A Declarative Approach to Robust Weighted Max-SAT *

Miquel Bofill
Dept. Informàtica i Matemàtica Aplicada
Universitat de Girona
E-17071 Girona, Spain
miquel.bofill@udg.edu

Dídac Busquets
Institut d'Informàtica i Aplicacions
Universitat de Girona
E-17071 Girona, Spain
didac.busquets@udg.edu

Mateu Villaret
Dept. Informàtica i Matemàtica Aplicada
Universitat de Girona
E-17071 Girona, Spain
mateu.villaret@udg.edu

Abstract

The presence of uncertainty in the real world makes robustness to be a desired property of solutions to constraint satisfaction problems. Roughly speaking, a solution is robust if it can be easily repaired when unexpected events happen. This issue has already been addressed in the frameworks of Boolean satisfiability (SAT) and Constraint Programming (CP). Most works on robustness implement search algorithms to look for such solutions instead of taking the declarative approach of reformulation, since reformulation tends to generate prohibitively large formulas, especially in the CP setting.

On the other hand, recent works suggest the use of SAT and Max-SAT encodings for solving CP instances. In this paper we present how robust solutions to weighted Max-SAT problems can be effectively obtained via reformulation into pseudo-Boolean formulae, thus providing a much flexible approach to robustness. We illustrate the use of our approach in the robust combinatorial auctions setting and provide some promising experimental results.

Categories and Subject Descriptors F.4.1 [*Mathematical Logic and Formal Languages*]: Mathematical Logic—Logic and constraint programming

General Terms Performance, Theory

Keywords Max-SAT, Robustness

1. Introduction

When dealing with real world problems, uncertainty is an important feature to take into account. For instance, in a job-shop scheduling problem (i.e., the problem of distributing tasks amongst machines, while minimizing makespan), if a machine breaks down, a new solution must be computed. Such a new solution should be fast to compute, and should be close to the initial one (e.g. in the job-shop problem, it is not desirable to reassign a large number of tasks). Therefore, instead of looking for an optimal solution, which may be brittle and not comply with the two mentioned requirements, one could directly look for a solution that can be "easily" repaired. Such a solution is said to be a *robust solution*. Obviously, this robust solution may probably be suboptimal, when comparing it with the outcome of the optimal solution to the problem if robustness were not to be considered. This fact has been sometimes called *the price of robustness* [3], but in many real world situations, it is worth sacrificing some optimality for a stronger solution.

There are existing works on solution robustness for CP [8–11, 13] and (plain) SAT [6, 19]. To deal with the problem, there are two main approaches: reformulation and search-based algorithms. The idea of reformulation is to extend the initial instance so that a solution to the extended instance is a robust solution to the initial one [6]. On the other hand, search-based algorithms look for a robust solution with backtracking, propagation and consistency techniques [8–11, 13]. Previous works [7] claim that, at least for CP, the reformulation approach results in prohibitively large formulas.

Max-SAT is becoming a competitive approach for solving combinatorial optimization problems [16] in the CP framework, and also to deal with Max-Constraint Satisfaction Problems [1]. For this reason, in this paper we consider the unaddressed problem of solution robustness in weighted Max-SAT, and show that reformulation is feasible in this setting. Our approach is declarative in the sense that the notion of robustness can be directly expressed in the original formulation of the problem with no need of changing the underlying solving method. Moreover, in contrast with search-based methods, with our declarative approach the reformulation is extremely flexible and it can be easily adapted to other notions of robustness, again without the need of changing the underlying solving method. For instance, we could add dependencies between repairable and breakable variables, introduce failure probabilities, or allow partial repairability, among others. Contrarily to this, in the search-based approach, if the notion of robustness is modified, the search-based algorithm must probably be modified too.

Generalizing Ginsberg's notion of robust solution for plain SAT [6], we define a robust solution to a weighted Max-SAT instance as a (minimal cost) solution fulfilling that when at most a variables break (i.e., change their truth value) then it can be repaired by changing the values of at most b disjoint variables, and the costs of the initial solution and the repaired ones are bounded by a given threshold β. This is called an (a, b, β)-super solution. Notice that in robust weighted Max-SAT we are interested not only in keeping the number of changes low, but also in the cost of the solutions. Hence, roughly speaking, while a is limiting the scope of the breakages, b is limiting the distance to the initial solution and β is limiting suboptimality. Actually, as we will see later in the paper, we have further generalized this notion of super solution so that it can also deal with don't-care variables for the repairs.

As a by-product, we improve the naive encoding of Ginsberg [6] for SAT by using pseudo-Boolean cardinality constraints. With our approach, the increase in size of the reformulated (robust) formula

* This material is based upon work supported by the Spanish Ministry of Science and Innovation through the project SuRoS (ref. TIN2008-04547/TIN).

http://doi.acm.org/10.1145/1836089.1836098

is $\mathcal{O}(n^a)$ instead of $\mathcal{O}(n^{a+b})$ (as it is the case in the Ginsberg's approach), where n is the number of variables, a is the number of breaks and b is the number of repairs. That is, in our case the number of repairs does not alter the size of the formula. This is especially relevant since a is usually expected to be low and, as far as we know, all previous works restricted a to be 1 and b to be at most 3 (but typically 0 or 1). Notice that in this case we have only a linear increase in size. Most importantly, we have done satisfactory experiments with a up to 3 and b up to 8 in the auctions setting, by using the standard encoding of an auction as a partial weighted Max-SAT problem [14], and introducing the concept of robustness to deal with the problem of resource unavailability[1].

Another non-negligible aspect of our approach is that a repair for each possible breakage is given directly by the solution to the reformulated problem, and so there would be no need to search for a repair in the event of a breakage occurring. Let us recall that, when dealing with uncertainty and robust solutions, it is not only important that a new solution close to the original one exists to deal with unexpected events, but also that this repaired solution can be quickly computed.

The rest of the paper is organized as follows. Section 2 introduces some basic notions used throughout the paper. The concept of solution robustness for weighted Max-SAT is defined in Section 3, and two different encodings through reformulation are presented in Sections 4 and 5. Section 6 shows the application of robustness in the auctions setting. Finally, in Section 7 we conclude.

2. Preliminaries

A *propositional variable* is a variable whose value can be either $true$ or $false$. A propositional variable x is also called an *atom*, and a variable x or its negation $\overline{x}$ is called a *literal*. A *clause* is a disjunction of literals $l_1 \vee \cdots \vee l_n$. A Boolean formula is in *conjunctive normal form* (CNF) if it is written as a conjunction of clauses $C_1 \wedge \cdots \wedge C_m$. CNF formulas are sometimes denoted as a set of clauses $\{C_1, \ldots, C_m\}$.

An *interpretation* (or *truth assignment*) I for a formula F is a function mapping the variables of F to $\{true, false\}$. An interpretation I *satisfies* a formula F, denoted $I \models F$, if under this interpretation of variables and the usual truth table interpretation of the logical connectives, the formula F evaluates to $true$. An interpretation satisfying a formula F is called a *model* of F. A formula having some model is called *satisfiable*, and *unsatisfiable* otherwise.

SAT is the problem of determining the satisfiability of a propositional formula. In many particular cases of the satisfiability problem, formulae are required to be in CNF. *Max-SAT* is the problem of finding the maximum number of clauses that can be satisfied by any truth assignment. *Weighted Max-SAT* is a variant of Max-SAT where every clause has a weight, i.e., a weighted Max-SAT formula is a conjunction of weighted clauses of the form (C, w), where C is a clause and w is a natural number indicating the cost (weight) of the falsification of C. The cost of a truth assignment for the formula is the sum of the costs of the clauses falsified by this assignment. Given a weighted formula, weighted Max-SAT is the problem of finding a truth assignment with minimal cost. A particular case of weighted Max-SAT is *partial weighted Max-SAT* where some clauses are mandatory and have an infinite cost of falsification. In this case, a model is every interpretation satisfying all mandatory clauses.

A closely related problem to that of weighted Max-SAT is pseudo-Boolean optimization. *Pseudo-Boolean constraints* (PB-constraints) are linear constraints over Boolean variables, that is, constraints of the form $C_0 l_0 + \cdots + C_{n-1} l_{n-1} \geq C_n$ where, for all i, l_i is a literal and C_i is an integer constant. A true literal is interpreted as 1 and a false literal is interpreted as 0, so that a truth assignment satisfies a PB-constraint if the sum of the C_i whose corresponding l_i is assigned to true exceeds or is equal to the right-hand constant C_n. Hence, PB-constraints can be seen as a generalization of clauses, that coincide with clauses in the case that all the C_i are 1. The *pseudo-Boolean optimization* (PBO) problem consists in finding a satisfying assignment to a set of clauses that minimizes a given objective function of the form $\sum_{j=1}^{m} C_j x_j$, where C_j is a non-negative integer cost associated to the variable x_j. PBO is indeed a particular case of integer linear programming (ILP) which is known as 0-1 integer programming. Moreover, every PBO instance can be translated into a partial weighted Max-SAT formula, where each PB-constraint is translated into a set of mandatory clauses [5] and the objective function is translated into a set of non-mandatory clauses, each summand $C_j x_j$ becoming a unit clause $(\overline{x}_j, C_j)$.

The following notation is used in the rest of the paper. We denote by $\mathcal{X}$ the set of all possible variables and by $Var(F)$ the set of all variables occurring in a formula F. Given a (Boolean or pseudo-Boolean) formula F and a set of variables S, by $F_{\overline{S}}$ we denote the formula F where all literals with variables in S have been flipped, i.e., negated. For example, if $F = \overline{p} \vee q \vee r$ and $S = \{p, r\}$, then $F_{\overline{S}} = p \vee q \vee \overline{r}$.

3. Weighted Max-SAT Robustness

The following definition generalizes the one of [6] to partial weighted Max-SAT.

Definition 1. *Let F be a partial weighted Max-SAT formula and S_1, S_2 and S_3 be sets of variables occurring in F, such that $(S_1 \cup S_2) \cap S_3 = \emptyset$. A* $(S_1^a, S_2^b, S_3, \beta)$-supermodel *of F is a (minimal cost) model of F such that if we modify the values taken by the variables in a subset of S_1 of size at most a (breakage), then another model can be obtained by modifying the values of the variables in a disjoint subset of S_2 of size at most b (repair) and the values of any number of variables in S_3 (don't-care variables), and moreover the solution and all possible repaired solutions have a cost of at most β.*

When the don't-care variables set S_3 is empty we simply talk of (S_1^a, S_2^b, β)*-supermodels.*

We remark that a, b and β are constants, and β is a bound on the cost of all the solutions, i.e., both for the solution found and for each of its possible reparations. We could have defined instead β to be a penalty to be paid for repairing the solution found, regardless of the exact cost of that solution. However notice that, as said in the introduction, a repairable solution will probably be suboptimal, and hence it is worth imposing a bound also to its cost. This is the reason why β applies to the solution and all possible repaired solutions. Moreover, for simplicity reasons, we have chosen β to directly denote a cost instead of, say, a percentage of increment with respect to the optimal cost. Thus, if we are interested in having a value of β relative to the optimal cost, we just need to compute such optimal cost before computing a robust solution. Along this line, from our declarative approach to robustness it follows that optimality is independent from robustness of the solution, i.e., a robust solution can be in turn optimal or not, in the same way as the solution to any problem.

Example 1. *Let $F = (p \vee q) \wedge (\overline{q} \vee r) \wedge (\overline{q} \vee s) \wedge (p, 1) \wedge (q, 2) \wedge (\overline{r}, 3)$, and let $S_1 = \{p, q\}$, $S_2 = \{p, q\}$ and $S_3 = \{r\}$. We have that:*

- *The interpretation I_1 where $I_1(p) = true$ and all other variables are assigned to false is a model of F with cost 2, but not*

[1] There are other works [12] dealing with robustness in auctions, although they use search-based techniques and deal with other breakage sources such as bid withdrawal.

a $(S_1^1, S_2^1, S_3, 4)$-supermodel of F: if p switches to false (one break in S_1) then the first clause requires q to be switched to true (one repair in S_2); but this change falsifies the second and third clauses unless r and s are set to true. This is possible for r but not for s, which does not belong to S_3 (the set of don't-care variables), and hence this model is not repairable for all breakages.

- *The interpretation I_2 where $I_2(p) = true$, $I_2(s) = true$ and the other variables are assigned to false is a $(S_1^1, S_2^1, S_3, 4)$-supermodel of F, since:*
  - *I_2 is a model of F with cost 2.*
  - *The breakage where p switches to false is repairable by switching q and r to true, and F has a cost of 4 under this repaired interpretation.*
  - *The breakage where q switches to true is repairable by switching r to true, and F has a cost of 3 under this repaired interpretation.*
  - *There is no other repairable model for F with a smaller cost.*

Notice however that requiring cost minimality does not necessarily imply uniqueness of supermodels.

In the next two sections we show how supermodels can be found by means of reformulation.

4. Encoding Weighted Max-SAT Robustness into Logical Combinations of Pseudo-Boolean Constraints

Unless otherwise stated, in the following we assume that $F = C \wedge W$ is a partial weighted Max-SAT formula, where C denotes the set of mandatory clauses and W denotes the set of weighted, non-mandatory clauses. For the sake of simplicity, we assume that W consists only of unary clauses, i.e., $W = (l_1, w_1) \wedge \cdots \wedge (l_k, w_k)$, where l_i is a literal[2] and w_i is a weight for all i in $1..k$.

We define

$$B = \sum_{j \in 1..k} \bar{l}_j \cdot w_j$$

which amounts to the cost of the unsatisfied clauses in W.

We then define a PBO instance F_{SM}, which will give us a $(S_1^a, S_2^b, S_3, \beta)$-supermodel of F:

$$\begin{array}{ll} F_{SM} = & \\ \textbf{Minimize} & B \\ \textbf{Subject To} & C \wedge (B \leq \beta) \wedge \\ \bigwedge_{S \subseteq S_1, 1 \leq |S| \leq a} & \Big(\bigvee_{T \subseteq (S_2 \setminus S) \cup S_3, |T \cap S_2| \leq b} C_{\overline{S \cup T}} \wedge (B_{\overline{S \cup T}} \leq \beta) \Big) \end{array}$$

Notice that F_{SM} is not a set of PB-constraints, but a conjunction of disjunctions of PB-constraints. As noted in [17], some problems are most directly specified in this latter form. Since in this section we are only interested in showing correctness of the transformation, we will keep the encoding in this form. Moreover, notice that we use literals in the objective function B rather than only variables, but this can be avoided by introducing new variables if necessary. Also, $\leq$-constraints can be changed into $\geq$-constraints by negating all constants. Observe that minimizing B is indeed optional, i.e., it is only necessary if we want the robust solution to be optimal in turn. However, since we are adding robustness to Max-SAT, we will assume that optimality is desired for robust solutions.

Roughly speaking, the meaning of the PBO instance F_{SM} is the following: since we are in a pseudo-Boolean setting, we have first replaced the weighted clauses W of the original formula F by the objective function to be minimized, which corresponds to B, the sum of the weights of the falsified clauses. Then we have C, that is, the mandatory clauses of the original formula, and we add $B \leq \beta$ to bound the cost. Next, we need to be able to repair all possible breakages. The big *and* accounts for all possible (*breakage*) sets S of size smaller than or equal to a, and the big *or* ensures the existence of a repair for each S. This is done by means of considering all possible (*repair*) sets T of variables from S_2 (excluding the ones in S) and S_3, and limiting the number of variables from S_2 to b. By flipping the breakages and the repairs, each subformula $(C_{\overline{S \cup T}} \wedge (B_{\overline{S \cup T}} \leq \beta))$ enforces (see Lemma 1) the existence of a possible repair T for the breakage S, with a cost which is bounded by β.

Observe that, since a and b are constants, the size of F_{SM} is polynomially bounded by the size of F. Namely, F_{SM} is $\mathcal{O}(n^{a+b})$ larger than F, where n is the number of variables of F. In Section 5 we show how, by encoding the disjunctions of F_{SM} into cardinality constraints, an equisatisfiable formula which is only $\mathcal{O}(n^a)$ larger than F can be obtained.

Example 2. *Taking formula F of Example 1, with $S_1 = \{p, q\}$, $S_2 = \{p, q\}$, $S_3 = \{r\}$, $a = 1$, $b = 1$ and $\beta = 4$, we would get the following PBO instance:*

$$\begin{array}{lll} F_{SM} = & & \\ \textbf{Minimize} & \overline{p} + 2\overline{q} + 3r & \\ \textbf{Subject To} & (p \vee q) \wedge (\overline{q} \vee r) \wedge (\overline{q} \vee s) \wedge & \\ & (\overline{p} + 2\overline{q} + 3r \leq 4) \wedge & \\ \textit{Break on } p \Big\{ & \textit{No repair} & \Big(\Big((\overline{p} \vee q) \wedge (\overline{q} \vee r) \wedge (\overline{q} \vee s) \wedge \\ & & (p + 2\overline{q} + 3r \leq 4) \Big) \vee \\ & \textit{Repair } q & \Big((\overline{p} \vee \overline{q}) \wedge (q \vee r) \wedge (q \vee s) \wedge \\ & & (p + 2q + 3r \leq 4) \Big) \vee \\ & \textit{Repair } r & \Big((\overline{p} \vee q) \wedge (\overline{q} \vee \overline{r}) \wedge (\overline{q} \vee s) \wedge \\ & & (p + 2\overline{q} + 3\overline{r} \leq 4) \Big) \vee \\ & \textit{Repair } q, r & \Big((\overline{p} \vee \overline{q}) \wedge (q \vee \overline{r}) \wedge (q \vee s) \wedge \\ & & (p + 2q + 3\overline{r} \leq 4) \Big) \Big) \wedge \\ \textit{Break on } q \Big\{ & \textit{No repair} & \Big(\Big((p \vee \overline{q}) \wedge (q \vee r) \wedge (q \vee s) \wedge \\ & & (\overline{p} + 2q + 3r \leq 4) \Big) \vee \\ & \textit{Repair } p & \Big((\overline{p} \vee \overline{q}) \wedge (q \vee r) \wedge (q \vee s) \wedge \\ & & (p + 2q + 3r \leq 4) \Big) \vee \\ & \textit{Repair } r & \Big((p \vee \overline{q}) \wedge (q \vee \overline{r}) \wedge (q \vee s) \wedge \\ & & (\overline{p} + 2q + 3\overline{r} \leq 4) \Big) \vee \\ & \textit{Repair } p, r & \Big((\overline{p} \vee \overline{q}) \wedge (q \vee \overline{r}) \wedge (q \vee s) \wedge \\ & & (p + 2q + 3\overline{r} \leq 4) \Big) \Big) \end{array}$$

As mentioned in Example 1, a $(S_1^1, S_2^1, S_3, 4)$-supermodel of formula F is $\{p, \overline{q}, \overline{r}, s\}$. Notice that this interpretation satisfies

[2] Notice that any Max-SAT formula can be transformed into an equisatisfiable one fulfilling this requirement by reification, i.e., by replacing any weighted clause (G, w) such that G is not unary by $(G \leftrightarrow b) \wedge (b, w)$, where b is a fresh variable.

the first block of clauses of F_{SM} (original formula F), the last set of clauses of the second block (a break on p with repairs on q and r), and also the third set of clauses of the last block (a break on q with a repair on r).

Now we proceed to prove the correctness of the F_{SM} reformulation.

Definition 2. (Flipped interpretation). *Let S be a set of variables and I be an interpretation. Then $I_{\overline{S}}$ denotes the interpretation such that $I_{\overline{S}}(x) = \overline{I(x)}$ if $x \in S$ and $I_{\overline{S}}(x) = I(x)$ for all other variables x.*

Lemma 1. *Let F be a Boolean (or pseudo-Boolean) formula and S be a set of propositional variables occurring in F. Then I is a model of F if and only if $I_{\overline{S}}$ is a model of $F_{\overline{S}}$.*

Proof. We proceed by induction on the size of S. If S is empty, the lemma holds trivially. To conclude we need to show that if $I \models F \Leftrightarrow I_{\overline{S}} \models F_{\overline{S}}$ then $I \models F \Leftrightarrow I_{\overline{S\cup\{p\}}} \models F_{\overline{S\cup\{p\}}}$ for every set of propositional variables S and every propositional variable p not included in S. For this observe that, for every literal l with the variable p (that is p or $\overline{p}$), we have $I_{\overline{S}}(l) = I_{\overline{S\cup\{p\}}}(\overline{l})$. Therefore, $I_{\overline{S}} \models F_{\overline{S}} \Leftrightarrow I_{\overline{S\cup\{p\}}} \models F_{\overline{S\cup\{p\}}}$ and hence if $I \models F \Leftrightarrow I_{\overline{S}} \models F_{\overline{S}}$ then $I \models F \Leftrightarrow I_{\overline{S\cup\{p\}}} \models F_{\overline{S\cup\{p\}}}$. □

The next theorem follows the spirit of the result of [6], but adding costs and *don't-care* variables.

Theorem 1. *Let F be a partial weighted Max-SAT formula, let S_1, S_2 and S_3 be sets of variables occurring in F such that $(S_1 \cup S_2) \cap S_3 = \emptyset$, and let a, b and β be non-negative integer constants. Then F has an $(S_1^a, S_2^b, S_3, \beta)$-supermodel if and only if the PBO instance F_{SM} has a solution.*

Proof. Let us recall that a $(S_1^a, S_2^b, S_3, \beta)$-supermodel of F is a minimal cost model of F such that if we modify the values taken by the variables in a subset of S_1 of size at most a, then another model can be obtained by modifying the values of the variables in a disjoint subset of S_2 of size at most b and the values of any number of variables of S_3 and, moreover, the solution and all possible repaired solutions have a cost of at most β.

By assumption, $F = C \wedge W$ where C is a set of mandatory clauses and W is a set of weighted (non-mandatory) unit clauses. For the right-to-left implication, let I be a solution (i.e., an optimal truth assignment) to F_{SM}. From the definition of B, we clearly have that the cost of the unsatisfied clauses in W is at most β. Notice also that I is a model of F, since F is included in the PB-constraints of F_{SM}. To conclude we will show that I is a model of F such that if we modify the values taken by the variables in a subset S of S_1 with $|S| \leq a$, then another model can be obtained by modifying the values of the variables in a disjoint subset T of $S_2 \cup S_3$ with $|T \cap S_2| \leq b$ and, moreover, the repaired solution has a cost of at most β. If $S = \emptyset$ then this trivially holds taking $T = \emptyset$. Otherwise $a > 0$ and $1 \leq |S| \leq a$ and, since I is a solution of F_{SM}, then $I \models \vee_{T\subseteq(S_2\setminus S)\cup S_3, |T\cap S_2|\leq b}(C_{\overline{S\cup T}} \wedge (B_{\overline{S\cup T}} \leq \beta))$, that is, $I \models C_{\overline{S\cup T}} \wedge (B_{\overline{S\cup T}} \leq \beta)$ for some T in $(S_2 \setminus S) \cup S_3$ with $|T \cap S_2| \leq b$. Now, since $I \models C_{\overline{S\cup T}}$, by Lemma 1 we have $I_{\overline{S\cup T}} \models (C_{\overline{S\cup T}})_{\overline{S\cup T}}$ and, since $(C_{\overline{S\cup T}})_{\overline{S\cup T}} = C$, we have $I_{\overline{S\cup T}} \models C$ as desired. Finally, we show that $I \models B_{\overline{S\cup T}} \leq \beta$ implies that the cost of the unsatisfied clauses in W under interpretation $I_{\overline{S\cup T}}$ is at most β. Notice that $I(B)$ is the cost of the unsatisfied clauses in W. Now, considering the interpretation $I_{\overline{S\cup T}}$ we have that $I_{\overline{S\cup T}}(B_{\overline{S\cup T}}) = I(B)$. Therefore, $I_{\overline{S\cup T}}((B_{\overline{S\cup T}})_{\overline{S\cup T}}) = I(B_{\overline{S\cup T}})$, that is, $I_{\overline{S\cup T}}(B) = I(B_{\overline{S\cup T}})$. From this and the fact that $I \models B_{\overline{S\cup T}} \leq \beta$, we finally get that $I_{\overline{S\cup T}}(B)$ (which amounts to the cost of the unsatisfied clauses in W under interpretation $I_{\overline{S\cup T}}$) is at most β.

For the left-to-right implication, let I be a model of F such that if we modify the values taken by the variables in a subset S of S_1 of size at most a, then another model can be obtained by modifying the values of the variables in a disjoint subset T of $S_2 \cup S_3$ such that $|T \cap S_2| \leq b$ (that is, $I_{\overline{S\cup T}} \models F$) and, moreover, both the solution and the repaired solution have a cost of at most β. We will show that every such an interpretation I is a solution of F_{SM}. Since the solution has cost at most β, we have that $I \models B \leq \beta$. Now it remains to be proved that it holds $I \models \wedge_{S\subseteq S_1, 1\leq|S|\leq a}(\vee_{T\subseteq(S_2\setminus S)\cup S_3, |T\cap S_2|\leq b}(C_{\overline{S\cup T}} \wedge (B_{\overline{S\cup T}} \leq \beta)))$. For this recall that, by assumption, for every subset S of S_1 with $|S| \leq a$ there exists a subset T of $(S_2 \setminus S) \cup S_3$ with $|T \cap S_2| \leq b$ such that $I_{\overline{S\cup T}} \models C$. Then, by Lemma 1, $(I_{\overline{S\cup T}})_{\overline{S\cup T}} \models C_{\overline{S\cup T}}$ and, since $(I_{\overline{S\cup T}})_{\overline{S\cup T}} = I$, we have $I \models C_{\overline{S\cup T}}$. In order to show that $I \models B_{\overline{S\cup T}} \leq \beta$, recall that the cost of the repaired solution is at most β, which means that $I_{\overline{S\cup T}} \models B \leq \beta$. Then since, as seen before, $I_{\overline{S\cup T}}(B) = I(B_{\overline{S\cup T}})$, we have $I \models B_{\overline{S\cup T}} \leq \beta$. □

Observe that, in the previous proof, we have established a bijection between supermodels of F and solutions of F_{SM}. Moreover, since B denotes the cost of the unsatisfied clauses in W, it is not difficult to see that optimality is preserved. Hence, it follows that a solution of F_{SM} is precisely a $(S_1^a, S_2^b, S_3, \beta)$-supermodel of F.

Corollary 1. *Let F be a partial weighted Max-SAT formula, let S_1, S_2 and S_3 be sets of variables occurring in F such that $(S_1 \cup S_2) \cap S_3 = \emptyset$, and let a, b and β be non-negative integer constants. Then every optimal solution of F_{SM} is a $(S_1^a, S_2^b, S_3, \beta)$-supermodel of F.*

5. Robust Max-SAT with Cardinality Constraints

In this section we show how we can get rid of the disjunctions of the previous encoding by means of cardinality constraints. A formula which is only $\mathcal{O}(n^a)$ larger than F is obtained. This is especially important, since it means that the complexity of our approach does not depend on the number of repairs, but only on the number of breakages, which is usually assumed to be low. In fact, to our knowledge, in all previous works on robustness the number of breakages has always been set to one. Notice that in this case we only have a linear increase in size.

Definition 3. (Variable renaming). *Let R and S be sets of variables. The function $\delta_{R,S} : \mathcal{X} \to \mathcal{X}$ is defined as $\delta_{R,S}(x) = x^S$ for every variable $x \in R$, where x^S is a new atom, and $\delta_{R,S}(x) = x$ if $x \notin R$. When S is irrelevant, we simply write δ_R.*

Definition 4. (Formula renaming). *Let F be a Boolean (or pseudo-Boolean) formula, and R and S be sets of variables. Then $F^{\delta_{R,S}}$ denotes the formula F where all occurrences of each variable x have been replaced by $\delta_{R,S}(x)$.*

Example 3. *If $F = \overline{p} \vee q \vee (r \wedge t \wedge p)$, $R = \{p, q\}$ and $S = \{r, s\}$, then $F^{\delta_{R,S}} = \overline{p}^{\{r,s\}} \vee q^{\{r,s\}} \vee (r \wedge t \wedge p^{\{r,s\}})$.*

Definition 5. (Difference cardinality). *Let R and S be sets of variables, and $\delta_{R,S}$ be a variable renaming function. Then we define the* difference cardinality formula *as*

$$\nabla^{\delta_{R,S}} = \sum_{x\in R}(x \neq \delta_{R,S}(x)) = \sum_{x\in R}(x \neq x^S).$$

Observe that we are using Boolean values as 0-1 integers in the definition of $\nabla^{\delta_{R,S}}$. Hence, if necessary, $\nabla^{\delta_{R,S}}$ can be translated into a pseudo-Boolean expression $\sum_{x\in R} d_{x^S}$ where $d_{x^S} = XOR(x, x^S)$, that is, having $(x + x^S + \overline{d_{x^S}} \geq 1) \wedge (x + \overline{x^S} + d_{x^S} \geq 1) \wedge (\overline{x} + x^S + d_{x^S} \geq 1) \wedge (\overline{x} + \overline{x^S} + \overline{d_{x^S}} \geq 1)$ for each $x \in R$.

Now we define the new encoding by removing the big disjunction on the repairs of F_{SM}. Instead, we have the following PBO instance with cardinality constraints:

$$
\begin{array}{ll}
F^{\nabla}_{SM} = & \\
\textbf{Minimize} & B \\
\textbf{Subject To} & C \wedge (B \leq \beta) \wedge \\
\bigwedge_{S \subseteq S_1, 1 \leq |S| \leq a} & \Big(C^{\delta_{((S_2 \setminus S) \cup S_3), S}}_{\overline{S}} \wedge (B^{\delta_{((S_2 \setminus S) \cup S_3), S}}_{\overline{S}} \leq \beta) \wedge \\
& (\nabla^{\delta_{(S_2 \setminus S), S}} \leq b) \Big)
\end{array}
$$

Notice that $\delta_{((S_2 \setminus S) \cup S_3), S}$ is a renaming of the variables in $(S_2 \setminus S) \cup S_3$, by superscripting them with S. Since in our transformation a different renaming is needed for every considered subset S of S_1, we just choose that set S for the renaming.

Example 4. *Taking formula F from Example 1, and being $S_1 = \{p, q\}$, $S_2 = \{p, q\}$, $S_3 = \{r\}$, $a = 1$, $b = 1$, and $\beta = 4$, we would get the following PBO instance:*

$$
\begin{array}{ll}
F^{\nabla}_{SM} = & \\
\textbf{\textit{Minimize}} & \overline{p} + 2\overline{q} + 3r \\
\textbf{\textit{Subject To}} & (p \vee q) \wedge (\overline{q} \vee r) \wedge (\overline{q} \vee s) \wedge \\
& (\overline{p} + 2\overline{q} + 3r \leq 4) \wedge \\
& (\overline{p} \vee q^{\{p\}}) \wedge (\overline{q}^{\{p\}} \vee r^{\{p\}}) \wedge (\overline{q}^{\{p\}} \vee s) \wedge \\
& (p + 2\overline{q}^{\{p\}} + 3r^{\{p\}} \leq 4) \wedge (q \neq q^{\{p\}} \leq 1) \\
& \wedge (p^{\{q\}} \vee \overline{q}) \wedge (q \vee r^{\{q\}}) \wedge (q \vee s) \wedge \\
& (\overline{p}^{\{q\}} + 2q + 3r^{\{q\}} \leq 4) \wedge (p \neq p^{\{q\}} \leq 1)
\end{array}
$$

One solution for F^{∇}_{SM} is $\{p, \overline{q}, \overline{r}, s, q^{\{p\}}, r^{\{p\}}, p^{\{q\}}, r^{\{q\}}\}$. That is, as already shown in Example 1, the initial solution would be $\{p, \overline{q}, \overline{r}, s\}$, the repair for a break on p would be to change the values of q and r from false ($\overline{q}, \overline{r}$) to true ($q^{\{p\}}, r^{\{p\}}$), and the repair for a break on q would be to change the value of r, again from false ($\overline{r}$) to true ($r^{\{q\}}$); under this interpretation, the value of p would be unaffected by a break on q (i.e., $p = p^{\{q\}}$).

Now we proceed to prove the correctness of the F^{∇}_{SM} reformulation.

Lemma 2. *Let F be a Boolean (or pseudo-Boolean) formula, R be a subset of the variables of F and I be an interpretation. Then $I \models F^{\delta_R}$ if and only if $I \models F_{\overline{D}}$, where $D = \{x \in Var(F) \mid I(x) \neq I(\delta_R(x))\}$.*

Proof. First of all notice that from the definition of D it follows that the domain of I is $Var(F) \cup Var(F^{\delta_R})$. Moreover, since F^{δ_R} is a variable renamed version of F and $F_{\overline{D}}$ is a variable flipped version of F, we have that both formulas are the same except for some variable renamings (for each variable in R) and negations (for each variable in D). Hence, for every literal l in $F_{\overline{D}}$ we have a corresponding literal l' in F^{δ_R}.

Now, let l be any literal occurring in $F_{\overline{D}}$, and l' its corresponding literal in F^{δ_R}. W.l.o.g., we assume that l is either of the form x or $\overline{x}$, where x is a variable in $Var(F)$. We distinguish between two cases. If $I(x) = I(\delta_R(x))$ then $x \notin D$ and, hence, l occurs in $F_{\overline{D}}$ with the same polarity with which l' occurs in F^{δ_R}. Then, since $I(x) = I(\delta_R(x))$, we have that $I(l) = I(l')$. If, otherwise, $I(x) \neq I(\delta_R(x))$ then $x \in D$ and, hence, l occurs in $F_{\overline{D}}$ with the opposite polarity with which l' occurs in F^{δ_R}. Then, since $I(x) \neq I(\delta_R(x))$, and literals l and l' have opposite polarity, we have that $I(l) = I(l')$.

Finally, since $I(l) = I(l')$ for all literals, we have that $I \models F^{\delta_R} \Leftrightarrow I \models F_{\overline{D}}$. □

Lemma 3. *Let F be a pseudo-Boolean formula, S_1, S_2 and S_3 be sets of variables occurring in F such that $(S_1 \cup S_2) \cap S_3 = \emptyset$, a and b be non-negative integer numbers, and I be an interpretation. Then*

$$I \models \bigwedge_{S \subseteq S_1, 1 \leq |S| \leq a} \Big(\bigvee_{T \subseteq (S_2 \setminus S) \cup S_3, |T \cap S_2| \leq b} F_{\overline{S \cup T}} \Big)$$

if and only if there exists an interpretation I' such that $I'(x) = I(x)$ for all $x \in Var(F)$ and

$$I' \models \bigwedge_{S \subseteq S_1, 1 \leq |S| \leq a} \Big(F^{\delta_{((S_2 \setminus S) \cup S_3), S}}_{\overline{S}} \wedge (\nabla^{\delta_{(S_2 \setminus S), S}} \leq b) \Big)$$

Proof. For the left-to-right direction, we have that for every set $S \subseteq S_1$ with $1 \leq |S| \leq a$, there exists a set $T \subseteq (S_2 \setminus S) \cup S_3$ with $|T \cap S_2| \leq b$ such that $I \models F_{\overline{S \cup T}}$. We can define an interpretation I^S whose domain is $Var(F) \cup Var(F^{\delta_{((S_2 \setminus S) \cup S_3), S}})$ and such that $I^S(x) = I(x)$ for all $x \in Var(F)$ and $I^S(x^S) \neq I(x)$ if and only if $x \in T$ for all $x^S \in Var(F^{\delta_{((S_2 \setminus S) \cup S_3), S}})$. Now observe that $F_{\overline{S \cup T}} = (F_{\overline{S}})_{\overline{T}}$ since S and T are disjoint. Moreover, since $I \models (F_{\overline{S}})_{\overline{T}}$ and $I^S(x) = I(x)$ for all $x \in Var(F)$, then also $I^S \models (F_{\overline{S}})_{\overline{T}}$. Then, by Lemma 2, taking $F_{\overline{S}}$ for F, $((S_2 \setminus S) \cup S_3)$ for R, I^S for I and T for D, we have $I^S \models F^{\delta_{((S_2 \setminus S) \cup S_3), S}}_{\overline{S}}$. Concerning $(\nabla^{\delta_{(S_2 \setminus S), S}} \leq b)$, by Definition 5 and construction of I^S we have $I^S(\nabla^{\delta_{(S_2 \setminus S), S}}) = |T \cap (S_2 \setminus S)| \leq |T \cap S_2| \leq b$. Finally, since all considered sets S are different, the domains of every pair of such interpretations I^S can only have non-renamed variables of F in common. Then, since all interpretations I^S give the same value to non-renamed variables of F, we conclude that there exists an interpretation I' that is compatible with all I^S, that is, an interpretation with the same truth assignment as every I^S.

For the right-to-left implication, we have the following: $I' \models F^{\delta_{((S_2 \setminus S) \cup S_3), S}}_{\overline{S}} \wedge (\nabla^{\delta_{(S_2 \setminus S), S}} \leq b)$ for every set $S \subseteq S_1$ with $1 \leq |S| \leq a$. We define T as the set of variables x in $(S_2 \setminus S) \cup S_3$ such that $I'(x^S) \neq I'(x)$. We trivially have that $T \subseteq (S_2 \setminus S) \cup S_3$. Now we show that $|T \cap S_2| \leq b$. For this, first of all note that, since S and S_3 are disjoint, we have that $T \cap (S_2 \setminus S) = T \cap S_2$. Then, by definition of T, we have that $\sum_{x \in (S_2 \setminus S)} (I'(x^S) \neq I'(x)) = |T \cap (S_2 \setminus S)| = |T \cap S_2|$. Moreover, by assumption, we have that $I' \models (\nabla^{\delta_{(S_2 \setminus S), S}} \leq b)$, i.e., $\sum_{x \in (S_2 \setminus S)} (I'(x^S) \neq I'(x)) \leq b$. Therefore, $|T \cap S_2| \leq b$. Next, since $I' \models F^{\delta_{((S_2 \setminus S) \cup S_3), S}}_{\overline{S}}$, we also have that $I' \models F_{\overline{S \cup T}}$ by Lemma 2, taking $F_{\overline{S}}$ for F, $((S_2 \setminus S) \cup S_3)$ for R, and I' for I. Finally since $I'(x) = I(x)$ for all $x \in Var(F)$, we have that $I \models F_{\overline{S \cup T}}$, which lets us conclude. □

Theorem 2. *Let F be a partial weighted Max-SAT formula, let S_1, S_2 and S_3 be sets of variables occurring in F such that $(S_1 \cup S_2) \cap S_3 = \emptyset$, and let a, b and β be non-negative integer constants. Then F has a $(S^a_1, S^b_2, S_3, \beta)$-supermodel if and only if the PBO instance F^{∇}_{SM} has a solution.*

Proof. By assumption, $F = C \wedge W$ where C is a set of mandatory clauses and $W = (l_1, w_1) \wedge \cdots \wedge (l_k, w_k)$ is a set of weighted, non-mandatory unit clauses. Moreover, we define $B = \sum_{j \in 1..k} \bar{l}_j \cdot w_j$, for all j in $1..k$. Hence, we conclude by Theorem 1 and Lemma 3, taking $F = C \wedge (B \leq \beta)$. □

Corollary 2. *Let F be a partial weighted Max-SAT formula, let S_1, S_2 and S_3 be sets of variables occurring in F such that $(S_1 \cup S_2) \cap S_3 = \emptyset$, and let a, b and β be non-negative integer constants. Then every optimal solution of F^{∇}_{SM} is a $(S^a_1, S^b_2, S_3, \beta)$-supermodel of F.*

In this section we have proved that finding supermodels for partial weighted Max-SAT formulae F amounts to solving the

corresponding PBO instances F^{∇}_{SM}. It is worth noting that solutions of F^{∇}_{SM} allow us to immediately find an appropriate repair for every permitted breakage: given a solution I', and a breakage S, a repair (set of variables that must flip their value) for S is the set $\{x \in Var(F) \mid I'(x) \neq I'(x^S)\}$. For instance, in Example 4, a solution for F^{∇}_{SM} is $\{p, \overline{q}, \overline{r}, s, q^{\{p\}}, r^{\{p\}}, p^{\{q\}}, r^{\{q\}}\}$. Notice that if the breakage is $\{p\}$ then the set of variables marked with $\{p\}$ that must change their value (w.r.t. the unmarked ones) is $\{q, r\}$. If the breakage is $\{q\}$ then the set of variables marked with $\{q\}$ that must change their value is $\{r\}$; p must not change its value since p and $p^{\{q\}}$ occur with the same polarity in the solution.

In the next section we show how our supermodels can be applied to a concrete problem, namely combinatorial auctions.

6. Application to Auctions

We have tested our approach for finding supermodels for partial weighted Max-SAT formulae in the setting of *Combinatorial Auctions* [4]. The problem posed by a combinatorial auction is equivalent to several existing CP problems, such as the matching problem, the set packing problem, or the multi-dimensional knapsack problem. Moreover, it is used as one of the benchmarks in the partial weighted Max-SAT category of the SAT competition [2]. Formally, in a combinatorial auction we have:

- a set of N items (or goods), and
- a set of M bids of the form (S_j, p_j), where each $S_j = \{i_1, \ldots, i_{n_j}\} \subseteq \{1..N\}$ is the subset of n_j goods (i.e., bundle) requested in bid j, and $p_j \in \mathbb{N}^+$ is the value (price) of the bid.

The problem is then to select the winning bids, so that no good is allocated more than once, and the revenue of the auctioneer is maximized:

$$\max \sum_{i=1}^{M} b_i \cdot p_i \quad \text{s.t.} \quad \forall j \in \{1..N\}, \sum_{\{i \mid j \in S_i\}} b_i \leq 1$$

where b_i is a binary variable indicating whether bid i is winner ($b_i = 1$) or loser ($b_i = 0$).

This problem can be posed as a CP problem where the set of variables is $\mathcal{X} = \{b_1, \ldots, b_M\}$, the domain of each variable is $\mathcal{D}(b_i) = \{0, 1\}$, and the constraints reflect the fact that one good can only be allocated to at most one bidder, and so they prevent each pair of bids requesting the same good from being both winners at the same time. This CP problem could then be solved with any CP solver, or also with integer programming solvers. However, as mentioned in the Introduction, there is an increasing interest in solving CP problems using Max-SAT, and this is the approach we have taken.

We use the following encoding of a combinatorial auction as a partial weighted Max-SAT problem [14]:

- For each pair of bids i and j, $i \neq j$, such that $S_i \cap S_j \neq \emptyset$, we state the mandatory clause $(\overline{b}_i \vee \overline{b}_j)$ to indicate incompatibility between bids requesting the same good.
- For each bid i, we add a weighted unit clause (b_i, p_i) indicating that if bid i is not accepted, then there is a loss of revenue of p_i.

Although the obtained formula encodes a combinatorial auction, it does not consider any robustness notion. Therefore, we next show how we have extended this encoding in order to deal with robustness against resource unavailability. This problem arises when, once a solution to the auction has been already computed, some resources become unavailable. In such a case, the winning bids requesting the "broken" resources cannot be fulfilled anymore, causing a decrease in the auctioneer's revenue. He could then decide to run the auction again, without the unavailable resources, to get a better revenue than with the "broken solution". However, the new solution may be completely different from the initial one, and bids that were initially winners may become losers, and vice-versa. Such behavior could not be well seen by the participating bidders, especially by the initially winner ones. Therefore, a better option for the auctioneer would be to directly look for a robust solution, repairable with few changes in case of a breakage occurring. Moreover, this robust solution should report a reasonable revenue, both in the case no break occurs, and also for the potential repairs.

In order to find a robust solution to the auction, we need to look for a supermodel of the formula encoding it. Since we want robustness against resource unavailability, we must first extend the encoding of the auction so that it takes into account the resources. To do so, we define a new set of Boolean variables, $\{g_1, \ldots, g_N\}$, where g_i indicates whether good i is available or not, and then for each bid j we add the clause $b_j \rightarrow g_{i_1} \wedge \cdots \wedge g_{i_{n_j}}$ to indicate that, whenever bid j is accepted, then all goods it requests must be available. Finally, the conjunction of the clauses defined previously, together with the resource availability clauses, defines the formula F^A encoding the auction.

Once we have encoded the auction into the partial weighted Max-SAT formula F^A, we are ready to look for a supermodel for it. Since the potential breaks come from the resources, we set the breakable set to be $S_1 = \{g_1, \ldots, g_N\}$. Similarly, since we can only repair bid assignments, and we want these changes to be bounded, we set the repairable set to be $S_2 = \{b_1, \ldots, b_M\}$, and $S_3 = \emptyset$. Finally, we should set the values of maximum number of allowed breakages (a), maximum number of repairs (b), and a threshold indicating the minimum revenue that a solution and its potential repairs should report (γ). These three parameters would be set by the auctioneer, according to its needs. Then, a robust solution to the auction would be a (S^a_1, S^b_2, β)-supermodel of F^A, with $\beta = \left(\sum_{i=1}^{M} p_i\right) - \gamma$, where p_i is the value of the i-th bid. Note that the change from γ to β amounts for the fact that in an auction we seek to maximize the revenue, while in weighted Max-SAT we seek to minimize the cost.

6.1 Experimental Results

We have used the Combinatorial Auctions Testsuite (CATS) [15] to generate several instances of auctions. The number of goods offered in the auction and the number of participating bids have been fixed to 20 and 40, respectively. For the rest of parameters we have varied them as follows: number of allowed breakages a in {1,2}, number of allowed repairs b in {1,2,4,8}, and the value of γ (which in turn defines the value of β) has been set according to several percentages of the revenue of the optimal solution ($\%opt$) when robustness is not considered: {80%, 60%}.

For each configuration of the parameters, we have generated 50 auction instances of the L7 distribution[3] of CATS. Then, for each instance we first find the optimal revenue without considering robustness, and then we solve each of the $(a, b, \%opt)$ robust versions of the problem. To solve each instance, we have used several solvers: two pseudo-Boolean solvers, BSOLO v3.1 [18] and SCIP v1.2 (with SoPLEX 1.4.2)[4], and two linear programming solvers, CPLEX 12.1[5] and GLPK 4.35[6]. We have set a solving timeout of

[3] We have chosen this distribution since it generates bids with bundles that are not too large (i.e., with just a few of the resources being auctioned). We wanted to avoid having large bundles because they do not help that much in repairing a broken solution. However, we should also study the behavior of our approach with the other distributions of CATS.

[4] http://scip.zib.de/

[5] http://www.ilog.com/products/cplex

[6] http://www.gnu.org/software/glpk

a	b	$\%opt$	%SAT	BSOLO %Sol	BSOLO SAT Time	BSOLO UNSAT Time	CPLEX %Sol	CPLEX SAT Time	CPLEX UNSAT Time	Size
1	1	60	90	100	0.25 (0.04)	0.25 (0.02)	100	0.54 (0.51)	3.27 (1.15)	3246 vars, 14714 constr
		80	2		0.25 (0.00)	0.21 (0.03)		0.44 (0.00)	0.74 (0.61)	
	2	60	98		0.30 (0.08)	0.50 (0.00)		0.76 (0.94)	5.62 (0.00)	
		80	2		0.24 (0.00)	0.27 (0.07)		0.38 (0.00)	2.87 (2.43)	
	4	60	100		0.38 (0.12)	-		0.70 (0.72)	-	
		80	18		0.52 (0.26)	0.30 (0.11)		15.02 (16.10)	6.24 (14.44)	
	8	60	100		0.25 (0.03)	-		0.31 (0.17)	-	
		80	54		0.29 (0.04)	0.21 (0.02)		1.61 (1.71)	0.40 (0.22)	
2	1	60	0	100	-	29.25 (3.39)	100	-	30.99 (11.28)	33160 vars, 149297 constr
		80	0		-	26.37 (5.40)	100	-	5.44 (1.71)	
	2	60	4		27.23 (1.79)	30.07 (3.87)	68	19.30 (7.86)	146.20 (83.96)	
		80	0		-	28.71 (3.28)	100	-	5.61 (3.55)	
	4	60	28		41.43 (7.11)	35.77 (8.02)	100	25.90 (0.00)	113.87 (96.76)	
		80	0		-	28.86 (3.50)	100	-	5.39 (7.27)	
	8	60	68		31.84 (3.31)	32.07 (4.12)	54	56.40 (64.48)	185.20 (90.39)	
		80	0		-	28.96 (3.74)	100	-	4.46 (4.89)	

a	b	$\%opt$	%SAT	GLPK %Sol	GLPK SAT Time	GLPK UNSAT Time	SCIP %Sol	SCIP SAT Time	SCIP UNSAT Time	Size
1	1	60	90	90	83.58 (74.57)	171.21 (77.63)	100	68.12 (24.62)	64.9 (4.71)	3246 vars, 14714 constr
		80	2	8	147.49 (0.00)	61.21 (44.01)	100	77.5 (0.00)	57.01 (19.52)	
	2	60	98	54	194.98 (77.67)	*time out*	86	165.53 (51.94)	*time out*	
		80	2	48	107.73 (0.00)	198.34 (61.06)	90	*time out*	136.33 (50.5)	
	4	60	100	52	230.55 (60.37)	-	92	191.63 (58.18)	-	
		80	18	28	201.98 (0.00)	213.24 (44.66)	12	*time out*	188.82 (55.15)	
	8	60	100	78	230.14 (43.57)	-	84	176.82 (65.1)	-	
		80	54	54	250.52 (23.68)	216.24 (50.50)	32	173.66 (40.15)	116.00 (0.00)	
2	*	*	*	0	*time out*		0	*time out*		

Table 1. Experimentation results. Average results (deviation in brackets) of 50 instances with 20 goods and 40 bids, generated from the L7 distribution of CATS, and solved with BSOLO, CPLEX, SCIP and GLPK. A '-' indicates that there were no SAT or UNSAT instances in a given configuration, and *time out* indicates that the solver reached the time out of 300 seconds before finding a solution on all instances.

300 seconds. The experiments have been performed on an Intel Core i5 CPU at 2.66 GHz, with 4GB of RAM, running openSUSE 11.2 (kernel 2.6.31).

Table 1 shows the average percentage of satisfiability of the instances of each parameter configuration (%SAT), i.e., a robust solution exists, and for each solver, the percentage of instances solved before the timeout (%Sol) and the average time and deviation (in seconds) for solving the instances[7], differentiating the satisfiable and unsatisfiable cases. We also show the average size of the generated pseudo-Boolean instances (number of variables and constraints).

The first thing to notice is that the pseudo-Boolean solver BSOLO is the only solver capable of solving all the instances before the timeout. CPLEX does solve all the (1,*) instances, but its solving performance falls down to about 90% for the (2,*) instances. As for SCIP and GLPK, they could only solve about 62% and 74% of the (1,*) instances, respectively, and none of the (2,*) instances. Moreover, the time taken by SCIP and GLPK is two and three orders of magnitude more than the time taken by BSOLO. Comparing the solving times of BSOLO and CPLEX, we can observe that when solving the (1,*) instances, BSOLO is in most cases twice as fast as CPLEX for satisfiable instances, and two orders of magnitude faster for most of the unsatisfiable ones. However, with the (2,*) instances, we can see an interesting effect. When the percentage of optimality is set to 80%, CPLEX is one order of magnitude faster than BSOLO, while with the percentage set to 60%, CPLEX encounters hard problems and it takes one order of magnitude more than BSOLO, except for the (2,1) instances. Moreover, CPLEX fails to solve about 20% of the instances. A possible explanation to this behavior is that asking for a revenue of at least 80% of the optimal is a very strict restriction, and so CPLEX rapidly finds out that there is no solution to the problem (note that in all cases where CPLEX solves the instances quickly, the percentage of satisfiability is null). However, when relaxing the revenue constraint, the search space is enlarged and so it takes more time to solve the problem. On the other hand, BSOLO is not affected at all by this parameter, and all its solving times are almost the same.

We can also observe the effect of allowing more breakages to occur: when increasing a to 2, the solving time increases two orders of magnitude; by contrast, the value of b has not a considerable effect on the solving time. Regarding satisfiability, as mentioned before, we can observe that asking for a robust solution being 80%-optimal is way too strict, and so only a few instances have such a robust solution (percentage that increases as b increases).

We have to point out that the main goal of the experimentation was not to perform a comparison of the solvers, but to show that our reformulation approach can be effectively implemented and solved. Actually, given the declarative nature of our approach, we are not bound to any specific solver, and we can profit from the advances in the state-of-the-art solvers, be them pure pseudo-Boolean solvers or more generic integer programming solvers.

Finally we should remark that, although not shown in the results, we have also solved (3,8) instances ($\sim$200000 variables and

[7] Computed only for those solved before the timeout.

1 million constraints) using BSOLO, with an average solving time of 180 seconds.

7. Conclusions and Further Work

In this paper we have proposed a mechanism for finding robust solutions to weighted Max-SAT problems. We have first extended the naive approach of Ginsberg [6] so that it can deal with the cost constraints needed in weighted Max-SAT and with don't-care variables. Then, we have simplified this encoding by using cardinality constraints. In doing so, the reformulation results in a much smaller problem in the pseudo-Boolean framework. Moreover, with our reformulation, the solution to the extended instance of the problem provides not only the supermodel for the initial problem, but also a possible repair for each of the potential breakages.

Our approach for finding supermodels using difference cardinality constraint resembles those presented in [7, 10, 11, 20] for the CP setting. The reformulation approach presented in [20] for finding (1,0)-super solutions, referred to as $P + P$ in [10], consists in duplicating the breakable variables and adding a *not equals* constraint between each original variable and its duplicate. In [7] it is mentioned that the same duplication approach could be generalized for finding (a, b)-super solutions. However, the authors argue that the size of the new problem would be prohibitive and opt for using a backtracking algorithm [11] where the original problem is only duplicated to check that an assignment can be repaired. Thus, although the algorithm finds a super solution, it does not provide the repairs, since it "forgets" them once it has checked the repairability of a solution. The main problem of using the reformulation approach in the CP setting is the space needed to store the variables' domains and the restrictions among them, which are replicated many times. In our approach we also duplicate the problem for each possible break through the renaming function, and limit the number of changes to be lower than b with the difference cardinality constraint. Fortunately, in the Boolean and pseudo-Boolean settings such reformulation is not that expensive, since the domains are restricted to $\{false, true\}$ ($\{0, 1\}$ in pseudo-Boolean) and the only restrictions are the Boolean clauses (inequalities in pseudo-Boolean).

Thanks to the parametric definition of supermodels, we can deal with distinct sources of breakage and repairs. For instance, in the auctions setting we have considered, we have focused on robustness with respect to good unavailability. However, we could easily take into account another source of breakage such as, e.g., bid withdrawal, by simply setting both the sets S_1 and S_2 to the variables that encode whether bids are winners or losers.

Our approach can be seen as a generic framework for robustness through reformulation, since with only slight changes in the encoding we can achieve other notions of robustness. For example, a robustness variant could be to directly designate the potential breaks to handle: instead of using S_1^a, we could decide what (combinations of) breaks deserve being repaired, which would be a subset of 2^{S_1}. This would be useful if we only want to consider those breaks having a non negligible probability of occurring. We could also think of a robustness notion where each breakable variable has a corresponding set of associated repairable variables. This could serve, for instance, in the presence of scheduling, where one should only look for repairs on the forthcoming assigned resources, or in an auction scenario where it is not permitted to make repairs by switching a winning bid to a loser one. Notice however that these robustness variants do not exactly fit our definition of $(S_1^a, S_2^b, S_3, \beta)$-supermodels for partial weighted Max-SAT, since the specification of the breakage and the repair sets would require more information. However, thanks to the high expressiveness of pseudo-Booleans we could easily directly encode such notions of robustness without moving out from this setting.

Taking a more pragmatic point of view, there may be domains where the existence of robust solutions according to our definition is really seldom. In such a case, it could be interesting to look for *most* robust solutions, by taking into account the number of breakages that can be actually repaired, instead of looking for a completely robust solution (similarly to what is done with the *partial fault tolerance* in [7]). This could be done, for instance, by turning the mandatory clauses corresponding to the existence of a repair for each possible breakage into weighted clauses. With this relaxation, a solution to the reformulated formula will always exist (as long as the initial formula is satisfiable), and by setting the appropriate weights to the repairability clauses, we could look for the most robust solution, i.e., a solution covering the highest number of breakages. In fact, with this method we could easily introduce the probability of failures (instead of the number of failures) by setting the weight of each reparation clause to be the probability of that particular breakage occurring, as well as balancing the trade-off between optimality and robustness. One could also think of adding the cost of repairs (instead of counting them) [13], which would require only slight changes to the reformulation we have presented.

Finally, we believe that our notion of robustness for weighted Max-SAT can be adapted to a notion of robustness for Max-Constraint Satisfaction Problems [1]. We also leave as further work this issue.

We have provided some preliminary but promising experimental results showing the feasibility of our approach. The results obtained are quite successful, especially if we consider them in relation with other works on robustness. As far as we know, there are very few results on reformulation approaches, and they are restricted to (1,0)- and (1,1)-supermodels [6]. Regarding results on search-based approaches, they are also restricted to find at most (1,3)-super solutions to CSP problems [7]. Although extensive experimentation with harder instances and other kind of problems than that of auctions, for instance the job-shop problem, should be done in order to better assess the scalability of our approach, we think that achieving a reasonable performance for finding up to (3,8)-supermodels as we do is a very promising result.

As noted in Section 2, PBO instances can be translated into partial weighted Max-SAT formulae. Hence, it would also be interesting to extend our performance comparisons by using solvers taking this approach, such as, e.g., MINISAT+ [5].

Acknowledgments

We thank Carlos Ansótegui and Víctor Muñoz for their helpful comments on this work.

References

[1] 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*, pages 1–14. Springer LNCS 4996, 2008.

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

[3] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[4] P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006.

[5] N. Eén and N. Sörensson. Translating pseudo-boolean constraints into SAT. *J. on Satisfiability, Boolean Modeling and Computation*, 2(1-4):1–26, 2006.

[6] M. L. Ginsberg, A. J. Parkes, and A. Roy. Supermodels and robustness. In *Proc. of AAAI'98*, pages 334–339, 1998.

[7] E. Hebrard. *Robust Solutions for Constraint Satisfaction and Optimisation under Uncertainty*. PhD thesis, University of New South Wales, 2006.

[8] E. Hebrard, B. Hnich, B. O'Sullivan, and T. Walsh. Finding diverse and similar solutions in constraint programming. In *AAAI'05: Proceedings of the 20th national conference on Artificial intelligence*, pages 372–377. AAAI Press, 2005.

[9] E. Hebrard, B. Hnich, and T. Walsh. Robust solutions for constraint satisfaction and optimization. In *Proc. of ECAI*, pages 186–190, 2004.

[10] E. Hebrard, B. Hnich, and T. Walsh. Super solutions in constraint programming. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 157–172. Springer LNCS 3011, 2004.

[11] E. Hebrard, B. Hnich, and T. Walsh. Improved algorithm for finding (a, b)-super solutions. In *Proc. of Workshop on Constraint Programming for Planning and Scheduling*, pages 236–248, 2005.

[12] A. Holland and B. O'Sullivan. Robust solutions for combinatorial auctions. In *ACM Conf. on Electronic Commerce*, 2005.

[13] A. Holland and B. O'Sullivan. Weighted super solutions for constraint programs. In *Proc. of AAAI'05*, pages 378–383, 2005.

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

[15] K. Leyton-Brown, M. Pearson, and Y. Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proc. of ACM Conference on Electronic Commerce*, pages 66–76, 2000.

[16] C. M. Li and F. Manyà. *Handbook of Satisfiability*, chapter MaxSAT, Hard and Soft Constraints, pages 613–631. 2009.

[17] L. Liu and M. Truszczynski. Satisfiability testing of boolean combinations of pseudo-boolean constraints using local-search techniques. *Constraints*, 12(3):345–369, 2007.

[18] V. M. Manquinho and J. Marques-Silva. Effective lower bounding techniques for pseudo-boolean optimization. In *Proc. of the conference on Design, Automation and Test in Europe*, pages 660–665, Washington, DC, USA, 2005. IEEE Computer Society.

[19] A. Roy. Fault tolerant boolean satisfiability. *J. Artificial Intelligence Research*, 25:503–527, 2006.

[20] R. Weigel and C. Bliek. On reformulation of constraint satisfaction problems. In *Proc. of ECAI*, pages 254–258, 1998.