New Insights into Encodings from MaxCSP into Partial MaxSAT*

Josep Argelich
DIEI, UdL
Lleida, Spain

Alba Cabiscol
DIEI, UdL
Lleida, Spain

Inês Lynce
IST, INESC-ID
Lisboa, Portugal

Felip Manyà
IIIA, CSIC
Bellaterra, Spain

Abstract

We analyze the existing encodings from MaxCSP into Partial MaxSAT, and report on a number of new insights that we have gained from our analysis, which can be summarized as follows: (i) the at-most-one (AMO) condition can be omitted in direct encodings from MaxCSP into Partial MaxSAT, and auxiliary variables are not needed; (ii) the sequential encoding of the cardinality constraint is, in fact, a reformulation of a regular encoding; (iii) the AllDifferent constraint based on regular literals may be simplified; (iv) if we represent, in support encodings, the supporting values of a variable using intervals, then we can derive a genuine regular support encoding without exponential blowup; and (v) the Equal constraint admits a concise representation with regular signs.

1 Introduction

Solving combinatorial decision and optimization problems via their reduction to Satisifability Problems (e.g., SAT and MaxSAT) depends on both the solver and the encoding. In previous work [2, 3, 4, 5], we presented a number of original encodings —that rely on the well-known direct and support encodings from CSP into SAT [9, 15]— that map Max-CSP instances into Partial Max-SAT instances. Besides, for the new direct and support encodings, we defined three different ways of modelling the at-least-one (ALO) and at-most-one (AMO) conditions, which were called *standard, regular, and sequential*.

In this paper we analyze in more detail than ever before the existing encodings from MaxCSP into Partial MaxSAT, and report on a number of new insights that we have gained from our analysis, which can be summarized as follows: (i) the at-most-one (AMO) condition can be omitted in direct encodings from MaxCSP into Partial MaxSAT, and auxiliary variables are not needed; (ii) the sequential encoding of the cardinality constraint is, in fact, a reformulation of a regular encoding; (iii) the AllDifferent constraint based on regular literals may be simplified; (iv) if we represent, in support encodings, the supporting values of a variable using intervals, then we can derive a genuine regular support encoding without exponential blowup; and (v) the Equal constraint admits a concise representation with regular signs.

The structure of the paper is as follows. Section 2 contains preliminary definitions about Max-SAT and Max-CSP. Section 3 surveys the existing encodings from Max-CSP into Partial Max-SAT, and Section 4 describes the new insights we have gained after analyzing the encodings defined in Section 3.

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 ϕ. In the 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 as it is better suited for representing and solving NP-hard problems. A Partial Max-SAT instance is

*This research was funded by the Generalitat de Catalunya under grant 2009-SGR-1434, the *Ministerio de Ciencia e Innovación* research projects CONSOLIDER CSD2007-0022, INGENIO 2010, TIN2006-15662-C02-02, TIN2007-68005-C04-04, and TIN2009-14704-C03-01, FCT research project SHIPs (PTDC/EIA/64164/2006), and European project Mancoosi (FP7-ICT-214898).

0195-623X/10 $26.00 © 2010 IEEE
DOI 10.1109/ISMVL.2010.17

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 maximizes the number of satisfied 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 $d(X_{i_1}) \times \cdots \times d(X_{i_k})$. The tuples belonging to R_i are called goods, and the rest of tuples are called nogoods.

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 $\langle v(X_{i_1}), \ldots, v(X_{i_k}) \rangle \in R_i$.

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.

We can convert an arbitrary CSP to an equivalent binary CSP [8]. 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 Encodings from Max-CSP into Partial Max-SAT

3.1 Standard Encodings

We associate a Boolean variable x_i with each value i that the CSP variable X can take. Assuming that X has a domain $d(X)$ of size m, the *ALO* clause of X is $x_1 \vee \cdots \vee x_m$, and ensures that the CSP variable X is given a value. The *AMO* clauses of X are the set of clauses $\{\overline{x}_i \vee \overline{x}_j | i, j \in d(X), i < j\}$, and ensure that the CSP variable X takes no more than one value.

Definition 3. The *direct encoding* (`dir`) 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 above ALO and AMO clauses for every CSP variable in $\mathcal{X}$, and 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 $\{X, Y\}$.

In the *support encoding* from CSP into 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 j of a CSP variable X across a binary constraint with scope $\{X, Y\}$ is the set of values of Y which allow $X = j$. If $v_1, v_2, \ldots, v_k$ are the supporting values of variable Y for $X = j$, we add the clause $\overline{x}_j \vee y_{v_1} \vee y_{v_2} \vee \cdots \vee y_{v_k}$ (called *support* clause). There is one support clause for each pair of variables X, Y involved in a constraint, and for each value in the domain of X. In the standard support encoding, a clause in each direction is used: one for the pair X, Y and one for Y, X [15].

In [2], we defined the *minimal support encoding*: it 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.

Definition 4. 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 ALO and AMO clauses for every CSP variable in $\mathcal{X}$, and as soft clauses the support clauses of the minimal support encoding from CSP into SAT.

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 ALO and AMO 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.

Example 1. The Partial Max-SAT direct encoding for the Max-CSP problem of the CSP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle = \langle \{X, Y\}, \{d(X) = \{1, 2, 3\}, d(Y) = \{1, 2, 3\}\}, \{X \leq Y\} \rangle$ is as follows:

ALO	$[x_1 \vee x_2 \vee x_3]$	$[y_1 \vee y_2 \vee y_3]$	
AMO	$[\overline{x}_1 \vee \overline{x}_2]$	$[\overline{x}_1 \vee \overline{x}_3]$	$[\overline{x}_2 \vee \overline{x}_3]$
	$[\overline{y}_1 \vee \overline{y}_2]$	$[\overline{y}_1 \vee \overline{y}_3]$	$[\overline{y}_2 \vee \overline{y}_3]$
conflict clauses	$(\overline{x}_2 \vee \overline{y}_1)$	$(\overline{x}_3 \vee \overline{y}_1)$	$(\overline{x}_3 \vee \overline{y}_2)$

We get the minimal support encoding if we replace the conflict clauses with $(\overline{x}_2 \vee y_2 \vee y_3)$, $(\overline{x}_3 \vee y_3)$, and get the support

encoding if we replace the conflict clauses with $(\overline{x}_2 \vee y_2 \vee y_3 \vee c_1), (\overline{y}_1 \vee x_1 \vee \overline{c}_1), (\overline{x}_3 \vee y_3 \vee c_1), (\overline{y}_2 \vee x_1 \vee x_2 \vee \overline{c}_1)$.

In the experiments we use the support encoding (supxy), and the minimal support encoding (supc), which is 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.

3.2 Regular Encodings

The regular encodings differ in the fact that they encode the ALO and AMO conditions using a regular signed encoding [1]. To this end, for every CSP variable X, we associate a Boolean variable x_i with each value i that can be assigned to the CSP variable X in such a way that x_i is true if $X = i$. Moreover, we associate a Boolean variable $x_i^{\geq}$ with each value i of the domain of X such that $x_i^{\geq}$ is true if $X \geq i$. Then, the *regular encoding of the ALO and AMO conditions* for a variable X with $d(X) = \{1, \ldots, n\}$ is formed by the following clauses [1]:

$$\begin{array}{ll} x_n^{\geq} \rightarrow x_{n-1}^{\geq} & x_1 \leftrightarrow \overline{x}_2^{\geq} \\ x_{n-1}^{\geq} \rightarrow x_{n-2}^{\geq} & x_2 \leftrightarrow x_2^{\geq} \wedge \overline{x}_3^{\geq} \\ \cdots\cdots\cdots\cdots & \cdots\cdots\cdots\cdots \\ x_3^{\geq} \rightarrow x_2^{\geq} & x_i \leftrightarrow x_i^{\geq} \wedge \overline{x}_{i+1}^{\geq} \\ x_2^{\geq} \rightarrow x_1^{\geq} & \cdots\cdots\cdots\cdots \\ & x_{n-1} \leftrightarrow x_{n-1}^{\geq} \wedge \overline{x}_n^{\geq} \\ & x_n \leftrightarrow x_n^{\geq} \end{array} \quad (1)$$

The clauses on the left encode the relationship among the different regular literals of a variable while the clauses on the right link the variables of the form x_i with the variables of the form $x_i^{\geq}$. Notice that $x_2^{\geq} \rightarrow x_1^{\geq}$ can be omitted.

Definition 5. The *regular direct, support, and minimal support encodings* are, respectively, the standard direct, support, and minimal support encodings from Max-CSP into Partial Max-SAT but using the regular encoding of the ALO and AMO conditions.

3.3 Sequential Encodings

The sequential encodings model the ALO condition as in the standard encoding, and the AMO condition using the following SAT encoding, based on sequential counters, of the cardinality constraint $\leq 1(x_1, \ldots, x_n)$ [14]:

$$(\overline{x}_1 \vee s_1) \wedge (\overline{x}_n \vee \overline{s}_{n-1}) \wedge \bigwedge_{1<i<n} \big((\overline{x}_i \vee s_i) \wedge (\overline{s}_{i-1} \vee s_i) \wedge (\overline{x}_i \vee \overline{s}_{i-1})\big),$$

where s_i, $1 \leq i \leq n-1$, are auxiliary variables. We refer to such an encoding as the sequential encoding of the AMO condition.

Definition 6. The *sequential direct, support, and minimal support encodings* are, respectively, the standard direct, support, and minimal support encodings from Max-CSP into Partial Max-SAT but using the sequential encoding of the AMO condition.

Example 2. The *sequential minimal support encoding* for the Max-CSP problem of the CSP instance from Example 1 is formed by the following clauses:

hard clauses	$[x_1 \vee x_2 \vee x_3]$	$[y_1 \vee y_2 \vee y_3]$	
	$[\overline{x}_1 \vee s_1^x]$	$[\overline{x}_3 \vee \overline{s}_2^x]$	
	$[\overline{x}_2 \vee s_2^x]$	$[\overline{s}_1^x \vee s_2^x]$	$[\overline{x}_2 \vee \overline{s}_1^x]$
	$[\overline{y}_1 \vee s_1^y]$	$[\overline{y}_3 \vee \overline{s}_2^y]$	
	$[\overline{y}_2 \vee s_2^y]$	$[\overline{s}_1^y \vee s_2^y]$	$[\overline{y}_2 \vee \overline{s}_1^y]$
support clauses	$(\overline{x}_2 \vee y_2 \vee y_3)$	$(\overline{x}_3 \vee y_3)$	

We get the *sequential support encoding* if we replace the previous support clauses with $(\overline{x}_2 \vee y_2 \vee y_3 \vee c_1), (\overline{y}_1 \vee x_1 \vee \overline{c}_1), (\overline{x}_3 \vee y_3 \vee c_1), (\overline{y}_2 \vee x_1 \vee x_2 \vee \overline{c}_1)$. Finally, we get the *sequential direct encoding* if we replace the previous support clauses with $(\overline{x}_2 \vee \overline{y}_1), (\overline{x}_3 \vee \overline{y}_1), (\overline{x}_3 \vee \overline{y}_2)$.

In the sequential encodings, the number of clauses for modelling the ALO and AMO conditions for a CSP variable X with domain $d(X)$ is on $\mathcal{O}(d(X))$.

4 New Insights

4.1 Insight 1: AMO clauses can be omitted in direct MaxSAT encodings and auxiliary variables are not needed

In encodings from CSP into SAT, CSP variables are often encoded into SAT using the ALO and AMO clauses. For the support encoding, ALO and AMO clauses are compulsory. Actually, if AMO clauses are omitted, then the encoding becomes incorrect as we show in the following example.

Example 3. Given the CSP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle = \langle \{X, Y\}, \{d(X) = \{1,2,3\}, d(Y) = \{1,2,3\}\}, \{X = Y\} \rangle$, the SAT support encoding without AMO clauses is as follows:

ALO	$x_1 \vee x_2 \vee x_3$	$y_1 \vee y_2 \vee y_3$	
support clauses	$\overline{x}_1 \vee y_1$	$\overline{x}_2 \vee y_2$	$\overline{x}_3 \vee y_3$
	$x_1 \vee \overline{y}_1$	$x_2 \vee \overline{y}_2$	$x_3 \vee \overline{y}_3$

Assume that all the variables are set to true: $x_1 = x_2 = x_3 = y_1 = y_2 = y_3 = true$. In this case, the encoding should be unsatisfiable because there are combinations such as $x_1 = y_3 = true$ which are not permitted. However, the encoding becomes satisfiable when all the variables are set to true because each clause contains at least one positive literal. This counterexample proves the incorrectness of

the encoding. The minimal support encoding has the same drawback.

As a consequence of the previous result, we conclude that ALO and AMO clauses are also compulsory in support MaxSAT encodings.

As pointed out in [13], AMO clauses can be omitted in the direct encoding from CSP into SAT in such a way that the CSP is satisfiable iff the resulting SAT encoding is satisfiable.

We claim that AMO clauses can also be omitted in MaxSAT direct encodings, and there is no need to introduce auxiliary variables. This is so because optimal solutions of a MaxSAT direct encoding of a MaxCSP instance violates exactly one clause per violated constraint, and we can therefore establish a mapping between MaxSAT optimal solutions and MaxCSP optimal solutions. If a MaxSAT interpretation violates more than one clause per constraint, then there is, for each violated constraint, at least one CSP variable of the constraint scope with more than one value set to true in its MaxSAT encoding; otherwise, there is exactly one violated clause corresponding to one nogood. By setting to false the values of that CSP variables that produce the violation of more than one clause, we get a new MaxSAT interpretation that violates just one clause per violated constraint. Observe that the new interpretation is correct because there is an ALO hard constraint for every CSP variable, and the number of violated soft clauses has decreased because now there is exactly one violated clause in every violated constraints and, moreover, there is no soft clause that becomes unsatisfied due to the new interpretation because all the soft clauses of a MaxSAT direct encoding are negative and we set some variables to false but we do not set any variable to true. Therefore, an interpretation violating more than one clause of a same constraint cannot be optimal.

4.2 Insight 2: The sequential encoding of the cardinality constraint $\leq 1(x_1, \ldots, x_n)$ is a regular encoding

The sequential encoding of the cardinality constraint $\leq 1(x_1, \ldots, x_n)$ [14], which models the AMO condition, is defined in Section 3.3. If we replace simultaneously x_1 with x_n, x_2 with x_{n-1}, ..., x_n with x_1, and s_1 with $x_n^{\geq}$, s_2 with $x_{n-1}^{\geq}$, ..., s_{n-1} with $x_2^{\geq}$, we get the following encoding of $\leq 1(x_1, \ldots, x_n)$:

$$(\overline{x}_n \vee x_n^{\geq}) \wedge (\overline{x}_1 \vee \overline{x}_2^{\geq}) \wedge \bigwedge_{1<i<n} \left((\overline{x}_i \vee x_i^{\geq}) \wedge (\overline{x}_{i+1}^{\geq} \vee x_i^{\geq}) \wedge (\overline{x}_i \vee \overline{x}_{i+1}^{\geq}) \right),$$

which is the regular encoding of the ALO and AMO conditions defined in Section 3.2 without the ternary clauses. Therefore, in a sense, the popular sequential encoding of $\leq 1(x_1, \ldots, x_n)$ reinvented the regular encoding (defined before). Moreover, this proves that the encoding resulting from eliminating the ternary clauses in the regular encoding is a valid encoding of the AMO condition.

Let us prove that the ternary clauses of the regular encoding provide an alternative encoding of the ALO condition. Given the clauses

$$\begin{array}{l} \overline{x}_2^{\geq} \rightarrow x_1 \\ x_2^{\geq} \wedge \overline{x}_3^{\geq} \rightarrow x_2 \\ \cdots\cdots\cdots\cdots \\ x_i^{\geq} \wedge \overline{x}_{i+1}^{\geq} \rightarrow x_i \\ \cdots\cdots\cdots\cdots \\ x_{n-1}^{\geq} \wedge \overline{x}_n^{\geq} \rightarrow x_{n-1} \\ x_n^{\geq} \rightarrow x_n, \end{array} \tag{2}$$

we have that if $x_1 = x_2 = \cdots = x_n = false$, we get a contradiction by applying unit propagation. Otherwise, if any variable x_i is true, then we can build a satisfying assignment by setting to true $x_j^{\geq}$ for $j \leq i$ and setting to false $x_k^{\geq}$ for $k > i$. So, these ternary clauses encode the ALO condition.

Another consequence of this insight is that the sequential encoding is the encoding resulting by replacing, in the regular encoding, the regular ALO condition by the standard ALO condition.

4.3 Insight 3: AllDifferent Constraint

The usual approach to encode the CSP constraint AllDifferent into SAT is the pairwise encoding, which is based on decomposing it into pairwise binary constraints $X \neq Y$, and then encode these constraints using the direct encoding.

Gent and Nightingale [10] suggested to use regular literals (called ladder variables in [10]) for obtaining efficient SAT encodings of the global constraint AllDifferent, and showed that their encoding scales better than the pairwise encoding. Given a set of variables and a set of values, they encode the ALO and AMO conditions as in the regular encoding (cf. Section 3.2), and then use an exactly one (ALO+AMO) condition to encode that each variable takes exactly one value, and an AMO condition to encode that each value can be used at most one.

For encoding the AMO condition with regular literals, they take the regular encoding of the ALO+AMO condition defined in Section 3.2 and add an additional variable. This additional variable indicates that no variables are set to true. Taking into account our Insight 2, we propose a simpler encoding of the AllDifferent constraint: we replace the AMO encoding containing an additional variable with the AMO encoding we have defined in Section 3.3. The new encoding does not need to use additional variables and has fewer clauses.

4.4 Insight 4: Supports by intervals avoid exponential blowup

The support clauses of the support and minimal support encodings are as follows: $\overline{x}_j \vee y_{v_1} \vee y_{v_2} \vee \cdots \vee y_{v_k}$. If we want to represent $y_{v_1} \vee y_{v_2} \vee \cdots \vee y_{v_k}$ using only regular signs, there may be an exponential blowup due to the application of distributivity to $(y^{\geq}_{v_1} \wedge \overline{y}^{\geq}_{v_1+1}) \vee (y^{\geq}_{v_2} \wedge \overline{y}^{\geq}_{v_2+1}) \vee \cdots \vee (y^{\geq}_{v_k} \wedge \overline{y}^{\geq}_{v_k+1})$. Note that $\overline{x}_j$ may be translated into $\overline{x}^{\geq}_j \vee x^{\geq}_{j+1}$ because it is a negative literal.

A partial solution for avoiding this exponential blowup was given in [1], using the so-called half regular encoding: negative literals are represented with regular signs, and positive literals are represented in the standard way (also known as monosigned). This technique mitigates the exponential blowup when most of the literals have negative polarity. However, in the support encodings, it is not useful because most of the literals are positive.

We propose a new regular encoding, based on intervals, in which $y_{v_1} \vee y_{v_2} \vee \cdots \vee y_{v_k}$ are entirely represented by regular signs. In our encoding, for each support clause, we need less than $d(Y)$ clauses having at most three literals.

Given a support clause $\overline{x}_j \vee y_{v_1} \vee y_{v_2} \vee \cdots \vee y_{v_k}$, the idea behind our new encoding is to represent by intervals the supporting values of variable Y for $X = j$, and then encode that the supporting variable Y has to take a value inside one of the allowed intervals. We illustrate this idea with an example: Assume that the domain of Y is $\{1, 2, \ldots, 10\}$, and that the support clause is $\overline{x}_1 \vee y_2 \vee y_3 \vee y_6 \vee y_8 \vee y_9$. Then, Y has to take a value in one of the following intervals: $[2, 3]$, $[6, 6]$, and $[8, 9]$. The interval-based regular encoding for this clause is as follows:

$$\begin{array}{l} \overline{x}^{\geq}_2 \rightarrow y^{\geq}_2 \\ \overline{x}^{\geq}_2 \wedge y^{\geq}_4 \rightarrow y^{\geq}_6 \\ \overline{x}^{\geq}_2 \wedge y^{\geq}_7 \rightarrow y^{\geq}_8 \\ \overline{x}^{\geq}_2 \rightarrow \overline{y}^{\geq}_{10} \end{array} \tag{3}$$

Another advantage of the interval-based encoding is that there is exactly one violated clause for each direction of each violated soft constraint. So, the new minimal support encoding, which we will call *minimal interval-based regular encoding*, does not need to introduce auxiliary variables; and the new support encoding, which we will call *interval-based regular encoding*, only needs an auxiliary variable per soft constraint.

Example 4. A *minimal interval-based regular encoding* for the Max-CSP problem of the CSP instance from Example 1 is formed by the following clauses:

hard clauses	$[\overline{x}^{\geq}_3 \vee x^{\geq}_2]$	$[\overline{y}^{\geq}_3 \vee y^{\geq}_2]$
support clauses	$(\overline{x}^{\geq}_2 \vee x^{\geq}_3 \vee y^{\geq}_2)$	
	$(\overline{x}^{\geq}_3 \vee y^{\geq}_3)$	

We get the *interval-based regular encoding* if we replace the previous support clauses with $(\overline{x}^{\geq}_2 \vee x^{\geq}_3 \vee y^{\geq}_2 \vee c_1)$, $(\overline{x}^{\geq}_3 \vee y^{\geq}_3 \vee c_1)(y^{\geq}_2 \vee \overline{x}^{\geq}_3 \vee \overline{c}_1)$ and $(\overline{y}^{\geq}_2 \vee y^{\geq}_3 \vee \overline{x}^{\geq}_2 \vee \overline{c}_1)$. Observe that when we only use regular signs, the number of hard clauses is reduced. We just need the left clauses of Equation 1 [1].

We conducted an empirical investigation to assess the performance of the new encodings. The experiments were performed on a cluster with 160 2 GHz AMD Opteron 248 Processors with 1 GB of memory.

In the experiments we use a variant of the minimal interval-based regular encoding, which is the minimal interval-based regular 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.

To see the practical impact of the new encodings we have solved, with MSUnCore [12], planning and warehouse location instances from the collection of benchmarks of the Soft Constraint Satisfaction Problem site[1], and we have solved, with WMaxSatz [11, 6], random binary CSPs[2]. These solvers were selected for being the most suitable for the different kinds of instances[3]. In our experiments we compared the support encoding (supxy), the minimal support encoding (supc), the interval-based regular encoding (supxy-regular), and the minimal interval-based regular encoding (supc-regular). The cutoff time of the structured instances was set to 30 minutes. We used the cutoff time of the MaxSAT Evaluation [7].

The experimental results obtained are shown in Table 1, Table 2 and Figure 1. Table 1 and Table 2 provide empirical evidence that the support encoding (supxy) is slightly superior to the interval-based regular encoding (supxy-regular) (it solves one additional planning instance and the same number of warehouse location instances), but the minimal interval-based regular encoding (supc-regular) is clearly superior to the minimal support encoding (supc) (it solves 6 additional planning instances and 22 additional warehouse location instances).

Figure 1 provides empirical evidence of the superiority of the (minimal) interval-based regular encoding over the (minimal) support encoding on random binary CSPs. It is worth noticing that the minimal interval-based regular encoding is up to three orders of magnitude faster than the

[1] http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/Benchmarks

[2] They were obtained with a generator of uniform random binary CSPs —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).

[3] We did not use instances from the MaxSAT Evaluation [7] because they are in CNF format, and we need to know which clauses belong to each constraint in order to derive the encodings we would like to test.

Instance set	#	supc-regular	supc	supxy-regular	supxy
bwt	11	3.70(10)	**1.25(11)**	46.60(10)	**3.85(11)**
depot	4	0.22(3)	**0.03(3)**	0.17(3)	**0.18(4)**
driverlog	20	0.07(2)	**0.03(2)**	0.15(2)	**0.19(3)**
driverlogs	2	**2.17(1)**	0.00(0)	0.00(0)	**2.61(1)**
logistics	4	0.54(4)	**0.13(4)**	1.58(4)	**0.83(4)**
mprime	12	**3.84(11)**	1.59(10)	16.90(11)	**4.03(11)**
rovers	4	2.57(3)	**0.40(3)**	0.00(0)	0.00(0)
satellite	7	**1.46(4)**	0.00(0)	0.00(0)	0.00(0)
zenotravel	8	**8.84(4)**	0.15(3)	**171.45(6)**	0.60(3)
Solved instances	72	**42**	36	36	**37**

Table 1. Planning benchmarks with MSUnCore. Mean time in seconds.

Instance set	#	supc-regular	supc	supxy-regular	supxy
cap	37	**24.25(34)**	1.41(13)	30.48(37)	**19.61(37)**
warehouse	2	**0.39(2)**	0.07(1)	0.43(2)	**0.26(2)**
Solved instances	39	**36**	14	**39**	**39**

Table 2. Warehouse location benchmarks with MSUnCore. Mean time in seconds.

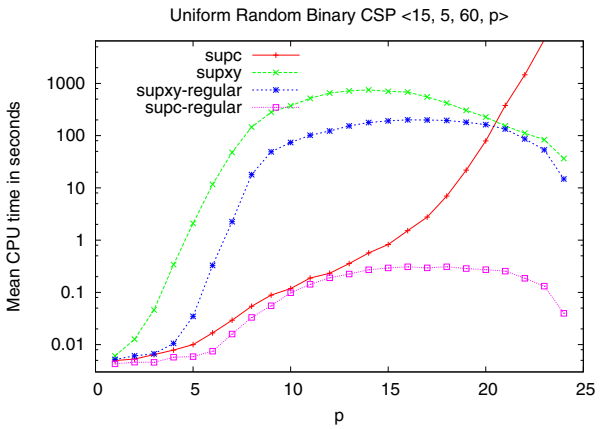

Figure 1. Random binary CSP benchmarks with WMaxSatz.

minimal support encoding on the tested instances.

4.5 Insight 5: Equal Constraint

The usual approach to encode the CSP constraint $X = Y$ is to include the ALO and AMO conditions for the CSP variables X and Y, and then add two clauses for every value i of the domain: $\overline{x}_i \vee y_i$ and $x_i \vee \overline{y}_i$ (which can also be represented by $x_i \leftrightarrow y_i$), where we assume that the domains of X and Y are equal ($d(X) = d(Y) = N$).

We propose a new encoding that is entirely based on regular signs. First of all, we notice that the naive encoding using regular signs would be to replace x_i with $x_i^{\geq} \wedge \overline{x}_{i+1}^{\geq}$ and y_i with $y_i^{\geq} \wedge \overline{y}_{i+1}^{\geq}$ in the previous encoding, and replace the AMO and ALO conditions with $\{\overline{x}_{i+1}^{\geq} \rightarrow x_i^{\geq} | 1 < i < |N|\} \cup \{\overline{y}_{i+1}^{\geq} \rightarrow y_i^{\geq} | 1 < i < |N|\}$. However, we propose to use a simpler and more efficient regular encoding of $x_i \leftrightarrow y_i$, which consists of the following set of clauses $\{x_i^{\geq} \leftrightarrow y_i^{\geq} | 1 < i \leq |N|\}$. Notice that the naive encoding produces 4 ternary clauses for each value of the domain, while our new encoding produces just 2 binary clauses. Also notice that it is not straightforward to see that $x_i \leftrightarrow y_i$ and $x_i^{\geq} \leftrightarrow y_i^{\geq}$ are equivalent.

In order to prove the correctness of the encoding we distinguish three cases: (i) $X = Y = i$: In this case we can add the unit clauses $x_i^{\geq}$, $\overline{x}_{i+1}^{\geq}$, $y_i^{\geq}$ and $\overline{x}_{i+1}^{\geq}$ (which encode $X = Y = i$) , and then we can derive the empty formula by applying unit propagation. Therefore, the encoding is satisfiable. (ii) $X > Y, X = i, Y = j$: In this case, the encoding is unsatisfiable because the clause $\overline{x}_i^{\geq} \vee y_i^{\geq}$ cannot be satisfied; and (iii) $Y > X, X = i, Y = j$: In this case, the encoding is unsatisfiable because the clause $x_j^{\geq} \vee \overline{y}_j^{\geq}$ cannot be satisfied.

References

[1] C. Ansótegui and F. Manyà. Mapping problems with finite-domain variables into problems with boolean variables. In *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing (Revised Se-*

lected Papers), SAT-2004, Vancouver, Canada, pages 1–15. Springer LNCS 3542, 2004.

[2] J. Argelich, A. Cabiscol, I. Lynce, and F. Manyà. Encoding Max-CSP into partial Max-SAT. In *Proceedings, 38th International Symposium on Multiple-Valued Logics (ISMVL), Texas/TX, USA*. IEEE CS Press, 2008.

[3] J. Argelich, A. Cabiscol, I. Lynce, and F. Manyà. Modelling Max-CSP as partial Max-SAT. In *Proceedings of the 11th International Conference on Theory and Applications of Satisfiability Testing, SAT-2008, Guangzhou, China*, pages 1–14. Springer LNCS 4996, 2008.

[4] J. Argelich, A. Cabiscol, I. Lynce, and F. Manyà. Regular encodings from Max-CSP into partial Max-SAT. In *Proceedings, 39th International Symposium on Multiple-Valued Logics (ISMVL), Okinawa, Japan*. IEEE CS Press, 2009.

[5] J. Argelich, A. Cabiscol, I. Lynce, and F. Manyà. Sequential encodings from Max-CSP into partial Max-SAT. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT-2009, Swansea, UK*, pages 161–166. Springer LNCS 5584, 2009.

[6] J. Argelich, C. M. Li, and F. Manyà. An improved exact solver for partial Max-SAT. In *Proceedings of the International Conference on Nonconvex Programming: Local and Global Approaches, NCP-2007, Rouen, France*, pages 230–231, 2007.

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

[8] F. Bacchus and P. van Beek. On the conversion between non-binary and binary constraint satisfaction problems. In *AAAI/IAAI*, pages 310–318, 1998.

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

[10] I. P. Gent and P. Nightingale. A new encoding of AllDifferent into SAT. In *Proceedings of the 3rd International Workshop on Modelling and Reformulating Constraint Satisfaction Problems, CP 2004 Workshop, Toronto, Canada*, pages 95–110, 2004.

[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] J. Marques-Silva and J. Planes. Algorithms for maximum satisfiability using unsatisfiable cores. In *Proceedings of Design, Automation and Test in Europe (DATE 2008)*, pages 408–413, Munich, Germany, 2008.

[13] S. D. Prestwich. CNF encodings. In A. Biere, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, pages 75–97. IOS Press, 2009.

[14] C. Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming, CP-2005, Sitges, Spain*, pages 827–831. Springer LNCS 3709, 2005.

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