 IEEE 47th International Symposium on Multiple-Valued Logic

Exploiting Many-Valued Variables in MaxSAT *

Josep Argelich
Universitat de Lleida

Chu Min Li
MIS, Université de Picardie

Felip Manyà
IIIA-CSIC

Abstract

Solving combinatorial optimization problems by reducing them to MaxSAT has shown to be a competitive problem solving approach. Since a lot of optimization problems have many-valued variables, we propose to exploit the domain information of the many-valued variables to enhance MaxSAT-based problem solving: first, we define a new way of encoding weighted maximum constraint satisfaction problems to both Boolean MaxSAT and many-valued MaxSAT; and second, we define a variable selection heuristic that takes into account the domain information and allow us to easily implement a many-valued MaxSAT solver. Moreover, the empirical results provide evidence of the good performance of the new encodings and the new branching heuristic on a representative set of instances.

1 Introduction

There has been tremendous progress in theoretical and applied aspects of the MaxSAT problem over the last decade. As a result, there are now a number of competitive solvers that are able to solve challenging optimization problems in different areas (see e.g. [1, 2, 9, 16] and the references therein for previous and related work).

Given that solving combinatorial optimization problems by reducing them to MaxSAT has shown to be a competitive generic problem solving approach for a number of domains, and that many problems are more naturally represented with many-valued variables instead of Boolean variables, we believe that it is worth to investigate the extension of existing Boolean MaxSAT results to the many-valued framework.

In this paper we propose to exploit the domain information of many-valued variables to enhance MaxSAT-based problem solving: first, we define a new way of encoding Weighted Maximum Constraint Satisfaction Problems (WMaxCSP) to both Boolean MaxSAT and many-valued MaxSAT; and second, we define a variable selection heuristic that takes into account the domain information and allows us to easily implement a many-valued MaxSAT solver. Moreover, the empirical results provide evidence of the good performance of the new encodings and the new branching heuristic on a representative set of instances.

The paper is structured as follows. Section 2 defines some basic concepts. Section 3 presents new encodings from WMaxCSP to many-valued MaxSAT, and new encodings from WMaxCSP to Boolean MaxSAT. Section 4 describes Mv-MaxSatz. Section 5 reports on the empirical investigation. Section 6 gives the conclusions.

2 Preliminaries

Given a set of variables $\{x_1, \ldots, x_n\}$, a Boolean literal is a variable x_i or its negation $\neg x_i$. A many-valued literal is an expression of the form $x_j = i$ or $x_j \neq i$, where i belongs to a truth value set or domain N such that $|N| \geq 2$. A weighted (Boolean/many-valued) clause is a pair (c, w), where c is a disjunction of (Boolean/many-valued) literals and w, its weight, is a positive integer or infinity. If its weight is infinity, it is a hard clause (we omit infinity weights for simplicity); otherwise it is a soft clause. A (Boolean/many-valued) Weighted Partial MaxSAT instance is a multiset of weighted (Boolean/many-valued) clauses.

A Boolean assignment assigns to each variable either 0 or 1, and a many-valued assignment assigns to each variable an element of N. A Boolean assignment satisfies x_i ($\neg x_i$) if x_i evaluates to 1 (0); and a many-valued assignment satisfies $x_j = i$ if x_j evaluates to i; otherwise it satisfies $x_j \neq i$. A (Boolean/many-valued) assignment satisfies a weighted clause (c, w) if it satisfies a literal of c, and satisfies a multiset of clauses if it satisfies all its clauses.

The (Boolean/many-valued) Weighted Partial MaxSAT problem, or WPMaxSAT, for an instance ϕ is to find a (Boolean/many-valued) assignment that satisfies the hard clauses and minimizes the sum of the weights of the unsatisfied soft clauses. The most common subproblems of WPMaxSAT are the following: Weighted MaxSAT (WMaxSAT), which is WPMaxSAT without hard clauses; Partial MaxSAT (PMaxSAT), which is WPMaxSAT when

*This work was supported by the Generalitat de Catalunya grant AGAUR 2014-SGR-118, and the MINECO-FEDER project RASO TIN2015-71799-C2-1-P/2-P. The third author was supported by Mobility Grant PRX16/00215 of the Ministerio de Educación, Cultura y Deporte.

2378-2226/17 $31.00 © 2017 IEEE
DOI 10.1109/ISMVL.2017.42

all the soft clauses have the same weight, and MaxSAT, which is PMaxSAT without hard clauses.

The (Boolean/many-valued) Weighted Partial MinSAT problem, or WPMinSAT, is to find a (Boolean/many-valued) assignment that satisfies the hard clauses and maximizes the sum of the weights of the unsatisfied soft clauses. It is the dual problem of WPMaxSAT, and is used in Section 3 to define more efficient MaxSAT encodings. The most common subproblems of WPMinSAT, which are defined as in the WPMaxSAT case, are WMinSAT, PMinSAT and MinSAT [14, 15].

A Constraint Satisfaction Problem (CSP) instance is 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, and $\mathcal{C} = \{C_1, \ldots, C_m\}$ is a set of constraints. Each $C_i = \langle S_i, R_i \rangle$ in $\mathcal{C}$ is a relation R_i over a subset of $S_i = \{X_{i_1}, \ldots, X_{i_k}\} \subseteq \mathcal{X}$, called the scope of C_i. R_i may be represented extensionally as a subset of the Cartesian product $d(X_{i_1}) \times \cdots \times d(X_{i_k})$. The tuples belonging to R_i represent the allowed values and are called goods, and the rest of tuples represent the forbidden values and are called nogoods. An assignment v for a CSP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is a mapping that assigns to each variable $X_i \in \mathcal{X}$ an element $v(X_i) \in d(X_i)$. It satisfies a constraint $\langle \{X_{i_1}, \ldots, X_{i_k}\}, R_i \rangle \in C$ iff $\langle v(X_{i_1}), \ldots, v(X_{i_k}) \rangle \in R_i$. The CSP for an instance P is to find a satisfying assignment for P.

A Weighted MaxCSP (WMaxCSP) instance is defined as a triple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{X}$ and $\mathcal{D}$ are variables and domains as in CSP, and $\mathcal{C}$ is a set of weighted constraints. A weighted constraint $\langle C, w \rangle$ is just a classical constraint C plus a weight w over a finite set of natural numbers. The cost of an assignment v is the sum of the weights of all constraints violated by v. An optimal solution of WMaxCSP for an instance P is an assignment for P with minimal cost.

3 Encodings from WMaxCSP to Many-Valued WMaxSAT

WMaxCSP is a framework for modeling optimization problems, and a remarkable collection of practical problems is available under this formalism. Because of that, it is important to define efficient encodings from WMaxCSP to WPMaxSAT, as was done from CSP to SAT.

We first define the many-valued counterparts of the direct and minimal support encodings from WMaxCSP to Boolean WPMaxSAT of [5], and then the new hybrid encodings. All the mentioned encodings are correct: solving a WMaxCSP instance is equivalent to solving the instance derived by any of our encodings. It is important to highlight that the minimal support encoding is only valid for binary constraints whereas the direct encoding is valid for constraints involving an arbitrary number of variables.

In the following, we associate a many-valued variable x_i with each CSP variable X_i. Besides, we assume that all the CSP variables have the same domain $d(X)$, and so the truth value set $N = d(X)$. If the CSP variables have different domains, then $N = \bigcup_{i=1}^{n} d(X_i)$, and we should add the unit clauses $x_i \neq j$ for all j such that $j \in N$ and $j \notin d(X_i)$.

We note that, for encoding WMaxCSP, hard clauses are needed in the Boolean case but not in the many-valued case. The hard clauses state that exactly one of the Boolean variables associated with a given CSP variable is true. Because of that we talk about encodings from WMaxCSP to Boolean WPMaxSAT, and encodings from WMaxCSP to many-valued WMaxSAT.

Definition 1. The **many-valued WMaxSAT direct encoding (*dir*)** of a WMaxCSP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ has a clause $(x_1 \neq i_1 \vee \cdots \vee x_m \neq i_m, w)$ for each nogood $(X_1 = i_1, \ldots, X_m = i_m)$ of each constraint $\langle C, w \rangle$ of $\mathcal{C}$ with scope $\{X_1, \ldots, X_m\}$.

The correctness of the encoding follows from the fact that there is a one-to-one mapping between CSP and many-valued assignments, and from the fact that the encoding forces the violation of exactly one clause for each violated constraint, and all the clauses are satisfied when the constraint is satisfied. So, minimizing the sum of the weights of the violated clauses is equivalent to minimizing the sum of the weights of the violated constraints. Moreover, from an optimal many-valued interpretation we derive an optimal CSP interpretation by assigning the value i to the CSP variable X iff the many-valued interpretation satisfies $x = i$.

Example 1. Let $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ be the WMaxCSP instance where $\mathcal{X} = \{X, Y\}$, $d(X) = d(Y) = \{1, 2, 3\}$, and $\mathcal{C} = \{\langle X \neq Y, 3 \rangle\}$. The many-valued WMaxSAT direct encoding contains the clauses $(x \neq 1 \vee y \neq 1, 3)$, $(x \neq 2 \vee y \neq 2, 3)$, and $(x \neq 3 \vee y \neq 3, 3)$.

In the support encodings from CSP to SAT and from WMaxCSP to Boolean WPMaxSAT, which are only valid for binary constraints, there are clauses that encode the support for a value instead of encoding conflicts via nogoods. The support for a value i of a variable X across a binary constraint with scope $\{X, Y\}$ is the set of values of Y which allow $X = i$. If $v_1, \ldots, v_k$ are the supporting values of variable Y for $X = i$, the clause $\neg x_i \vee y_{v_1} \vee \cdots \vee y_{v_k}$ (called support clause) is added. There is one support clause for each value in the domain and for each pair of variables X, Y involved in a constraint. In the support encoding, a clause in each direction is used: one for the pair X, Y and one for Y, X. However, in the minimal support encoding, the added clauses are the support clauses either for all the domain values of either X or Y. We focus on the minimal support encoding because it performed better in [5] and in our tests.

Definition 2. The **many-valued WMaxSAT minimal support encoding (*sup*)** of a binary WMaxCSP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ contains, for each constraint $\langle C, w \rangle$ of $\mathcal{C}$ with scope $\{X, Y\}$, either the clause $(x \neq i \vee y = v_1 \vee \cdots \vee y = v_n, w)$ for each $i \in d(X)$, where $v_1, \ldots, v_n$ is the support for i, or the clause $(y \neq j \vee x = u_1 \vee \cdots \vee x = u_m, w)$ for each $j \in d(Y)$, where $u_1, \ldots, u_m$ is the support for j.

The many-valued WMaxSAT minimal support encoding also forces the violation of one clause for each violated constraint. Observe that we have to select the support clauses either for the variable X or for the variable Y. In the experiments, we select the variable which produces clauses of smaller size: we give a score of 16 to unit clauses, a score of 4 to binary clauses, a score of 1 to ternary clauses and a score of 0 to the rest of clauses, and then select the variable with the higher sum of scores.

Example 2. Let $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ be the WMaxCSP instance where $\mathcal{X} = \{X, Y\}$, $d(X) = d(Y) = \{1, 2, 3\}$, and $\mathcal{C} = \{\langle X \neq Y, 2 \rangle\}$. The many-valued WMaxSAT minimal support encoding contains the clauses $(x \neq 1 \vee y = 2 \vee y = 3, 2)$, $(x \neq 2 \vee y = 1 \vee y = 3, 2)$, and $(x \neq 3 \vee y = 1 \vee y = 2, 2)$. We could also define this encoding by adding the support clauses for the domain values of Y instead of X: $(y \neq 1 \vee x = 2 \vee x = 3, 2)$, $(y \neq 2 \vee x = 1 \vee x = 3, 2)$, and $(y \neq 3 \vee x = 1 \vee x = 2, 2)$.

The experiments about mappings from WMaxCSP to Boolean WPMaxSAT instances described in the literature indicate that both the encoding size and the inference achieved with a particular encoding are decisive in the performance of the encoding [5]. To produce encodings of smaller size whenever possible, we take a different approach: we do not apply the same encoding to the whole WMaxCSP [6]. Instead of that, we choose the most convenient encoding for each individual constraint, and refer to such encodings as hybrid encodings. For example, the constraint $X \neq Y$ has a linear number of literal occurrences in the domain size for the direct encoding whereas it has a quadratic number for the minimal support encoding. In contrast, the constraint $X = Y$ has a quadratic number of literal occurrences for the direct encoding and a linear number for the minimal support encoding.

Our first hybrid encoding, called binary hybrid encoding, combines the direct and minimal support encodings, and is only valid for binary constraints. It relies on the following observation: When the number of nogoods ($\#nogoods$) is low, we get smaller encodings with the direct encoding, but when the number of goods ($\#goods$) is low, we get smaller encodings with the minimal support encoding. On the other hand, the minimal support encoding outperforms the direct encoding on a wide range of benchmarks [5], presumably due to the inference achieved. So, we propose a hybrid encoding that gives priority to the minimal support encoding over the direct encoding but prefers the direct encoding when the number of nogoods is much smaller. We control this situation with a parameter k in the definition below; k was set to 0.3 in our experiments.

Definition 3. The **many-valued WMaxSAT binary hybrid encoding (*hyb2*)** of a binary WMaxCSP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ contains, for each constraint $\langle C, w \rangle$ of $\mathcal{C}$ with scope $\{X, Y\}$, the many-valued WMaxSAT direct encoding of $\langle C, w \rangle$ if $\#nogoods < k \times \#goods$, where $k \in (0, 1)$; otherwise, it contains the many-valued WMaxSAT minimal support encoding of $\langle C, w \rangle$.

Example 3. Let $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ be the WMaxCSP instance where $\mathcal{X} = \{X, Y, Z\}$, $d(X) = d(Y) = d(Z) = \{1, \ldots, 7\}$, $\mathcal{C} = \{\langle X \neq Y, 2 \rangle, \langle X = Z, 3 \rangle\}$. The many-valued WMaxSAT binary hybrid encoding uses the direct encoding for the constraint $X \neq Y$ because $\#nogoods = 7$ and $\#goods = 42$ ($\#nogoods/\#goods = 0.16 < 0.3$), and the minimal support encoding for the constraint $X = Z$ because $\#nogoods = 42$ and $\#goods = 7$ ($\#nogoods/\#goods = 6 \geq 0.3$). Hence, the encoding contains the clauses $(x \neq 1 \vee y \neq 1, 2), (x \neq 2 \vee y \neq 2, 2), \ldots, (x \neq 7 \vee y \neq 7, 2), (x \neq 1 \vee z = 1, 3), (x \neq 2 \vee z = 2, 3), \ldots, (x \neq 7 \vee z = 7, 3)$.

Next we define the n-ary hybrid encoding, which is valid for non-binary constraints too. Recall that the only available option for non-binary constraints is the direct encoding. The idea behind this new encoding is to exploit the complementarity between MaxSAT and MinSAT with the ultimate goal of producing an encoding with as few n-ary clauses as possible.

The first observation is that the many-valued WMinSAT counterpart encoding of the many-valued WMaxSAT direct encoding adds a clause for every good instead of adding a clause for every nogood [7]. This means that, for a given constraint, the many-valued WMinSAT direct encoding is smaller than the many-valued WMaxSAT encoding when the number of goods is smaller than the number of nogoods. Since the goal of WMinSAT is to maximize the sum of the weights of the violated clauses, the many-valued WMinSAT direct encoding is correct because it violates exactly one clause for each satisfied constraint, and no clause is violated if the constraint is violated. Thus, maximizing the sum of the weights of the violated clauses is equivalent to maximize the sum of the weights of the satisfied constraints. In other words, minimizing the sum of the weights of the satisfied clauses is equivalent to minimize the sum of the weights of the violated constraints.

On the other hand, any many-valued WMinSAT instance can be transformed into an equivalent many-valued

WMaxSAT instance as follows [8, 18]: replace each clause $c = (l_1 \neq i_1 \vee l_2 \neq i_2 \vee \cdots \vee l_m \neq i_m, w)$ by the set of clauses $S = \{(l_1 = i_1, w), (l_1 \neq i_1 \vee l_2 = i_2, w), \ldots, (l_1 \neq i_1 \vee l_2 \neq i_2 \vee \cdots \vee l_m = i_m, w)\}$. It holds that an assignment satisfies c iff it violates exactly one clause of S, and violates c iff it satisfies all the clauses of S. So, for a constraint with k goods and weight w, an assignment violates $k-1$ clauses of the above transformation for each satisfied constraint, and violates k clauses for each violated constraint. Therefore, the difference of the sum of weights of violated clauses between violating and satisfying a constraint is w. Thus, if a WMaxCSP instance P is first translated into a many-valued WMinSAT instance P_{minsat} and then P_{minsat} is translated into a many-valued WMaxSAT instance P_{maxsat} using the above transformation, it turns out that any optimal many-valued assignment of P_{maxsat} is an optimal CSP assignment of P, and vice versa. Observe that the minimum sum of weights of the violated clauses in P_{maxsat} can be different from the minimum sum of weights of the violated constraints in P. We just need to calculate the weights in P after instantiating its variables with an optimal assignment.

Definition 4. The **many-valued WMaxSAT n-ary hybrid encoding (*hybN*)** of a WMaxCSP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ contains, for each constraint $\langle C, w \rangle$ of $\mathcal{C}$ with scope $\{X_1, \ldots, X_n\}$, the encoding with the smaller number of n-ary clauses between the many-valued WMaxSAT direct encoding of $\langle C, w \rangle$ and the encoding obtained after transforming the many-valued WMinSAT direct encoding of $\langle C, w \rangle$ into many-valued WMaxSAT.

Example 4. Let $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ be the WMaxCSP instance where $\mathcal{X} = \{X, Y\}$, $d(X) = d(Y) = \{1, \ldots, 8\}$, and $\mathcal{C} = \{\langle X = Y, 5 \rangle\}$. The many-valued WMaxSAT direct encoding has 56 clauses ($\{(x \neq i \vee y \neq j, 5) \,|\, i, j \in d(X), i \neq j\}$) while the many-valued WMinSAT direct encoding has 8 clauses ($\{(x \neq i \vee y \neq i, 5) \,|\, i \in d(X)\}$). The transformation into WMaxSAT of the WMinSAT encoding produces 16 clauses ($\{(x = i, 5), (x \neq i \vee y = i, 5) \,|\, i \in d(X)\}$). Thus, the last encoding is selected. Observe that if $\mathcal{C} = \{\langle X \neq Y, 5 \rangle\}$, then we should select the many-valued WMaxSAT direct encoding.

The direct and minimal support encodings from WMaxCSP to Boolean WPMaxSAT were defined in [5], but no Boolean hybrid MaxSAT encoding has been defined so far. To obtain the direct and minimal support encodings from WMaxCSP to Boolean WPMaxSAT from their many-valued counterparts, as well as the new Boolean binary and n-ary hybrid encodings, we must associate the Boolean literal x_i with the many-valued literal $x = i$, and the Boolean literal $\neg x_i$ with the many-valued literal $x \neq i$. Besides, for each CSP variable X with domain $\{1, \ldots, m\}$, the following hard clauses must be added: (i) $x_1 \vee \cdots \vee x_m$ (ALO clause), and (ii) $\{\neg x_i \vee \neg x_j | 1 \leq i < j \leq m\}$(AMO clauses). The ALO (at-least-one) and AMO (at-most-one) clauses ensure that exactly one of the Boolean variables associated with a CSP variable evaluates to true in each feasible assignment.

4 The Solver Mv-MaxSatz

Mv-MaxSatz is a many-valued WPMaxSAT solver built on top of the Boolean WPMaxSAT solver MaxSatz [10, 13]. We converted a Boolean WPMaxSAT solver into a many-valued WPMaxSAT solver because it is much easier than developing a solver from scratch, and because in this way new solving Boolean WPMaxSAT techniques may be easily incorporated into many-valued WPMaxSAT.

Given an input many-valued WPMaxSAT instance, Mv-MaxSatz works as follows: (i) It generates the corresponding Boolean WPMaxSAT encoding as explained at the end of the previous section, and records the set of Boolean variables associated with each many-valued variable; (ii) it finds an optimal Boolean WPMaxSAT solution with Mv-MaxSatz; and (iii) it returns an optimal solution to the input many-valued WPMaxSAT instance, derived from the solution in the previous step.

MaxSatz implements the branch-and-bound scheme, and the search space is formed by a tree representing all the possible truth assignments. We refer to [10, 13] for a detailed description, due to lack of space. Its main characteristic is that it computes lower bounds by detecting disjoint inconsistent subsets with unit propagation [11, 12], and its goal is to minimize the number of violated clauses. MaxSatz selects the branching variables applying the following heuristic: Let $hard(l)$ ($soft(l)$) be the number of occurrences of literal l in hard (soft) clauses, and let $score(l) = 2 \times hard(l) + soft(l)$. It chooses a variable x with the highest value of $score(x) \times score(\neg x) + score(x) + score(\neg x)$.

Mv-MaxSatz differs in two aspect from MaxSatz: (i) It updates the state of the set of Boolean variables associated with a many-valued variable by removing the Boolean variables which are instantiated to false; in this way, the solver knows the current domain of each many-valued variable; and (ii) it considers the domain information in the branching heuristic: Firstly, it creates a candidate set of Boolean variables formed by the variables that encode many-valued variables of minimum domain size. Secondly, it applies the original variable selection heuristic of MaxSatz only to the Boolean variables belonging to the candidate set. So, Mv-MaxSatz is able to exploit the structural information about the domain that is hidden in standard Boolean encodings. As we show in the experiments, this new heuristic allows one to achieve significant speed-ups. A similar approach to build many-valued solvers was applied in [3, 4, 6].

Table 1. Results for binary WMaxCSPs solved with MaxSatz and Mv-MaxSatz comparing encodings *hyb2* and *sup*. Instances have 25 variables of domain size 5, a number of constraints ranging from 230 to 280, and a random number of goods per constraint ranging from 1 to 24. Timeout: 1800 seconds. Mean time in seconds.

(n, d, c)	#	Mv-MaxSatz (*hyb2*)	Mv-MaxSatz (*sup*)	MaxSatz (*hyb2*)	MaxSatz (*sup*)
(25, 5, 230)	100	**910 (68)**	1027 (63)	1152 (49)	1203 (48)
(25, 5, 240)	100	**1055 (41)**	1083 (28)	1342 (32)	1204 (28)
(25, 5, 250)	100	1021 (22)	**1187 (23)**	1308 (20)	1330 (20)
(25, 5, 260)	100	**1228 (16)**	1111 (10)	1253 (8)	1256 (7)
(25, 5, 270)	100	**1276 (9)**	1286 (7)	1443 (1)	1621 (2)
(25, 5, 280)	100	**1546 (3)**	0 (0)	1223 (1)	1276 (2)
Total instances	600	**159**	131	111	107

Table 2. Results for binary WMaxCSPs solved with MaxSatz and Mv-MaxSatz comparing encodings *hybN* and *dir*. Instances have 22 variables of domain size 5, a number of constraints ranging from 150 to 200, and a random number of goods per constraint ranging from 1 to 24. Timeout: 1800 seconds. Mean time in seconds.

(n, d, c)	#	Mv-MaxSatz (*hybN*)	Mv-MaxSatz (*dir*)	MaxSatz (*hybN*)	MaxSatz (*dir*)
(22, 5, 150)	100	**22 (100)**	57 (100)	500 (98)	1786 (1)
(22, 5, 160)	100	**36 (100)**	107 (100)	628 (91)	0 (0)
(22, 5, 170)	100	**67 (100)**	201 (99)	820 (70)	0 (0)
(22, 5, 180)	100	**109 (100)**	322 (100)	1142 (51)	0 (0)
(22, 5, 190)	100	**162 (100)**	519 (97)	1244 (24)	0 (0)
(22, 5, 200)	100	**24 (100)**	661 (97)	1334 (10)	0 (0)
Total instances	600	**600**	593	344	1

5 Empirical Investigation

We compared the performance of the following encodings from WMaxCSP to Boolean WPMaxSAT and many-valued WMaxSAT: direct encoding (*dir*), minimal support encoding (*sup*), binary hybrid encoding (*hyb2*) and n-ary hybrid encoding (*hybN*). The Boolean encodings were solved with MaxSatz and the many-valued encodings were solved with Mv-MaxSatz. Experiments were executed on a cluster with Intel Xeon CPU E5-2620 @ 2GHz processors with 4GB of RAM.

We solved sets of 100 binary and ternary WMaxCSP instances generated using the so-called model B [17], considering that all the constraints have the same weight. For an instance with n variables of domain size d, we choose a random subset of exactly c constraints with scope $\{X_1, \ldots, X_k\}$, with a random number of nogoods ranging from 1 to $d^k - 1$.

In the tables showing the experimental results, the first column displays the input parameters of the generators, the second column displays the number of tested instances, and the rest of columns display, for the encoding and solver indicated in the first row, the mean CPU time needed to solve an instance among the instances solved within a cutoff time of 1800 seconds, followed by the total number of solved instances in parentheses. The best results are in bold.

Tables 1 and 2 show the results for binary WMaxCSPs, and Table 3 for ternary WMaxCSPs. Table 1 does not show results for the direct encoding and encoding *hybN* because no instance was solved within the cutoff time. As we mentioned above, the minimal support encoding is generally quite superior to the direct encoding. Because of that, we solved the instances in Table 2, which are easier, to compare encoding *hybN* with the direct encoding on binary constraints.

We draw the following conclusions from the results:

i) Many-valued MaxSAT outperforms Boolean MaxSAT. For binary WMaxCSP, 48 and 256 additional instances instances are solved if we take the best results for each case (Tables 1 and 2). For ternary WMaxCSPs, 396 additional instances are solved (Table 3). It is clear that the exploitation of the domain information is decisive for get-

Table 3. Results for ternary WMaxCSPs solved with MaxSatz and Mv-MaxSatz comparing encodings *hybN* and *dir*. Instances have 18 variables of domain size 5, a number of constraints ranging from 100 to 150, and a random number of goods per constraint ranging from 1 to 124. Timeout: 1800 seconds. Mean time in seconds.

(n, d, c)	#	Mv-MaxSatz (*hybN*)	Mv-MaxSatz (*dir*)	MaxSatz (*hybN*)	MaxSatz (*dir*)
(18, 5, 100)	100	**317 (98)**	645 (96)	1463 (10)	0 (0)
(18, 5, 110)	100	**444 (95)**	874 (63)	1617 (1)	0 (0)
(18, 5, 120)	100	**871 (92)**	1125 (31)	0 (0)	0 (0)
(18, 5, 130)	100	**985 (62)**	1343 (11)	0 (0)	0 (0)
(18, 5, 140)	100	**1151 (37)**	1278 (4)	0 (0)	0 (0)
(18, 5, 150)	100	**1386 (23)**	0 (0)	0 (0)	0 (0)
Total instances	600	**407**	205	11	0

ting faster solvers. It is also worth noticing that the cost of our approach to building many-valued solvers from Boolean solvers is very low.

ii) The hybrid encoding *hyb2* produces significant speed-ups. In the Boolean (many-valued) case, 5 (28) additional instances are solved.

iii) The hybrid encoding *hybN* has a highly competitive performance profile. For binary WMaxCSPs, it solves 343 instances more than the direct encoding in the Boolean case. For ternary WMaxCSPs, it solves 202 instances more than the direct encoding in the many-valued case.

6 Concluding Remarks

We showed that many-valued MaxSAT-based problem solving can be more competitive than Boolean MaxSAT-based problem solving. Besides, we provided an efficient approach to easily build a many-valued MaxSAT solver from a Boolean MaxSAT solver. We also defined original hybrid encodings from WMaxCSP to both many-valued MaxSAT and Boolean MaxSAT, which are much more efficient than the corresponding direct and support encodings.

References

[1] A. Abramé and D. Habet. On the resiliency of unit propagation to Max-Resolution. In *Proc. of IJCAI*, pages 268–274, 2015.

[2] C. Ansótegui, J. Gabàs, and J. Levy. Exploiting subproblem optimization in SAT-based MaxSAT algorithms. *J. Heuristics*, 22(1):1–53, 2016.

[3] C. Ansótegui, J. Larrubia, C. M. Li, and F. Manyà. Exploiting multivalued knowledge in variable selection heuristics for sat solvers. *Annals of Mathematics and Artificial Intelligence*, 49(1-4):191–205, 2007.

[4] C. Ansótegui, J. Larrubia, and F. Manyà. Boosting Chaff's performance by incorporating CSP heuristics. In *Proc. of CP*, pages 96–107, 2003.

[5] J. Argelich, A. Cabiscol, I. Lynce, and F. Manyà. Efficient encodings from CSP into SAT, and from MaxCSP into MaxSAT. *Multiple-Valued Logic and Soft Computing*, 19(1-3):3–23, 2012.

[6] J. Argelich, C. M. Li, F. Manyà, and Z. Zhu. Many-valued MinSAT solving. In *Proc. of ISMVL*, pages 32–37, 2014.

[7] J. Argelich, C. M. Li, F. Manyà, and Z. Zhu. MinSAT versus MaxSAT for Optimization Problems. In *Proc. of CP*, pages 133–142, 2013.

[8] A. Kügel. Natural Max-SAT encoding of Min-SAT. In *Proc. of LION 6*, 2012.

[9] C. M. Li and F. Manyà. MaxSAT, hard and soft constraints. In A. Biere, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, pages 613–631. IOS Press, 2009.

[10] C. M. Li, F. Manyà, N. O. Mohamedou, and J. Planes. Resolution-based lower bounds in MaxSAT. *Constraints*, 15(4):456–484, 2010.

[11] C. M. Li, F. Manyà, and J. Planes. Exploiting unit propagation to compute lower bounds in branch and bound Max-SAT solvers. In *Proc. of CP*, pages 403–414, 2005.

[12] C. M. Li, F. Manyà, and J. Planes. Detecting disjoint inconsistent subformulas for computing lower bounds for Max-SAT. In *Proc. of AAAI*, pages 86–91, 2006.

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

[14] C. M. Li, Z. Zhu, F. Manyà, and L. Simon. Minimum satisfiability and its applications. In *Proc. of IJCAI*, pages 605–610, 2011.

[15] C. M. Li, Z. Zhu, F. Manyà, and L. Simon. Optimizing with minimum satisfiability. *Artificial Intelligence*, 190:32–44, 2012.

[16] A. Morgado, F. Heras, M. Liffiton, J. Planes and J. Marques-Silva. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints*, 18(4):478–534, 2013.

[17] B. Smith and M. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, 81:155–181, 1996.

[18] Z. Zhu, C. M. Li, F. Manyà, and J. Argelich. A New Encoding from MinSAT into MaxSAT. In *Proc. of CP*, pages 455–463, 2012.