Many-Valued MinSAT Solving*

Josep Argelich
Univ. de Lleida

Chu Min Li
MIS, Univ. de Picardie

Felip Manyà
IIIA-CSIC

Zhu Zhu
MIS, Univ. de Picardie

Abstract

Solving combinatorial optimization problems via their reduction to Boolean MinSAT is an emerging generic problem solving approach. In this paper we extend MinSAT with many-valued variables, and refer to the new formalism as Many-Valued MinSAT. For Many-Valued MinSAT, we describe an exact solver, Mv-MinSatz, which builds on the Boolean branch-and-bound solver MinSatz, and exploits the domain information of many-valued variables. Moreover, we also define efficient and robust encodings from optimization problems with many-valued variables to MinSAT. The empirical results provide evidence of the good performance of the new encodings, and of Many-Valued MinSAT over Boolean MinSAT on relevant optimization problems.

1 Introduction

Boolean MinSAT is the problem of finding a truth assignment that minimizes the number of satisfied clauses in a Boolean CNF formula, and Boolean MaxSAT is the problem of finding a truth assignment that maximizes the number of satisfied clauses. Despite the similarity between these two problems, both the solving techniques and the encodings applied in MinSAT [13, 14, 6] are quite different from those applied in MaxSAT [5, 10]. Actually, MinSAT and MaxSAT are complementary rather than competing generic approaches for solving optimization problems.

In this paper we define Many-Valued MinSAT as the problem of minimizing the number of satisfied clauses in a many-valued CNF formula. The main advantage of Many-Valued MinSAT w.r.t. Boolean MinSAT, from a problem solving perspective, is that the domain information of problems containing many-valued variables is not hidden in the encoding, and this information is relevant because solvers can define more powerful variable selection heuristics using that variable domain information. Since creating a Many-Valued MinSAT solver from scratch would be very costly, in our first contribution, we create an exact Many-Valued MinSAT solver, Mv-MinSatz, by adapting the Boolean MinSAT solver MinSatz [14] to deal with many-valued CNF formulas, and exploit variable domain information.

In our second contribution, we define improved encodings from optimization problems with many-valued variables to Boolean/Many-Valued MinSAT. In particular, we define two new and robust encodings from Weighted MaxCSP to Weighted Partial MinSAT. The first encoding selects, for each constraint, the most efficient encoding from Weighted MaxCSP to Weighted Partial MinSAT, and is valid for binary constraints. The second encoding is a new direct encoding that combines the direct encodings from Weighted MaxCSP to both Weighted Partial MinSAT and Weighted Partial MaxSAT, and is valid for non-binary constraints too. The new encodings, in contrast to existing encodings, may select a different encoding for each constraint, and it is in this sense that we will say that they are hybrid.

In our third contribution, we report on an experimental investigation conducted to evaluate the previous contributions. The results obtained provide empirical evidence that our new hybrid encodings outperform the existing encodings, and Many-Valued MinSAT outperforms Boolean MinSAT on relevant optimization problems.

The paper is structured as follows. Section 2 defines basic concepts. Section 3 surveys recent work on MinSAT solving. Section 4 shows how to map Many-Valued WPMinSAT to Boolean WPMinSAT. Section 5 describes Mv-MinSatz. Section 6 presents existing encodings from Weighted MaxCSP to Weighted Partial MinSAT, and defines two original hybrid encodings. Section 7 reports on the conducted empirical investigation. Section 8 concludes and points out future research directions.

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

*This work has been partially funded by the Generalitat de Catalunya under grant AGAUR 2009-SGR-1434, and the Ministerio de Economía y Competividad research projects AT CONSOLIDER CSD2007-0022, INGENIO 2010, and TASSAT (TIN2010-20967-C04-01/03) (funded by the Ministerio de Ciencia y Tecnología until 2011).

0195-623X/14 $31.00 © 2014 IEEE
DOI 10.1109/ISMVL.2014.14

belongs to a 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 natural number or infinity. If its weight is infinity, it is hard (we omit infinity weights for simplicity); otherwise it is soft. A (Boolean/Many-Valued) Weighted Partial MinSAT (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 if x_i evaluates to 1, and satisfies $\neg x_i$ if x_i evaluates to 0; and a many-valued assignment satisfies $x_j = i$ if x_j evaluates to i; otherwise satisfies $x_j \neq i$. A (Boolean/many-valued) assignment satisfies a weighted clause (c, w) if it satisfies at least one literal of c, and satisfies a multiset of clauses if it satisfies all its clauses.

The (Boolean/Many-Valued) Weighted Partial MinSAT (MaxSAT) problem, or WPMinSAT (WPMaxSAT), for an instance ϕ consists in finding a (Boolean/Many-Valued) assignment in which the sum of weights of the satisfied soft clauses is minimal (maximal), and all the hard clauses are satisfied.

Subproblem notation: Weighted MinSAT (MaxSAT), or WMinSAT (WMaxSAT), is WPMinSAT (WPMaxSAT) without hard clauses. Partial MinSAT (MaxSAT), or PMinSAT (PMaxSAT), is WPMinSAT (WPMaxSAT) when all the soft clauses have the same weight. MinSAT (MaxSAT) is PMinSAT (PMaxSAT) without hard clauses.

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 consists in finding 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 natural numbers. The cost of an assignment v is the sum of the weights of all constraints violated by v. An optimal solution is an assignment with minimal cost.

3 Related Work

The work on MinSAT for solving optimization problems may be divided into four categories: (I) Transformations between MinSAT and MaxSAT: Reductions from MinSAT to PMaxSAT were defined in [12], but these reductions do not generalize to WPMinSAT. This drawback was overcome with the definition of the natural encoding [9], which was improved in [16]. Reductions of WPMinSAT to Group MaxSAT were evaluated in [7]. (II) Encodings from WMaxCSP to WPMinSAT: Efficient encodings were defined in [6]. (III) Branch-and-bound solvers: The only existing WPMinSAT solver, MinSatz [13, 14], is based on MaxSatz [11], and implements upper bounds that exploit clique partition algorithms and MaxSAT technology. (IV) SAT-based solvers: There exist two WPMinSAT solvers of this class [3, 7]. They differ with SAT-based MaxSAT solvers in the way of relaxing soft clauses. A closely related problem has been recently analyzed in [8].

4 Mapping Many-Valued WPMinSAT to Boolean WPMinSAT

For mapping a Many-Valued WPMinSAT instance ϕ to an equivalent Boolean WPMinSAT instance ϕ', we define the following algorithm:

1. For each many-valued variable x occurring in ϕ, we identify its domain $d(x) = \{v_1, \ldots, v_n\}$, and define an associated set of Boolean variables $\{x_{v_1}, \ldots, x_{v_n}\}$. The intended meaning of Boolean literal x_{v_i} is that it is satisfied iff many-valued literal $x = v_i$ is satisfied.

2. For each variable x occurring in ϕ, we add hard clauses in ϕ' encoding that (i) at least one (ALO) of the Boolean variables in $\{x_{v_1}, \ldots, x_{v_n}\}$ must be true, and (ii) at most one (AMO) of the Boolean variables in $\{x_{v_1}, \ldots, x_{v_n}\}$ must be true. The ALO clause of $\{x_{v_1}, \ldots, x_{v_n}\}$ is $x_{v_1} \vee \cdots \vee x_{v_m}$, and the AMO clauses are the set of clauses $\{\neg x_{v_i} \vee \neg x_{v_j} | v_i, v_j \in d(x), i < j\}$. Adding the ALO and AMO clauses ensures that exactly one variable is true in each assignment.

3. For each many-valued hard clause $x_{j^1} = v_{i_1} \vee \cdots \vee x_{j^m} = v_{i_m}$ occurring in ϕ, we add in ϕ' the Boolean hard clause $x_{j^1{}_{v_{i_1}}} \vee \cdots \vee x_{j^m{}_{v_{i_m}}}$.

4. For each many-valued soft clause $(x_{l^1} = v_{i_1} \vee \cdots \vee x_{l^k} = v_{i_k}, w)$ occurring in ϕ, we add in ϕ' the Boolean soft clause $(x_{l^1{}_{v_{i_1}}} \vee \cdots \vee x_{l^k{}_{v_{i_k}}}, w)$.

This mapping is used by our Many-Valued WPMinSAT solver. Actually, it uses both the generated Boolean instance, and the information concerning the sets of Boolean

variables of step 1 that encode a many-valued variable. Such sets are decisive for exploiting the variable domain information. It is worth noticing that there are other alternatives to encode the ALO and AMO constraints (see e.g. [4, 5]).

5 The solver Mv-MinSatz

Mv-MinSatz is a Many-Valued MinSAT solver built on top of the Boolean solver MinSatz [13, 14]. Given an input Many-Valued MinSAT instance, Mv-MinSatz works as follows: (i) It generates both a Boolean MinSAT instance, and the sets of Boolean variables associated with each many-valued variable using the algorithm of Section 4; (ii) it finds an optimal Boolean MinSAT solution with the modified version of MinSatz explained below; and (iii) it returns an optimal solution to the input Many-Valued MinSAT instance, derived from the solution in the previous step.

MinSatz implements the branch-and-bound scheme, and the search space is formed by a tree representing all the possible truth assignments. We refer to [13, 14] for a detailed description, due to lack of space. Its main characteristic is that it implements upper bounds that exploit clique partition algorithms and MaxSAT technology, and its goal is to maximize the number of falsified clauses (recall that MaxSAT solvers work by minimizing that number). MinSatz 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 $score(x) \times score(\neg x) + score(x) + score(\neg x)$.

Our modified version of MinSatz differs in two aspect from MinSatz: (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 MinSatz only to the Boolean variables belonging to the candidate set. So, MinSatz 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. In [1, 2], a similar approach was applied in SAT.

6 Encodings from WMaxCSP to Many-Valued WPMinSAT

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 WPMinSAT, as was done for WPMaxSAT [5].

We first define the many-valued counterparts of the direct and minimal support encodings from WMaxCSP to WPMinSAT of [6], and then the new hybrid encodings. The Boolean versions of the encodings are obtained by applying the mapping of Section 4. All the mentioned encodings are correct: Solving a WMaxCSP instance is equivalent to solving the instance derived by any of our encodings. The minimal support encoding is valid for binary constraints, and the direct encoding is valid for non-binary constraints too. In the following, for each CSP variable X_i, we associate a many-valued variable x_i.

Definition 1. The **WPMinSAT direct encoding** of a WMaxCSP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ has a soft clause $(x_1 \neq i_1 \vee \cdots \vee x_m \neq i_m, w)$ for each good $(X_1 = i_1, \ldots, X_m = i_m)$ of each constraint of $\mathcal{C}$ with scope $\{X_1, \ldots, X_m\}$ and weight w.

Since WPMinSAT maximizes the sum of weights of falsified soft clauses, we force the violation of one soft clause for each satisfied constraint, because of that we negate the goods instead of the nogoods as is done in WPMaxSAT encodings of WMaxCSP [5]. Notice that if a constraint is violated, all the derived clauses are satisfied.

Example 1. Given a WMaxCSP instance where $\mathcal{X} = \{X, Y\}$, $d(X) = d(Y) = \{1, 2, 3\}$, and $\mathcal{C} = \{X = Y\}$ with weight 3. The Many-Valued WPMinSAT direct encoding contains the soft clauses $(x \neq 1 \vee y \neq 1, 3)$, $(x \neq 2 \vee y \neq 2, 3)$, and $(x \neq 3 \vee y \neq 3, 3)$. While WPMaxSAT needs a quadratic number of binary soft clauses in the domain size for encoding the equality constraint, WPMinSAT just needs a linear number of clauses.

In the support encoding from CSP to SAT, besides the ALO and AMO clauses, there are clauses that encode the support for a value instead of encoding conflicts. 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, while in the minimal support encoding [5], 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 much better in our tests. We need now to define the *negative support* for a value j of a CSP variable X across a binary constraint with scope $\{X, Y\}$ as the set of values of Y which forbid $X = j$.

Definition 2. The **WPMinSAT minimal support encoding** of a WMaxCSP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ contains, for each constraint with scope $\{X, Y\}$ and weight w, either the soft 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 negative support for i, or the soft 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 negative support for j.

In the WPMaxSAT encoding [5], the support clauses include the positive support instead of the negative one. We have now a violated clause per satisfied constraint. In the experiments, we select the variable that produces clauses of smaller size: 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 select the variable with higher sum of scores.

Example 2. Given a WMaxCSP instance where $\mathcal{X} = \{X, Y\}$, $d(X) = d(Y) = \{1, 2, 3\}$, and $\mathcal{C} = \{X \neq Y\}$ with weight 2. The Many-Valued WPMinSAT minimal support encoding contains the soft clauses $(x \neq 1 \vee y = 1, 2)$, $(x \neq 2 \vee y = 2, 2)$, and $(x \neq 3 \vee y = 3, 2)$. We could also define this encoding by adding the clauses with $(x = 1 \vee y \neq 1, 2)$, $(x = 2 \vee y \neq 2, 2)$, and $(x = 3 \vee y \neq 3, 2)$ if we add the support clauses for the domain values of Y instead of X. The length of the clauses of the WPMaxSAT encoding is linear in the domain size for the inequality constraint whereas in WPMinSAT all the clauses are binary.

Our experience with solving WPMinSAT/WPMaxSAT instances indicates that both the encoding size and the inference achieved with a particular encoding are decisive for performance. To produce encodings of smaller size whenever possible, we take a novel approach: we do not apply the same encoding to the whole WCSP. Instead of that, we choose the most convenient encoding for each individual constraint, and refer to such encodings as hybrid encodings.

We first define the binary hybrid encoding. Since it combines the direct and minimal support encodings, it is only valid for binary constraints. It relies on the following observation: When the number of goods ($\#goods$) is low, we get smaller encodings with the direct encoding while when the number of nogoods ($\#nogoods$) 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, 6], 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 there is a reduced $\#goods$: In practice, we use $k < 0.3$ in the definition below.

Definition 3. The **WPMinSAT binary hybrid encoding (2HE)** of a binary WMaxCSP instance contains, for each constraint with scope $\{X, Y\}$ and weight w, as soft clauses the minimal support encoding of the constraint if $\#goods > k \times \#nogoods$, where $k \in (0, 1]$; otherwise, it contains the direct encoding.

We define now 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 combine the WPMinSAT and WPMaxSAT encodings, depending on the size of the derived encoding, because WPMinSAT adds a clause for every good, and WPMaxSAT adds a clause for every nogood. Of course, to get a valid encoding the WPMaxSAT clauses should be transformed into equivalent WPMinSAT clauses without increasing a lot the encoding complexity. To do it, we use the natural flow network encoding (NFNE) [16][1]. For example, if we consider the constraint $X \neq Y$ with weight 5 and $d(X) = d(Y) = \{1, 2, 3, 4, 5, 6, 7, 8\}$, the WPMinSAT direct encoding has 56 clauses ($\{x \neq i \vee y \neq j \,|\, i, j \in d(X), i \neq j\}$) while the WPMaxSAT encoding has 8 clauses ($\{(x \neq i \vee y \neq i, 5) \,|\, i \in d(X)\}$). Now, we apply the NFNE to the WPMaxSAT instance and get the WPMinSAT encoding ($\{(x = i, 5), (x \neq i \vee y = i, 5) \,|\, i \in d(X)\}$), which contains 16 clauses, and is a valid WPMinSAT encoding of $X \neq Y$. On the other hand, if we consider the constraint $X = Y$, we would instead add the WPMinSAT direct encoding.

Definition 4. The **WPMinSAT n-ary hybrid encoding (nHE)** of a WMaxCSP instance contains, for each constraint with scope $\{X_1, \ldots, X_n\}$ and weight w, the encoding with a small number of clauses between the direct encoding, and the encoding resulting of applying the NFNE to the WPMaxSAT direct encoding of the constraint under consideration.

7 Empirical Investigation

We compared the performance of the direct encoding (*dir*), the minimal support encoding (*sup*), the network flow natural network encoding (*NFNE*), and the new hybrid encodings (*2HE, and nHE*) of random WMax-CSP instances with both the last available version of MinSatz, and our

[1] NFNE states that a Many-Valued WPMaxSAT instance can be transformed into Many-Valued WPMinSAT by replacing each WPMaxSAT clause $(l_1 = i_1 \vee l_2 = i_2 \vee \cdots \vee l_m = i_m, w)$ with $(l_1 \neq i_1, w), (l_1 = i_1 \vee l_2 \neq i_2, w), \ldots, (l_1 = i_1 \vee l_2 = i_2 \vee \cdots \vee l_m \neq i_m, w)$.

(n,d, c)	#	Mv-MinSatz (2HE)	Mv-MinSatz (sup)	MinSatz (2HE)	MinSatz (sup)
(25,5,230)	100	**100.38(100)**	101.74(100)	328.57(100)	376.01(100)
(25,5,240)	100	**153.03(100)**	153.33(100)	479.23(99)	559.92(99)
(25,5,250)	100	216.89(100)	**215.23(100)**	640.63(96)	700.95(94)
(25,5,260)	100	313.07(100)	**307.88(100)**	857.06(91)	936.52(88)
(25,5,270)	100	**447.39(100)**	453.17(100)	1027.93(73)	1040.89(63)
(25,5, 280)	100	**592.09(98)**	637.36(98)	1009.07(48)	1054.19(42)
Total	600	**598**	598	507	486

Table 1. Results for binary WMaxCSPs solved with MinSatz and Mv-MinSatz comparing encoding 2HE and the minimal support encoding. 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-MinSatz (nHE)	MinSatz (nHE)	Mv-MinSatz (dir)	MinSatz (dir)
(22,5,150)	100	**88.97(100)**	137.25(100)	351.47(100)	401.47(100)
(22,5,160)	100	**125.12(100)**	207.72(100)	565.61(99)	630.57(96)
(22,5,170)	100	**235.73(100)**	391.38(100)	864.90(86)	913.70(77)
(22,5,180)	100	**391.45(99)**	607.57(99)	1182.61(54)	1201.39(41)
(22,5,190)	100	**623.69(100)**	869.79(93)	1292.21(20)	1319.35(12)
(22,5,200)	100	**840.61(93)**	1096.21(82)	1450.77(8)	1654.88(2)
Total	600	**592**	574	367	328

Table 2. Results for binary WMaxCSPs solved with MinSatz and Mv-MinSatz comparing encoding nHE and the direct encoding. 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.

solver Mv-MinSatz. We solved sets of 100 binary and ternary WMaxCSP instances generated using the so-called model B [15]. 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 no-goods ranging from 1 to $d^k - 1$. Experiments were performed on a cluster with Intel Xeon CPU E5-2620 @ 2GHz processors with 4GB of RAM.

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 with the direct encoding, encoding nHE, and encoding NFNE 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 nHE with the direct encoding. Encoding NFNE was not competitive with any of the previous encodings.

The results obtained allow us to draw some conclusions:

- Many-Valued WPMinSAT outperforms Boolean WPMinSAT thanks to the exploitation of the domain information. For example, the number of additional solved instances for binary WMaxCSP is 91 and 18 instances (Tables 1 and 2), and for ternary WMaxCSPs is 50 instances (Table 3). It is also worth noticing that the cost of our approach to building many-valued solvers from Boolean solvers is very low compared with the cost of building a many-valued solver from scratch.
- The hybrid encoding nHE has a very good performance profile. It solves more than 200 (100) additional instances in Table 2 (Table 3). This is specially important for dealing with non-binary constraints because support encodings are only valid for binary constraints, and the only available option for non-binary

(n,d, c)	#	Mv-MinSatz (nHE)	MinSatz (nHE)	Mv-MinSatz (dir)	MinSatz (dir)
(18,5,100)	100	**756.55(92)**	859.77(88)	1062.93(74)	1067.02(68)
(18,5,110)	100	**1043.15(80)**	1157.68(66)	1317.40(51)	1345.19(43)
(18,5,120)	100	**1219.65(49)**	1404.61(37)	1474.57(14)	1585.15(11)
(18,5,130)	100	**1367.58(24)**	1375.42(12)	1467.86(4)	1498.17(3)
(18,5,140)	100	**1561.98(9)**	1578.97(4)	0.00(0)	0.00(0)
(18,5,150)	100	**1551.30(3)**	0.00(0)	0.00(0)	0.00(0)
Total	600	**257**	207	143	125

Table 3. Results for ternary WMaxCSPs solved with MinSatz and Mv-MinSatz comparing encoding nHE and the direct encoding. 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.

constraints is the direct encoding. Moreover, this encoding can be adapted to WPMaxSAT: We should apply the NFNE to the WPMinSAT clauses instead of the WPMaxSAT clauses.

- The hybrid encoding 2HE does not produce so significant speed-ups as encoding nHE, but it allows to solve 21 additional instances in the Boolean case.

8 Concluding Remarks

We have studied for the first time Many-Valued WPMinSAT, described how to easily build a many-valued solver from a Boolean WPMinSAT solver, and provided empirical evidence of the superiority of Many-Valued WPMinSAT over Boolean WPMinSAT on the tested instances. We have also defined original hybrid encodings from WMaxCSP to WPMinSAT. In particular, we would like to highlight the good results achieved with encoding nHE, as well as the fact that this encoding can be easily adapted to WPMaxSAT.

Since Boolean formalisms are a de facto standard in the SAT community, in order to apply the contributions of this paper to existing Boolean encodings, we propose to investigate methods for automatically detecting finite-domain variables that are hidden in Boolean encodings.

References

[1] 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.

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

[3] C. Ansótegui, C. M. Li, F. Manyà, and Z. Zhu. A SAT-based approach to MinSAT. In *Proceedings of CCIA-2012*, pages 185–189, 2012.

[4] J. Argelich, A. Cabiscol, I. Lynce, and F. Manyà. New insights into encodings from MaxCSP into partial MaxSAT. In *Proceedings of ISMVL-2010*, 2010.

[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. MinSAT versus MaxSAT for optimization problems. In *Proceedings of CP 2013*, pages 133–142, 2013.

[7] F. Heras, A. Morgado, J. Planes, and J. Marques-Silva. Iterative SAT solving for minimum satisfiability. In *Proceedings of ICTAI 2012*, pages 922–927, 2012.

[8] A. Ignatiev, A. Morgado, J. Planes, and J. Marques-Silva. Maximal falsifiability - definitions, algorithms, and applications. In *Proceedings of LPAR-2013*, pages 439–456. Springer, LNAI 8312, 2013.

[9] A. Kügel. Natural Max-SAT encoding of Min-SAT. In *Proceedings of the Learning and Intelligent Optimization Conference, LION 6*, 2012.

[10] 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.

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

[12] C. M. Li, F. Manyà, Z. Quan, and Z. Zhu. Exact MinSAT solving. In *Proceedings of SAT-2010*, pages 363–368. Springer LNCS 6175, 2010.

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

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

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

[16] Z. Zhu, C. M. Li, F. Manyà, and J. Argelich. A new encoding from MinSAT into MaxSAT. In *Proceedings of CP 2012*, pages 455–463, 2012.