Noname manuscript No.
(will be inserted by the editor)

Reformulation based MaxSAT robustness

Miquel Bofill · Dídac Busquets · Víctor Muñoz · Mateu Villaret

the date of receipt and acceptance should be inserted later

Abstract The use of SAT and MaxSAT encodings for solving constraint satisfaction problems (CSP) has been gaining wide acceptance in the last years. On the other hand, the presence of uncertainty in the real world makes robustness to be a desired property of solutions to CSPs. Roughly speaking, a solution is robust if it can be easily repaired when unexpected events happen. In fact, 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 a reformulation approach, since reformulation tends to generate prohibitively large formulas, especially in the CP setting.

In this paper we consider the unaddressed problem of robustness in weighted MaxSAT, by showing how robust solutions to weighted MaxSAT instances can be effectively obtained via reformulation into pseudo-Boolean formulae. Our encoding provides a reasonable balance between increase in size and performance, as shown by our experiments in the robust resource allocation framework. We also address the problem of flexible robustness, where some of the breakages may be left unrepaired if a totally robust solution does not exist.

In conclusion, we provide an easy-to-implement new method for achieving robustness in combinatorial optimization problems.

This is an extended version of the work [7] presented at the 12th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming (PPDP 2010) in Hagenberg, Austria.

Miquel Bofill · Mateu Villaret
Departament d'Informàtica i Matemàtica Aplicada, Universitat de Girona, Spain
E-mail: {mbofill,villaret}@ima.udg.edu

Dídac Busquets
Department of Electrical and Electronic Engineering, Imperial College London, UK
E-mail: didac.busquets@imperial.ac.uk

Víctor Muñoz
Newronia SL, Parc Científic i Tecnològic, Universitat de Girona, Spain
E-mail: victor.munoz@newronia.com

1 Introduction

Uncertainty is inherent to most real world problems. For instance, in job-shop scheduling (where tasks have to be distributed amongst appropriate machines while minimizing the overall makespan), if a machine breaks down, a new solution must be computed. Such a new solution should be fast to compute and, ideally, should also be close to the initial one (e.g., in the job-shop problem, it is apparent that 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 (where *easy* refers both to time and number of repairs). 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 was not to be considered. This fact has been sometimes called *the price of robustness* [6] but, in many real world situations, it is worth sacrificing some optimality for a stronger solution.

On the other hand, MaxSAT is becoming a competitive approach for solving combinatorial optimization problems [25] in the CP framework, and also to deal with Max-Constraint Satisfaction Problems (Max-CSP) [2]. For this reason, in this paper we consider the unaddressed problem of solution robustness in weighted MaxSAT. We illustrate its application to resource allocation problems.

There are existing works on solution robustness for (plain) SAT [11,31] and CP [14,15,16,17,21]. 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 [11]. On the other hand, search-based algorithms look for a robust solution with backtracking, propagation and consistency techniques [14,15,16,17, 21]. Previous works [13] claim that, at least for CP, the reformulation approach results in prohibitively large formulas.

In this paper we show that reformulation is still feasible in the setting of weighted MaxSAT. This has several advantages. First of all, the notion of robustness can be directly expressed in the original formulation of the problem with no need of changing the underlying solving method. Additionally, our reformulation can be easily adapted to interesting extensions of the notion of robustness. For instance, we can easily add dependencies between breakable and repairable variables, introduce failure probabilities, or allow partial repairability, among others. Contrarily to this, in search-based approaches, if the notion of robustness is modified, the search-based algorithm must probably be modified too.

Generalizing the notion of robust solution for plain SAT of Ginsberg et al. [11], we define a robust solution to a weighted MaxSAT 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 MaxSAT we are interested not only in keeping the number of changes low, but also in the cost of the solutions. Hence, while a is limiting the scope of the breakages, b is limiting the distance to the initial solution and β is limiting sub-optimality (this is an addition to the definition in [11]). Moreover, as can be seen in Section 4, we

have further generalized the 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 encoding in [11] for SAT by using pseudo-Boolean cardinality constraints. With our approach, the increase in size of the reformulated (robust) formula is $\mathcal{O}(n^a)$ instead of $\mathcal{O}(n^{a+b})$ (as it is the case in [11]), 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, most 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.

Actually, as shown in Section 6, we have done satisfactory experiments with a up to 2 and b up to 8 in some resource allocation problems. This is far better from previous results in the CP setting. However, it must be taken into account that a change in the value of a variable in the CP setting can correspond to a change in the value of more than one variable in the Boolean setting. For instance, a finite domain variable x with range $1..n$ (e.g., indicating which machine is assigned to task at a certain time) can possibly translate into n Boolean variables $x_1, \ldots, x_n$, where $x_i = \mathit{true}$ iff $x = i$ (in what is called a direct encoding into SAT in [33]). Then a change in the value of x from i to j implies changing two variables at the Boolean level: x_i from *true* to *false* and x_j from *false* to *true*.

Another important 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 is 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.

Finally, we generalize our definitions and encodings so that partially robust solutions can be obtained for problem instances lacking totally robust solutions.

The rest of the paper is organized as follows. In Section 2 we review some basic concepts and notation on Boolean satisfiability. In Section 3 we discuss on related work. Section 4 is the main section of the paper, including our definitions and encodings to obtain robust solutions to weighted MaxSAT problems, together with the corresponding correctness proofs. In Section 5 we address flexible robustness, i.e., the problem of finding partially robust solutions. Section 6 is devoted to empirical evaluation. Finally, in Section 7 we conclude.

2 Preliminaries

In this section we recall basic concepts on the Boolean satisfiability problem.

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 $\{\mathit{true}, \mathit{false}\}$. 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. *MaxSAT* [25] is the problem of finding the maximum number of clauses that can be satisfied by any truth assignment. *Weighted MaxSAT* is a variant of MaxSAT where every clause has a weight, i.e., a weighted MaxSAT 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 MaxSAT is the problem of finding a truth assignment with minimal cost. A particular case of weighted MaxSAT is *partial weighted MaxSAT* 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 MaxSAT is pseudo-Boolean optimization [30]. *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 MaxSAT formula, where each PB-constraint is translated into a set of mandatory clauses [9] and the objective function is translated into a set of non-mandatory clauses, each summand $C_j x_j$ becoming a unit clause $(\overline{x}_j, C_j)$.

Weighted Boolean Optimization (WBO) is a recently proposed framework extending both partial weighted MaxSAT and PBO, for which algorithms inspired in unsatisfiability-based algorithms for MaxSAT have been developed with very promising results [27]. In this framework, a formula is composed of two sets of pseudo-Boolean constraints, one consisting of mandatory (hard) constraints and the other consisting of non-mandatory (soft) constraints with an associated cost of falsification. Notice that this is a generalization of weighted partial MaxSAT by allowing the use of pseudo-Boolean constraints instead of just propositional clauses. Moreover, any PBO instance can be linearly encoded into WBO, by encoding the pseudo-Boolean constraints into hard constraints and the objective function into a set of unary soft constraints, as explained above.

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 Related Work

Here we briefly review the existing approaches to robustness for propositional logic (supermodels) and for constraint programming (super solutions).

3.1 Supermodels

The seminal work on robust solutions for propositional logic formulae is the one from Ginsberg et al. [11], where the notion of *supermodel* is introduced. The complexity for finding such supermodels in several propositional logic fragments has been studied in [31].

Definition 1 (Ginsberg et al. [11]) An (S_1^a, S_2^b)-*supermodel* of a Boolean formula F is a 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).

An (S_1^a, S_2^b)-supermodel in which the breakage and the repair set are unrestricted is referred to as an (a, b)-supermodel.

The task of finding (a, b)-supermodels is NP-complete in general. The main idea is to encode the supermodel requirements of a formula F as a new formula F_{SM} whose size is polynomially bounded by the size of F. This new formula F_{SM} has a model if and only if F has an (a, b)-supermodel. More concretely, the formula F_{SM} for finding an (S_1^a, S_2^b)-supermodel of a formula F is defined as follows[1]:

$$F_{SM} = F \wedge \bigwedge_{S \subseteq S_1,\, 1 \leq |S| \leq a} \left(\bigvee_{T \subseteq (S_2 \setminus S), |T| \leq b} C_{\overline{S \cup T}} \right) \tag{1}$$

For instance, the formula $F = p \vee q$ has three models, $\{p, q\}$, $\{\overline{p}, q\}$ and $\{p, \overline{q}\}$, which are all $(1, 1)$-supermodels. The encoding F_{SM} for a $(1, 1)$-supermodel of F, according to [11] and using Equation 1, would be

$$F_{SM} = \overbrace{(p \vee q)}^{\text{original } F} \wedge \overbrace{\left(\overbrace{(\overline{p} \vee q)}^{\text{no repair } (T=\emptyset)} \vee \overbrace{(\overline{p} \vee \overline{q})}^{\text{repair } q\ (T=\{q\})} \right)}^{\text{break in } p\ (S=\{p\})} \wedge \overbrace{\left(\overbrace{(p \vee \overline{q})}^{\text{no repair } (T=\emptyset)} \vee \overbrace{(\overline{p} \vee \overline{q})}^{\text{repair } p\ (T=\{p\})} \right)}^{\text{break in } q\ (S=\{q\})}$$

[1] Note that the formula is defined with the notation that will be used throughout the paper, and not the original notation of Ginsberg's et al. paper [11]

Note that, for instance, if the satisfying interpretation (model) chosen for F_{SM} is $\{\overline{p}, q\}$, i.e., $\{p = \mathit{false}, q = \mathit{true}\}$, then $p \vee q$ is satisfied and, moreover, if q switches to *false*, then a new model for $p \vee q$ can be obtained by switching p to *true*. That is, a break in q has a repair in p. The key idea is that the value of the sub-formula $\overline{p} \vee \overline{q}$ under the initial interpretation coincides with the value of $p \vee q$ under the repaired interpretation and, hence, F_{SM} has a model if and only if F has a supermodel. Note also that only the first model $\{p, q\}$ is a $(1, 0)$-supermodel.

3.2 Super Solutions

The concept of (a, b)-supermodel for propositional logic is generalized to that of (a, b)-*super solution* in the context of constraint programming (CP) in [16]. An (a, b)-*super solution* is one in which if at most a variables lose their values, the solution can be repaired by assigning these variables with new values and also changing the values of at most b other variables. The new values taken by the variables can be any other in their respective domains.

The generalization presented in [16] mainly focuses on $(1, 0)$-super solutions, since finding super solutions for values of a higher than 1 is very hard, and provides two alternative approaches for finding such robust solutions: either via reformulation or via search. The search approach uses an ad-hoc branch-and-bound algorithm to look for repairable solutions. In the reformulation approach, called $P + P$, variables are duplicated (with the same domain as the original ones) and some constraints are added. In the case of $(1, 0)$-super solutions, the new variables must be linked to original ones by the same constraints as their original counterpart. Additional constraints are added between each original variable and its duplicate so that they cannot have the same value. Figure 1 shows an example of reformulation to obtain a $(1, 0)$-super solution in a problem with three variables a, b and c and two restrictions r_1 and r_2. Notice that, e.g., both a and a' are linked to b by the same constraint r_1, while a and a' are restricted to be different. Thus, given a solution, the assignment to the original variables is a super solution where the repair is given by the value of the duplicated variables.

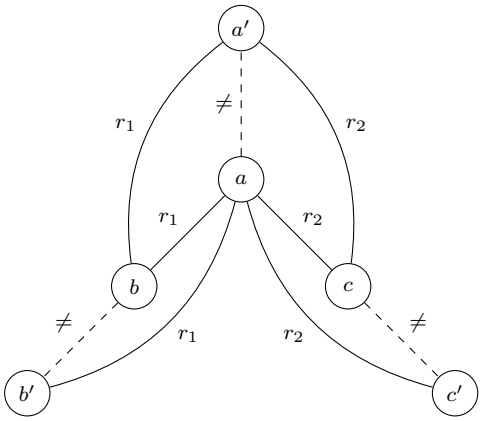

Fig. 1 Example of (1,0)-super solution reformulation for CP.

This work also addresses the problem of finding the "most robust solution" for the cases in which a robust solution does not exist. The most robust solution is a solution that maximizes the number of repairable variables. In a later paper [15] the same authors also address the problem of finding the "most robust optimal solution" and the "optimal robust solution". In Section 5 we also generalize our robustness notion considering this kind of flexibility.

For certain combinatorial problems super solutions are not expressive enough because they only consider the number of repairs. For this reason, the (α, β)-*weighted super solutions* [21] were developed, replacing the parameters a and b of super solutions with the breakage probability (α) and the cost of repair (β). Weighted super solutions have been mainly applied to combinatorial auctions [20].

4 Weighted MaxSAT Robustness

The following definition generalizes the one of Ginsberg et al. [11] to partial weighted MaxSAT.

Definition 2 Let F be a partial weighted MaxSAT 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 repairs. 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.

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 $I_1(p) = \mathit{true}$ and all other variables are assigned to *false* is a model of F with cost 2, but not 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 $I_2(p) = \mathit{true}$, $I_2(s) = \mathit{true}$ 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.

4.1 Encoding Weighted MaxSAT 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 MaxSAT 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:

$$\begin{aligned} F_{SM} = \ & \textbf{Minimize} && B \\ & \textbf{Subject To} && C \wedge (B \leq \beta) \wedge \\ & && \bigwedge_{S \subseteq S_1,\, 1 \leq |S| \leq a} \left(\bigvee_{T \subseteq (S_2 \setminus S) \cup S_3,\, |T \cap S_2| \leq b} C_{\overline{S \cup T}} \wedge (B_{\overline{S \cup T}} \leq \beta) \right) \end{aligned} \tag{2}$$

Alternatively, we could think of obtaining a WBO instance by directly using the set W of weighted clauses instead of the objective function B.

Notice that F_{SM} is not a set of PB-constraints, but a conjunction of disjunctions of PB-constraints. As noted in [26], 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

[2] Notice that any MaxSAT 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. However, it must be taken into account that the addition of fresh variables can impact the performance of the solver.

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 MaxSAT, 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 sub-formula $(C_{\overline{S \cup T}} \wedge (B_{\overline{S \cup T}} \leq \beta))$ enforces 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 4.2 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:

$$
\begin{aligned}
F_{SM} = \textbf{Minimize} \quad & \overline{p} + 2\overline{q} + 3r \\
\textbf{Subject To} \quad & (p \vee q) \wedge (\overline{q} \vee r) \wedge (\overline{q} \vee s) \wedge \\
& (\overline{p} + 2\overline{q} + 3r \leq 4) \wedge \\
\left.\begin{array}{l}\text{Break on } p\end{array}\right\{ & \left.\begin{array}{ll}
\text{No repair} & \Big(\Big((\overline{p} \vee q) \wedge (\overline{q} \vee r) \wedge (\overline{q} \vee s) \wedge \\
& \quad (p + 2\overline{q} + 3r \leq 4) \Big) \vee \\
\text{Repair } q & \Big((\overline{p} \vee \overline{q}) \wedge (q \vee r) \wedge (q \vee s) \wedge \\
& \quad (p + 2q + 3r \leq 4) \Big) \vee \\
\text{Repair } r & \Big((\overline{p} \vee q) \wedge (\overline{q} \vee \overline{r}) \wedge (\overline{q} \vee s) \wedge \\
& \quad (p + 2\overline{q} + 3\overline{r} \leq 4) \Big) \vee \\
\text{Repair } q, r & \Big((\overline{p} \vee \overline{q}) \wedge (q \vee \overline{r}) \wedge (q \vee s) \wedge \\
& \quad (p + 2q + 3\overline{r} \leq 4) \Big) \Big) \wedge
\end{array}\right. \\
\left.\begin{array}{l}\text{Break on } q\end{array}\right\{ & \left.\begin{array}{ll}
\text{No repair} & \Big(\Big((p \vee \overline{q}) \wedge (q \vee r) \wedge (q \vee s) \wedge \\
& \quad (\overline{p} + 2q + 3r \leq 4) \Big) \vee \\
\text{Repair } p & \Big((\overline{p} \vee \overline{q}) \wedge (q \vee r) \wedge (q \vee s) \wedge \\
& \quad (p + 2q + 3r \leq 4) \Big) \vee \\
\text{Repair } r & \Big((p \vee \overline{q}) \wedge (q \vee \overline{r}) \wedge (q \vee s) \wedge \\
& \quad (\overline{p} + 2q + 3\overline{r} \leq 4) \Big) \vee \\
\text{Repair } p, r & \Big((\overline{p} \vee \overline{q}) \wedge (q \vee \overline{r}) \wedge (q \vee s) \wedge \\
& \quad (p + 2q + 3\overline{r} \leq 4) \Big) \Big)
\end{array}\right.
\end{aligned}
$$

As noted in Example 1, a $(S_1^1, S_2^1, S_3, 4)$-supermodel of formula F is $\{p, \overline{q}, \overline{r}, s\}$. This interpretation satisfies 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)[3], and also the third set of clauses of the last block (a break on q with a repair on r).

The correctness of the F_{SM} reformulation can be easily proved. In order to ease the reading, we only give an sketch of the proof. Detailed proofs can be found in the appendix.

Definition 3 *(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.

[3] Recall that r belongs to the set of *don't-care variables* (S_3) and therefore its repair is not bounded by b.

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 $F_{\overline{S}}$.*

The next theorem follows the spirit of the result of Ginsberg et al. [11], but adding costs and *don't-care* variables.

Theorem 1 *Let F be a partial weighted MaxSAT 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 See Appendix. Sketch: 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 F_{SM}. To conclude we 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. Looking at the structure of the F_{SM} formula we can see that for every possible breakage (big *and*) there is at least a possible repair (big *or*), by flipping the necessary variables. Finally, we conclude using Lemma 1.

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 show that every such an interpretation I is a solution of F_{SM}. Since the solution has cost at most β, we have that $I \models B \leq \beta$. On the other hand, using Lemma 1, we have that $I \models C_{\overline{S \cup T}} \wedge (B_{\overline{S \cup T}} \leq \beta)$ for every possible breakage S and its repair T, and hence $I \models \wedge_{S \subseteq S_1, 1 \leq |S| \leq a} (\vee_{T \subseteq (S_2 \setminus S) \cup S_3,\, |T \cap S_2| \leq b} (C_{\overline{S \cup T}} \wedge (B_{\overline{S \cup T}} \leq \beta)))$. □

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 MaxSAT 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.*

4.2 Robust MaxSAT with Cardinality Constraints

Boolean *cardinality constraints* allow us to specify for a given set of Boolean variables E and a given number p, that at most p variables of E can be set to true [30].

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 (in size) 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 most 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.

The key idea in order to get rid of the disjunctions of the previous encoding of F_{SM} is the following: instead of encoding the different repairs by explicitly flipping the variables, simply rename the variables and restrict the number of variables that can change their value by means of cardinality constraints. As we need a different repair for each possible breakage, a different renaming of the repair (and don't-care) variables is necessary for each possible breakage. For this reason, we introduce a renaming function based on labeling every variable of a repair set R with the name of the breakage set S which is repairing.

Definition 4 *(Variable renaming).* Let R and S be sets of variables. The function $\mathsf{ren}_{R,S} : \mathcal{X} \to \mathcal{X}$ is defined as $\mathsf{ren}_{R,S}(x) = x^S$ for every variable $x \in R$, where x^S is a new atom, and $\mathsf{ren}_{R,S}(x) = x$ if $x \notin R$. When S is irrelevant, we simply write ren_R.

Definition 5 *(Formula renaming).* Let F be a Boolean (or pseudo-Boolean) formula, and R and S be sets of variables. Then $F^{\mathsf{ren}_{R,S}}$ denotes the formula F where all occurrences of each variable x have been replaced by $\mathsf{ren}_{R,S}(x)$.

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

Definition 6 *(Difference cardinality).* Let R and S be sets of variables, and $\mathsf{ren}_{R,S}$ be a variable renaming function. Then we define the *difference cardinality formula* as

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

Observe that we are using Boolean values as 0-1 integers in the definition of $\nabla^{\mathsf{ren}_{R,S}}$. Hence, if necessary, $\nabla^{\mathsf{ren}_{R,S}}$ 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:

$$\begin{array}{lll} F^{\nabla}_{SM} = & \textbf{Minimize} & B \\ & \textbf{Subject To} & C \wedge (B \leq \beta) \wedge \\ & \bigwedge\limits_{S \subseteq S_1,\, 1 \leq |S| \leq a} & \Big(C_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} \wedge \\ & & B_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} \leq \beta \wedge \\ & & \nabla^{\mathsf{ren}_{S_2 \setminus S,\, S}} \leq b \Big) \end{array} \tag{3}$$

Notice that $\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}$ is a renaming of the variables in $(S_2 \setminus S) \cup S_3$, by labeling them with S. As said, since a different renaming is needed for every

considered subset S of S_1, we just choose that set S for the renaming, as it improves readability.

Example 4 Continuing Example 2, we would get the following PBO instance when making use of renamings and cardinality constraints:

$$\begin{aligned} F_{SM}^{\nabla} = \textbf{Minimize} \quad & \overline{p} + 2\overline{q} + 3r \\ \textbf{Subject To} \quad & (p \vee q) \wedge (\overline{q} \vee r) \wedge (\overline{q} \vee s) \wedge \\ & (\overline{p} + 2\overline{q} + 3r \leq 4) \wedge \\ & (\overline{p} \vee q^{\{p\}}) \wedge (\overline{q}^{\{p\}} \vee r^{\{p\}}) \wedge (\overline{q}^{\{p\}} \vee s) \wedge \\ & (p + 2\overline{q}^{\{p\}} + 3r^{\{p\}} \leq 4) \wedge (q \neq q^{\{p\}} \leq 1) \wedge \\ & (p^{\{q\}} \vee \overline{q}) \wedge (q \vee r^{\{q\}}) \wedge (q \vee s) \wedge \\ & (\overline{p}^{\{q\}} + 2q + 3r^{\{q\}} \leq 4) \wedge (p \neq p^{\{q\}} \leq 1) \end{aligned}$$

Notice that flipping of variables is still used to denote each possible breakage. However, the repairs are encoded using cardinality constraints on renamed variables. The don't-care variables have also been renamed according to each breakage.

One solution for F_{SM}^{∇} is $\{p, \overline{q}, \overline{r}, s, q^{\{p\}}, r^{\{p\}}, p^{\{q\}}, r^{\{q\}}\}$. That is, as already noted 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., $p = p^{\{q\}}$).

The correctness of the F_{SM}^{∇} encoding for finding supermodels of a partial weighted MaxSAT formula F follows easily from the correctness of the F_{SM} encoding. Detailed proofs can be found in the appendix.

The following is an auxiliary lemma stating that a model of a formula, after renaming some of its variables, is a model of the original formula after flipping the variables which occur with different polarity than its renamed counterpart in the model.

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 $I \models F^{\mathsf{ren}_R}$ if and only if $I \models F_{\overline{D}}$, where $D = \{x \in Var(F) \mid I(x) \neq I(\mathsf{ren}_R(x))\}$.*

The next lemma is the main ingredient needed for proving the correctness of the F_{SM}^{∇} formulation. It states how certain disjunctions of formulas, which only differ in the polarity of some literals, can be encoded by means of cardinality constraints.

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 $(S_1 \cup S_2) \cap S_3 = \emptyset$, let a and b be non-negative integer numbers, and I be an interpretation. Then*

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

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

$$I' \models \bigwedge_{S \subseteq S_1,\, 1 \leq |S| \leq a} \left(F_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} \wedge (\nabla^{\mathsf{ren}_{S_2 \setminus S,\, S}} \leq b) \right)$$

Proof See Appendix. Sketch: For the left-to-right implication, for every set $S \subseteq S_1$ we identify a set $T \subseteq (S_2 \setminus S) \cup S_3$ such that $I \models F_{\overline{S \cup T}}$. Then we construct an interpretation I^S such that $I^S \models F^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}}_{\overline{S}}$, using Lemma 2. Concerning $(\nabla^{\mathsf{ren}_{S_2 \setminus S,\, S}} \leq b)$, by Definition 6 we have $I^S(\nabla^{\mathsf{ren}_{S_2 \setminus S,\, S}}) = |T \cap (S_2 \setminus S)| \leq |T \cap S_2| \leq b$, as we take I^S such that $I^S(x) = I(x)$ for all $x \in \mathit{Var}(F)$ and $I^S(x^S) \neq I(x)$ if and only if $x \in T$ for all $x^S \in \mathit{Var}(F^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}})$. Finally, we conclude that there exists an interpretation I' that is compatible with all I^S.

For the right-to-left implication, 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 have that $I' \models (\nabla^{\mathsf{ren}_{S_2 \setminus S,\, S}} \leq b)$ implies $\sum_{x \in (S_2 \setminus S)} (I'(x^S) \neq I'(x)) \leq b$ and, on the other hand, $\sum_{x \in (S_2 \setminus S)} (I'(x^S) \neq I'(x)) = |T \cap (S_2 \setminus S)|$. Since S and S_3 are disjoint, we have that $T \cap (S_2 \setminus S) = T \cap S_2$, and thus $|T \cap S_2| \leq b$. Finally, by Lemma 2, $I' \models F^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}}_{\overline{S}}$ implies $I' \models F_{\overline{S \cup T}}$ and, since $I'(x) = I(x)$ for all $x \in \mathit{Var}(F)$, we conclude that $I \models F_{\overline{S \cup T}}$. □

Theorem 2 *Let F be a partial weighted MaxSAT 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 $(S_1^a, S_2^b, S_3, \beta)$-supermodel if and only if the PBO instance F^{∇}_{SM} has a solution.*

Proof See appendix. Sketch: It follows from Theorem 1 and Lemma 3. □

Corollary 2 *Let F be a partial weighted MaxSAT 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.*

In this section we have proved that finding supermodels for partial weighted MaxSAT formulae F amounts to solving the corresponding PBO instances F^{∇}_{SM}. It is very important to realize that solutions of F^{∇}_{SM} allow us to immediately find an appropriate repair for every possible breakage, i.e., a solution of F^{∇}_{SM} not only includes a solution of F that can be repaired, but a repair for every possible breakage in the solution of F. Given a solution I of F^{∇}_{SM} (and hence of F) and a breakage involving a set of variables S, a repair for S (set of variables that must flip their value in order to get a new solution of F) is the set $\{x \in \mathit{Var}(F) \mid I(x) \neq I(x^S)\}$. For instance, in Example 4, a model I for F^{∇}_{SM} is $\{p, \overline{q}, \overline{r}, s, q^{\{p\}}, r^{\{p\}}, p^{\{q\}}, r^{\{q\}}\}$, i.e., $I(p) = \mathit{true}$, $I(q) = \mathit{false}$, $I(r) = \mathit{false}$, $I(s) = \mathit{true}$, $I(q^{\{p\}}) = \mathit{true}$, $I(r^{\{p\}}) = \mathit{true}$, $I(p^{\{q\}}) = \mathit{true}$ and $I(r^{\{q\}}) = \mathit{true}$. 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 $\{q, r\}$, since $I(q) \neq I(q^{\{p\}})$ and $I(r) \neq I(r^{\{p\}})$. If the breakage is $\{q\}$ then the set of variables marked with $\{q\}$ that must change their value is $\{r\}$, since $I(r) \neq I(r^{\{q\}})$; p must not change its value since p and $p^{\{q\}}$ occur with the same polarity in the solution, that is, $I(p) = I(p^{\{q\}})$. This is especially important since, as said in the introduction, we are looking for a solution that can be easily repaired, where easy refers both to number of repairs and time.

5 Flexible Robustness

The encodings presented in Section 4 force all solutions to be repairable, i.e., if a single breakage is not repairable then the formula is not satisfiable, and so there

is no robust solution. Although having a completely robust solution would be the ideal situation, it may not be always possible to find it. This fact can be observed in the experiments of the previous section, where the percentage of solved instances in some cases is very low (in particular when a high percentage of optimality is required, or when the parameter a is set to be greater or equal to 2). Therefore, in order to be able to obtain some amount of robustness in hard instances, we add the possibility of allowing some of the breakages to be left unrepaired, in what we call *flexible robustness*.

The concept of flexibility has also been tackled in [16]. Although that paper focuses on defining super solutions for constraint programming, it also points out a notion of flexibility by defining a "most robust solution" for the cases where a robust solution does not exist. In that setting, a most robust solution is defined as a solution that maximizes the number of repairable variables. In a later paper [15] the same authors addressed the balance between optimality and robustness by searching for a "most robust optimal solution" (an optimal solution whose number of repairable variables is maximal) and an "optimal robust solution" (a super solution with the best, although possibly suboptimal, objective function). This work is pretty close to ours, but it is based on search instead of on reformulation and, moreover, does not consider the cost of the repair but only the total number of variables changing their value.

Notice that we are already seeking for optimal robust solutions. Hence, we propose the following two variants:

- *Most robust optimal solutions.* In this setting we are concerned with finding, from all optimal solutions, the one which is most robust (like in [15]).
- *Optimal most robust solutions.* Here we look, among the solutions that have the maximum possible level of robustness, for the one with the minimal cost.

Our technique for adding flexibility is based on *soft constraints* [29]. Soft constraints provide a way to model preferences over constraints, so that some constraints can be violated for over-constrained problems, still such violation of constraints is avoided as far as possible. Alternatively, we could apply a Boolean Multilevel Optimization approach [4], since there is a hierarchy in the objectives (first optimality and then robustness, or vice-versa).

5.1 Encoding Most Robust Solutions

We define a *most robust solution* as a solution with the highest number of repairable breakages. In order to compute most robust solutions, we need to slightly modify our encoding so that breakages are not mandatorily repairable. The idea for doing that is to introduce a fresh Boolean variable r_S for each breakage S, and force S to be repairable only if r_S is true. Then, finding a most robust solution simply amounts to find a solution which maximizes the number of r_S variables that are set to true. This can also be seen as a constraint optimization problem, where constraints are weighted and the goal is to find a solution maximizing the weight of satisfied constraints. However, since our original problem already includes weighted constraints (the ones corresponding to the cost function), we have to be careful so that the new ones have lower cost if we want to give priority to optimality, and higher cost if we want to give priority to robustness.

Then, e.g., Formula 3 would be modified as follows:

$$\begin{aligned} F^{\nabla}_{SM} = \ & \textbf{Minimize} && B + R \\ & \textbf{Subject To} && C \wedge (B \leq \beta) \wedge \\ & \bigwedge_{S \subseteq S_1,\, 1 \leq |S| \leq a} && r_S \Rightarrow \Big(C_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} \wedge \\ & && \quad B_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} \leq \beta \wedge \\ & && \quad \nabla^{\mathsf{ren}_{S_2 \setminus S,\, S}} \leq b \Big) \end{aligned} \tag{4}$$

where

$$R = \sum_{S \subseteq S_1, 1 \leq |S| \leq a} \overline{r_S} \cdot w_S$$

being w_S the weight associated with breakage S.

Note that, with this encoding, the minimization of R implies the maximization of the weights of the repairable breakages: whenever $\overline{r_S}$ is 0, r_S is *true* and hence $C_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} \wedge (B_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} \leq \beta) \wedge (\nabla^{\mathsf{ren}_{S_2 \setminus S,\, S}} \leq b)$ must be *true*, meaning that the breakage S is repairable. However, since we are in fact minimizing $B+R$ (where $B = \sum_{j \in 1..k} \bar{l}_j \cdot w_j$ amounts to the sum of the weights of the unsatisfied clauses in the initial problem), as said, we have to be careful in choosing the weights w_S for the breakages. Depending on whether we are interested in an *optimal most robust solution* or in a *most robust optimal solution* (where optimality refers to the weight of unsatisfied clauses in the initial problem), we must respectively

- choose $w_S > \sum_{j \in 1..k} w_j$ for every breakage S, or either
- choose, e.g., $w_S = 1$ and multiply each w_j, $j \in 1..k$, by a number greater than the number of breakages $\#\{S \subseteq S_1 \mid 1 \leq |S| \leq a\}$, e.g., $|S_1|^{a+1}$.

This way, minimization of B and R do not interfere, i.e., we are giving priority to either optimality or robustness, but not to a mixture of both concepts, since each breakage has superior weight than all initial weighted clauses together, or each initial clause has superior weight than all breakages together. Moreover, by multiplying the weights of the initial clauses by a same constant, we are scaling their weights proportionally. Hence, an optimal solution to the initial problem will also be optimal when giving priority to optimality in presence of robustness.

5.1.1 Translation into Pseudo-Boolean Constraints

Formula 4 can be easily translated into a conjunction of pseudo-Boolean constraints, as needed by most pseudo-Boolean solvers. Notice that

$$r_S \Rightarrow \Big(C_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} \wedge (B_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} \leq \beta) \wedge (\nabla^{\mathsf{ren}_{S_2 \setminus S,\, S}} \leq b) \Big)$$

is equivalent to

$$\begin{aligned} & \Big(\overline{r_S} \vee C_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} \Big) \\ \wedge \ & \Big(\overline{r_S} \vee (B_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} \leq \beta) \Big) \\ \wedge \ & \left(\overline{r_S} \vee (\nabla^{\mathsf{ren}_{S_2 \setminus S,\, S}} \leq b) \right). \end{aligned}$$

Now, since $C_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}}$ is assumed to be a conjunction of clauses, by applying distributivity of $\vee$ over $\wedge$ the sub-formula

$$\overline{r_S} \vee C_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}}$$

can be rewritten into a conjunction of clauses and hence of pseudo-Boolean constraints as well.

For the rest of sub-formulas, we can make use of the generic *Big-M* transformation (see, e.g., [35]). This transformation is based on the fact that a disjunction $b \vee (x \leq 0)$, where b is a propositional variable, is equivalent to $x \leq \mathit{ubound}(x) \cdot b$, where *ubound* is an upper bound on the value of the variable x. Hence,

$$\begin{aligned} & \overline{r_S} \vee (B_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} \leq \beta) \\ = \; & \overline{r_S} \vee (B_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} - \beta \leq 0) \end{aligned}$$

can be equivalently rewritten into

$$B_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} - \beta \leq \mathit{ubound}(B_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} - \beta) \cdot \overline{r_S}$$

and similarly for $\overline{r_S} \vee (\nabla^{\mathsf{ren}_{S_2 \setminus S,\, S}} \leq b)$, obtaining equivalent pseudo-Boolean constraints.

For every set S, an upper bound of $B_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}}$ is the sum of the weights of the initial clauses $\sum_{j \in 1..k} w_j$, and an upper bound of $\nabla^{\mathsf{ren}_{S_2 \setminus S,\, S}}$ is $|S_2|$.

5.1.2 Probability of Failures

The flexibility we have added to the encoding allows us to easily deal with new interesting aspects such as probability of failures. It is likely that some breakages can have higher probability of occurring than others, and so they should have different weights. By simply multiplying the weights w_S of the breakages by its percent probability of occurring, the higher probability a breakage has, the higher probability of being repairable will have. Obviously, for irrelevant breakages we shall set $w_S = 0$.

6 Experimental evaluation

To demonstrate the feasibility and practicality of our approach for finding robust solutions to partial weighted MaxSAT formulae, we have considered the problem of resource allocation. In particular, we have casted the resource allocation problem as a *combinatorial auction* [8], since the problem of finding a solution to an auction is easily translated into a weighted MaxSAT formula. Moreover, combinatorial auctions are used as one of the benchmarks in the partial weighted MaxSAT category of the SAT competition [3].

In the next subsections we first formalize the problem of resource allocation as a combinatorial auction, then describe the experimental setup and the different solvers we have used, and finally provide experimental results.

6.1 Resource Allocation as a Combinatorial Auction

The problem of resource allocation is how to distribute a set of resources to a set of agents requesting them. Each agent may request several combinations of resources, and it must indicate a utility value for each of these combinations. The goal is to find such an allocation that (1) each resource is assigned to at most one agent (i.e. the resources are not shareable), and (2) the allocation maximizes the overall system utility (sum of each agent's utility, according to the resources it has been allocated).

The conversion of a resource allocation problem to a combinatorial auction is straightforward. The resources to be distributed are the items being offered in the auction. The agents requesting resources play the role of bidders, and their resources requests are translated to bids (see below). Finally, the decision maker agent (the one distributing the resources) plays the role of auctioneer.

The problem posed by a combinatorial auction is equivalent to several existing combinatorial problems, such as the matching problem, the set packing problem, or the multi-dimensional knapsack problem. 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:

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

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

We use the following encoding of a combinatorial auction as a partial weighted MaxSAT problem [22, 18]:

- 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[4]. 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

[4] Robustness against bid withdrawal has been considered in [20] using weighted super solutions

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 MaxSAT 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_1^a, S_2^b, 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 MaxSAT we seek to minimize the cost.

6.2 Experimental setup

We have used the Combinatorial Auctions Testsuite (CATS) [24] to generate several instances of auctions. CATS is a suite of distributions for modeling realistic bidding behavior. This suite is grounded in previous research on specific applications of combinatorial auctions. Each distribution defines the complementarity relationship[5] among the goods being auctioned. We have considered the following distributions:

- *Paths*: this distribution involves bidding on paths in space (e.g. truck routes, rights on railway tracks, etc.). For our experiments, we have used the default parameter setting with 30 goods (path segments) and 50 bids (distinct paths requested). See Table 4 for the results on this distribution.
- *Regions*: this distribution considers problems with complementarity from adjacency in two-dimensional space (e.g. sale of adjacent pieces of real estate). For

[5] This relationship establishes dependencies on bidding for sets of goods.

our experiments, we have used the default parameter setting[6] with 30 goods (regions) and 50 bids. See Table 2 for the results on this distribution.
- *Temporal matching*: in this distribution temporal complementarity between goods is taken into account (e.g. allocation of airport take-off and landing slots). For our experiments, we have used the default parameter setting with 28 goods[7] (slots) and 50 bids. See Table 3 for the results on this distribution.
- *Arbitrary relationships*: in this last distribution, the complementarity among goods is weaker than in the previous distributions, but it exhibits still some regularity. For our experiments, we have used the default parameter setting[8] with 30 goods and 50 bids. See Table 1 for the results on this distribution.

We should note that the number of bids has been set to low values since auctions with a large number of bids are inherently robust. This fact has been shown in [19,28]. In both works, a sensibility analysis is performed, assessing the effect of breakages in combinatorial auctions; the first work focuses on breakages caused by bid withdrawal, while in the second the breakages are caused by resource unavailability. The conclusions of both analysis are that when there are many bids, optimal solutions are inherently robust, since when a breakage occurs there are still many other bids with which an alternate solution can be formed, achieving the same optimal –or very close to optimal– benefit. However, when the number of bids is rather low, optimal solutions are greatly affected by breakages, both in the reduction of benefit for alternate solutions, as well as the number of changes needed to find such a repair.

We have considered several solvers and techniques to solve pseudo-Boolean problems selecting the most efficient solvers according to the results in the pseudo-Boolean competitions[9]. More concretely, the Weighted MaxSAT robustness encoding we propose falls into the OPT-SMALLINT-LIN category (optimization problems using small integer values and linear constraints) as well as in PARTIAL-SMALLINT-LIN category (weighted Boolean optimization with some hard clauses and using small integer values and linear constraints). Namely we have used the following solvers:

- SAT4J (v20120210): Pseudo-Boolean solver and optimizer, based on conflict driven clause learning. It allows to solve both Pseudo-Boolean and Weighted Boolean optimization problems [5].
- CLASP 2.0.5: Pseudo-Boolean solver based on answer sets [10].
- CPLEX 12.4 ILOG's state of the art integer linear programming solver [23].
- Gurobi 4.0: state-of-the-art solver for integer linear programming [12][10].

The experiments have been carried out on an Intel Core i5 @2.67GHz, running OpenSuSE 11.2, with kernel 2.6.31.14.

[6] We have also set the `addloc` parameter to 0.5. This parameter denotes the probability of adding a new region in a bid.

[7] In the temporal matching distribution, the number of goods must be a multiple of 4.

[8] We have also set the `addloc` parameter to 0.5. This parameter denotes the probability of adding a new good in a bid.

[9] `http://www.cril.univ-artois.fr/PB11`

[10] Although this solver has not participated in the pseudo-Boolean competitions, we have observed that it is faster than CPLEX in some instances. We have used version 4.0 since this gave much better results than version 4.6.

6.3 Experiments on Robustness in Resource Allocation

Tables 1-4 show the results of our experiments on robustness in resource allocation. For each distribution we have considered 1 or 2 breakages (a), $1, 4$ or 8 repairs (b), and required a minimum utility γ (which in turn defines the value of β) according to several percentages of the revenue of the (precalculated) optimal solution (%Opt) when robustness is not considered: {60%, 70%, 80%, 90%}. We have generated 20 instances for each (a,b,%Opt) configuration, resulting in a total of 1600 instances to be solved by each solver[11]. The timeout per instance has been set to 1000 seconds. Each table shows the percentage of satisfiability (%SAT) for each configuration[12], as well as the percentage of the optimal solutions given by the solver[13] that are actually robust (OR), and the optimality of the robust solutions found (%O). Moreover, for each solver we indicate the percentage of instances solved within the timeout (%Sol) and the average (median) solving time for the SAT (TS) and UNSAT (TU) instances. If the solver reached the timeout, the table shows *t.o.*

The results show that our approach is clearly feasible for one breakage and any number of repairs. When considering two breakages we have a significant increase on the average solving time. This was expected since the increase in size of the reformulated (robust) formula is $\mathcal{O}(n^a)$, being a the number of breakages. We can also observe that the pseudo-Boolean solver SAT4J is able to solve almost all instances (with just a few exceptions on the Arbitrary and Paths distributions). The linear programming solver Gurobi also performs very well but does not solve as many instances. Finally, CPLEX and CLASP exhibit worse results.

Concerning the hardness of the instances we can observe a certain phases transition phenomenon around instances $(2, 4/8, 70/80\%)$. When the percentage of optimality (%Opt) required is high, e.g. in configurations $(2, 4, 80/90\%)$ and $(2, 8, 80/90\%)$, instances are likely to be unsatisfiable as shown in the %SAT column. In particular, when asking for a 90% of optimality, all solvers find this unsatisfiability very quickly.

As for the robustness of optimal solutions, we can observe that the Matching and Paths distributions are inherently robust, where most of the optimal solutions found are already robust. On the other hand, for the Arbitrary and Regions distributions, the robustness of optimal solutions is directly related to the percentage of optimality required (Opt). It is in such cases that looking for robust solutions may be interesting. Moreover, the loss of optimality of the robust solutions (the price of robustness) is very low as shown in column %O.

6.4 On translating the PBO instances into partial weighted MaxSAT

As said in Section 2, every PBO instance can be translated into a partial weighted MaxSAT formula, where each PB-constraint is translated into a set of mandatory (hard) clauses and the objective function is translated into a set of non-mandatory (soft) clauses, each summand $C_j x_j$ becoming a unit clause $(\overline{x}_j, C_j)$.

Several techniques are implemented in MiniSat+ [9], including support for objective functions. For the PB-constraints, those techniques include conversion

[11] The configurations (2,1,-) have not been used, since they are always unsatisfiable

[12] We have run Gurobi without timeouts to obtain the actual satisfiability percentage.

[13] We have taken the first optimal solution found by Gurobi

					Gurobi			CPLEX			CLASP			SAT4J		
a	*b*	Opt	%SAT	%OR	%O	%S	TS	TU	%S	TS	TU	%S	TS	TU	%S	TS	TU
1	1	60	100	100	100	100	0.78	-	100	1.45	-	100	0.17	-	100	0.6	-
		70	90	100	100	100	0.76	12.46	100	1.43	16.96	100	0.21	0.1	100	0.66	0.52
		80	55	82	99	100	1.21	2.9	100	1.97	4.5	100	0.18	0.15	100	0.65	0.64
		90	0	0	0	100	-	0.91	100	-	2.08	100	-	0.16	100	-	0.6
1	4	60	100	100	100	100	0.78	-	100	1.51	-	100	0.42	-	100	0.61	-
		70	100	95	99	100	0.76	-	100	1.87	-	100	1.63	-	100	0.8	-
		80	85	65	98	100	1.7	17.34	100	3.03	17.25	100	4.16	1.91	100	1.49	0.87
		90	0	0	0	100	-	1.6	100	-	4.99	100	-	0.45	100	-	0.78
1	8	60	100	100	100	100	0.76	-	100	1.58	-	100	0.42	-	100	0.58	-
		70	100	100	100	100	0.73	-	100	1.92	-	100	3.46	-	100	0.75	-
		80	100	75	99	100	1.12	-	100	2.31	-	100	23	-	100	1.56	-
		90	15	0	96	100	10.26	0.84	100	9.52	2.3	100	65.78	0.2	100	3.64	0.83
2	4	60	95	74	99	85	41.1	*t.o.*	65	112.77	*t.o.*	65	331.01	*t.o.*	100	79.47	107.99
		70	65	38	94	50	33.66	*t.o.*	20	55.31	*t.o.*	75	501.79	916.68	100	378.21	59.45
		80	0	0	0	90	-	58.53	65	-	581.96	95	-	99.26	100	-	31.26
		90	0	0	0	100	-	19.47	100	-	127.28	100	-	3.15	100	-	6.6
2	8	60	100	90	100	100	33.34	-	70	385.85	*t.o.*	25	803.31	*t.o.*	100	21.12	-
		70	85	47	99	85	49.7	*t.o.*	30	471.81	*t.o.*	5	-	*t.o.*	100	97.9	7.07
		80	10	0	95	85	999.04	43.87	65	-	443.36	55	-	485.68	95	*t.o.*	15.68
		90	0	0	0	100	-	17.89	100	-	110.72	100	-	2.32	100	-	5.92

Table 1 Results for arbitrary distribution

					Gurobi			CPLEX			CLASP			SAT4J		
a	*b*	Opt	%SAT	%OR	%O	%S	TS	TU	%S	TS	TU	%S	TS	TU	%S	TS	TU
1	1	60	100	95	100	100	0.45	-	100	0.75	-	100	0.21	-	100	0.63	-
		70	100	90	99	100	0.43	-	100	0.9	-	100	0.21	-	100	0.62	-
		80	60	83	99	100	0.43	1.71	100	0.81	2.55	100	0.24	0.2	100	0.84	0.71
		90	0	0	0	100	-	0.75	100	-	1.59	100	-	0.15	100	-	0.62
1	4	60	100	95	100	100	0.46	-	100	0.94	-	100	0.34	-	100	0.6	-
		70	100	90	100	100	0.47	-	100	1.14	-	100	0.92	-	100	0.67	-
		80	90	83	100	100	0.54	7.03	100	1.27	5.01	100	1.46	1.23	100	1.09	0.92
		90	0	0	0	100	-	0.57	100	-	1.32	100	-	0.31	100	-	0.79
1	8	60	100	100	100	100	0.48	-	100	0.86	-	100	0.33	-	100	0.6	-
		70	100	90	100	100	0.47	-	100	1.04	-	100	1.12	-	100	0.64	-
		80	95	79	100	100	0.53	0.79	100	1.3	1.07	100	11.22	0.12	100	1.16	0.53
		90	10	0	100	100	0.93	0.47	100	1.96	1	100	91.17	0.16	100	1.9	0.68
2	4	60	100	85	99	100	17.2	-	100	30.26	-	90	108.5	*t.o.*	100	42.03	-
		70	80	63	98	85	19.22	*t.o.*	85	64.55	879.44	90	597.21	81.89	100	110.46	26.15
		80	0	0	0	95	-	19.41	90	-	209.97	90	-	31.18	100	-	18.53
		90	0	0	0	100	-	15.32	100	-	62.09	100	-	2.72	100	-	9.81
2	8	60	100	85	100	100	17.52	-	100	84.38	-	35	118.01	*t.o.*	100	8.97	-
		70	85	71	100	100	19.53	590.16	90	216.37	853.41	20	537.66	*t.o.*	100	50.04	7.41
		80	5	0	99	100	523.93	16.31	90	-	134.56	65	-	42.44	100	173.85	8.76
		90	0	0	0	100	-	15.16	100	-	44.22	95	-	2.13	100	-	5.54

Table 2 Results for regions distribution

into either a BDD, a network of adders or a network of sorters. Then, on top of this, any partial weighted MaxSAT algorithm can be implemented.

The nice thing here is that we are closing the loop, since we are encoding weighted MaxSAT robustness into weighted MaxSAT. Unfortunately with the experiments carried out we have observed that, for the instances of Section 6.2, the performance obtained with this approach is much worse than with pure pseudo-Boolean solvers.

					Gurobi			CPLEX			CLASP			SAT4J		
a	*b*	Opt	OR	%SAT	%O	%S	TS	TU	%S	TS	TU	%S	TS	TU	%S	TS	TU
1	1	60	100	95	100	100	0.41	0.53	100	0.89	0.95	100	0.91	0.09	100	1.24	0.6
		70	100	95	100	100	0.41	0.39	100	0.98	0.85	100	0.79	0.06	100	1.25	0.48
		80	90	50	100	100	0.45	0.86	100	1.07	1.2	100	1.04	1	100	1.65	1.42
		90	0	0	0	100	-	0.49	100	-	1.19	100	-	0.84	100	-	1.53
1	4	60	100	95	100	100	0.43	0.34	100	0.7	0.79	100	0.93	0.05	100	1.03	0.51
		70	100	95	100	100	0.43	0.35	100	1.01	0.6	100	2.65	0.05	100	1.07	0.46
		80	100	55	100	100	0.49	0.51	100	1.25	0.89	100	7.57	1.03	100	1.5	1.17
		90	0	0	0	100	-	0.44	100	-	0.93	100	-	1.41	100	-	1.49
1	8	60	100	95	100	100	0.42	0.33	100	0.66	0.7	100	1.19	0.07	100	0.89	0.5
		70	100	95	100	100	0.43	0.33	100	0.69	0.56	100	5.62	0.05	100	0.98	0.58
		80	100	60	100	100	0.45	0.51	100	1	0.82	95	33.24	1.61	100	2.05	1.2
		90	0	0	0	100	-	0.43	100	-	0.8	100	-	1.06	100	-	1.3
2	4	60	100	75	100	95	27.27	254.69	95	25.43	138.58	65	238.97	*t.o.*	100	23.3	17.74
		70	100	15	89	100	74.89	43.86	90	61.3	50.58	45	-	*t.o.*	100	361.66	22.78
		80	0	0	0	100	-	10.84	100	-	26.69	95	-	47.9	100	-	22.13
		90	0	0	0	100	-	9.96	100	-	12.89	100	-	14.46	100	-	16.66
2	8	60	100	75	100	100	21.55	247.72	100	200.86	57.51	30	221.67	*t.o.*	100	25.43	5.8
		70	100	25	100	100	28.84	35.42	95	346.49	33.45	40	-	*t.o.*	100	84.99	9.54
		80	0	0	0	100	-	10.68	100	-	43.01	80	-	29.62	100	-	10.18
		90	0	0	0	100	-	9.98	100	-	13.23	100	-	9.47	100	-	10.47

Table 3 Results for matching distribution

					Gurobi			CPLEX			CLASP			SAT4J		
a	*b*	Opt	%SAT	%OR	%O	%S	TS	TU	%S	TS	TU	%S	TS	TU	%S	TS	TU
1	1	60	100	100	100	100	0.49	-	100	1.25	-	100	0.49	-	100	0.97	-
		70	100	100	100	100	0.5	-	100	1.38	-	100	0.48	-	100	0.92	-
		80	100	100	100	100	0.51	-	100	1.21	-	100	0.56	-	100	0.96	-
		90	5	100	100	100	0.42	0.8	100	0.95	2.36	100	0.44	0.83	100	0.83	1.28
1	4	60	100	100	100	100	0.54	-	100	1.25	-	100	0.37	-	100	0.73	-
		70	100	100	100	100	0.56	-	100	1.43	-	100	0.57	-	100	0.78	-
		80	100	100	100	100	0.58	-	100	1.58	-	100	1.49	-	100	1.18	-
		90	15	67	97	100	1.31	0.72	100	3.29	1.75	100	10.66	1.7	100	1.77	1.75
1	8	60	100	100	100	100	0.53	-	100	1.15	-	100	0.38	-	100	0.75	-
		70	100	100	100	100	0.53	-	100	1.28	-	100	0.82	-	100	0.78	-
		80	100	100	100	100	0.57	-	100	1.34	-	100	7.32	-	100	1.47	-
		90	25	100	100	100	0.74	0.6	100	1.85	1.01	100	99.04	1.1	100	4.58	1.59
2	4	60	100	100	100	100	26.75	-	95	25.74	*t.o.*	100	136.7	-	100	7.61	-
		70	100	100	100	100	32.19	-	75	69.3	*t.o.*	95	322.79	1000	95	79.29	1004.65
		80	30	100	100	85	29.11	283.53	45	347.44	*t.o.*	60	426.67	481.88	95	620.59	142.05
		90	0	0	0	100	-	15.16	100	-	110.29	100	-	22.55	100	-	23.19
2	8	60	100	100	100	100	26.52	-	95	311.47	*t.o.*	90	223.91	1000	100	7.02	-
		70	100	100	100	100	29.12	-	55	812.49	*t.o.*	20	726.9	1000	100	95.09	-
		80	50	80	100	100	31.56	62.89	35	815.94	*t.o.*	25	-	1000	75	978.66	34.37
		90	0	0	0	100	-	15.01	100	-	83.04	85	-	24.94	100	-	12.7

Table 4 Results for paths distribution

We have carried out some experiments with MiniSat+. Also, since MiniSat+ outputs the generated set of hard clauses, we have been able to run some state-of-the-art weighted MaxSAT algorithms on top of these after adding the soft clauses corresponding to the objective function. In particular, we have run the weighted MaxSAT version of SAT4J [5], and the WPM1 algorithm [1]. In the best situation we obtained two orders of magnitude higher running times than the ones shown in Section 6.3. For this reason we omit the results of these experiments.

We have checked that the decisional version of our instances, i.e., the set of hard clauses, is already difficult for state-of-the-art SAT solvers. In particular, we have run the solver CryptoMiniSat [32], which is able to give a special treatment to the XOR clauses, which our encoding is plenty of. Nevertheless, this solver has shown very poor results.

6.5 Experiments on Flexible Robustness

In order to show the usefulness of flexible robustness, we have experimented with the resource allocation instances with worst satisfiability percentages, i.e., the matching distribution. For each instance of this distribution we have looked for the *most robust optimal solutions* and the *optimal most robust solutions*. We have used the solver Gurobi since it is one of the solvers exhibiting the best performances in the previous experiments. Table 5 shows the results of this experimentation. For each (a,b,%Opt) configuration and for each flexibility variant we provide the percentage of solved instances within the timeout (%Sol), which has been set to 1000 seconds, and the average solving time (Avg Time), as well as the percentage of optimality of the solution with respect to the corresponding non-robust problem (%O) and the percentage of repaired breakages (%R). To facilitate the comparison between strict and flexible robustness we also show the percentage of satisfiable instances when using strict robustness (%SAT), extracted from Table 3. Recall that with flexible robustness all instances are satisfiable, since we allow to leave any number of breaks unrepaired.

Observing the results for the *most robust optimal solution*, we see that all of them have a 100% of optimality, since by definition they are all optimal solutions. As for their robustness, when only one variable can be broken ($a = 1$), the number of breakages that can be repaired is above 90% in all cases, except for the case of the required optimality being 90%, which is really restricting the number of repairable breakages. When two variables can be broken ($a = 2$), the problem is harder, and the percentage of repairability slightly decreases, specially when the required optimality is 80% or more. Nonetheless, it should be noted that these are major improvements when comparing the results with the non-flexible approach (with many configurations being always unsatisfiable).

As for the *optimal most robust solutions*, the results are quite similar. In this case however, since the main objective is to find the most robust solution, there is a small decrease in optimality, although this is always above 99%, so it may be considered negligible. As for the robustness measure, when $a = 1$, the values are basically the same as with the first variant, with a small increase in some configurations. However, in the configurations with $a = 2$, the problems become harder, and there are many instances that cannot be solved, namely for (2,4,_). Therefore, the difference in the percentage of repaired breakages (%R) between the *most robust optimal solutions* and *optimal most robust solutions* is meaningless in this case.

Regarding solving times, for the *most robust optimal solutions* they are quite similar to the non-flexible approach, with the exception of the (2,4,80%) which takes significantly longer. As for the *most robust optimal solutions*, there are some configurations, (1,1,80/90%), (2,4,_) and (2,8,80%) that take between one and three orders of magnitude more of time.

				Most robust optimal				Most optimal robust			
a	b	%Opt	%SAT	%Sol	%O	%R	Avg Time	%Sol	%O	%R	Avg Time
1	1	60	95	100	100	96.18	0.42	100	100	96.18	1.25
		70	95	100	100	98.71	0.43	100	100	99.03	2.55
		80	50	100	100	90.08	0.97	90	99.99	91.85	144.4
		90	0	100	100	53.83	0.64	85	99.9	61.64	207.24
1	4	60	95	100	100	94	0.5	100	100	96.32	0.5
		70	95	100	100	96.72	0.5	100	100	99.52	0.51
		80	55	100	100	96.32	0.65	100	99.92	94.27	0.76
		90	0	100	100	70.8	0.6	100	98.1	80.12	19.2
1	8	60	95	100	100	96.32	0.45	100	100	96.32	0.45
		70	95	100	100	98.42	0.47	100	100	99.52	0.47
		80	60	100	100	94.27	0.55	100	100	94.27	0.56
		90	0	100	100	74.03	0.54	100	99.81	81.11	0.77
2	4	60	75	100	100	93.56	30.88	95	99.04	94.61	105.72
		70	15	100	100	95.07	24.25	50	93.17	97.6	177.35
		80	0	100	100	80.52	176.52	5	100	53.34	160.4
		90	0	100	100	45.5	31.84	10	100	60.24	673.19
2	8	60	75	100	100	93.63	21.44	100	100	93.63	20.65
		70	25	100	100	96.55	22.11	100	98.91	98.17	50.25
		80	0	100	100	84.09	52.35	90	99.38	84.6	236.88
		90	0	100	100	49.61	22.6	80	99.48	58.89	91.29

Table 5 Results for the matching distribution with flexible robustness

However, despite this increase in solving time, from these experiments we can conclude that giving away the strictness of full repairability (i.e. no flexibility at all) results in obtaining very good solutions, being optimal or near optimal, and being able to repair most of the potential breakages.

7 Conclusion

In this paper we have proposed a mechanism for finding robust solutions to weighted MaxSAT problems. We have first extended the approach of Ginsberg et al. [11] so that it can deal with the cost constraints needed in weighted MaxSAT 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. We have also proved the correctness of our approach. 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. This is an important issue, since quickly finding a repair solution is as important as knowing whether a robust solution exists.

Our approach for finding supermodels using difference cardinality constraint resembles those presented in [34, 16, 17, 13] for the CP setting. The reformulation approach presented in [34] for finding (1,0)-super solutions, referred to as $P + P$ in [16], implies duplicating the breakable variables and adding a *not equals* constraint between each original variable and its duplicate, among others. In [13] 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 [17] 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).

Our (a, b, β)-super solutions do not exactly match the (a, b)-super solutions of [15] because the cost β is not explicitly considered there. Notice that imposing a constraint on the cost of (a, b)-super solutions would only guarantee it for the solution found, but not on the possible repairs. Therefore, if we wanted such restriction to hold also for the repairs, we should modify the algorithm to find (a, b)-super solutions, whilst our approach guarantees a cost of β either for the initial solution and for every possible repair.

Thanks to the parametric definition of supermodels, we can deal with distinct sources of breakage and repairs. For instance, in the resource allocation setting, we have focused on robustness with respect to resource unavailability. However, we could easily take into account another source of breakage such as, e.g., request withdrawal, by simply setting both the sets S_1 and S_2 to the variables that encode requests.

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 always 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, in particular in very large problems. We could also think of a robustness notion where each breakable variable has a corresponding set of associated repairable variables.

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 MaxSAT, 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.

Since there are domains where the existence of robust solutions is really seldom, we have generalized our encoding so that partially robust solutions can be obtained. This is achieved 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 can give priority either to robustness or to optimality.

We have provided 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 [11]. Regarding results on search-based approaches, they are also restricted to find at most (1,3)-super solutions to CSP problems [13].

References

1. Ansótegui, C., Bonet, M.L., Levy, J.: Solving (Weighted) Partial MaxSAT through Satisfiability Testing. In: 12th International Conference on Theory and Applications of Satisfiability Testing, SAT 2009, *LNCS*, vol. 5584, pp. 427–440. Springer (2009)
2. Argelich, J., Cabiscol, A., Lynce, I., Manyà, F.: Modelling Max-CSP as Partial Max-SAT. In: 11th International Conference on Theory and Applications of Satisfiability Testing, SAT 2008, *LNCS*, vol. 4996, pp. 1–14. Springer (2008)
3. Argelich, J., Li, C.M., Manyà, F., Planes, J.: The First and Second Max-SAT Evaluations. Journal on Satisfiability, Boolean Modeling and Computation **4**(2-4), 251–278 (2008)
4. Argelich, J., Lynce, I., Silva, J.P.M.: On Solving Boolean Multilevel Optimization Problems. In: 21st International Joint Conference on Artificial Intelligence, IJCAI 2009, pp. 393–398 (2009)
5. Berre, D.L., Parrain, A.: The Sat4j library, release 2.2. Journal on Satisfiability, Boolean Modeling and Computation **7**(2-3), 59–6 (2010)
6. Bertsimas, D., Sim, M.: The Price of Robustness. Operations Research **52**(1), 35–53 (2004)
7. Bofill, M., Busquets, D., Villaret, M.: A Declarative Approach to Robust Weighted Max-SAT. In: 12th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP 2010, pp. 67–76. ACM (2010)
8. Cramton, P., Shoham, Y., Steinberg, R. (eds.): Combinatorial Auctions. MIT Press (2006)
9. Eén, N., Sörensson, N.: Translating Pseudo-Boolean Constraints into SAT. Journal on Satisfiability, Boolean Modeling and Computation **2**(1-4), 1–26 (2006)
10. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Schneider, M.: Potassco: The Potsdam answer set solving collection. AI Communications **24**(2), 105–124 (2011)
11. Ginsberg, M.L., Parkes, A.J., Roy, A.: Supermodels and robustness. In: 15th National Conference on Artificial Intelligence and 10th Innovative Applications of Artificial Intelligence Conference, AAAI/IAAI 1998, pp. 334–339. AAAI Press / The MIT Press (1998)
12. Gu, Z., Rothberg, E., Bixby, R.: Gurobi 4.0.2. `http://www.gurobi.com` (2010)
13. Hebrard, E.: Robust Solutions for Constraint Satisfaction and Optimisation under Uncertainty. Ph.D. thesis, University of New South Wales (2006)
14. Hebrard, E., Hnich, B., O'Sullivan, B., Walsh, T.: Finding Diverse and Similar Solutions in Constraint Programming. In: 20th National Conference on Artificial Intelligence and 17th Innovative Applications of Artificial Intelligence Conference, AAAI/IAAI 2005, pp. 372–377. AAAI Press / The MIT Press (2005)
15. Hebrard, E., Hnich, B., Walsh, T.: Robust Solutions for Constraint Satisfaction and Optimization. In: 16th Eureopean Conference on Artificial Intelligence, ECAI 2004, pp. 186–190. IOS Press (2004)
16. Hebrard, E., Hnich, B., Walsh, T.: Super Solutions in Constraint Programming. In: 8th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR 2004, *LNCS*, vol. 3011, pp. 157–172. Springer (2004)
17. Hebrard, E., Hnich, B., Walsh, T.: Improved Algorithm for Finding (a,b)-Super Solutions. In: Workshop on Constraint Programming for Planning and Scheduling, pp. 236–248 (2005)
18. Heras, F., Larrosa, J., de Givry, S., Schiex, T.: 2006 and 2007 Max-SAT Evaluations: Contributed Instances. Journal on Satisfiability, Boolean Modeling and Computation **4**(2-4), 239–250 (2008)
19. Holland, A.: Risk management for combinatorial auctions. Ph.D. thesis, Department of Computer Science, National University of Ireland, Cork (2005)
20. Holland, A., O'Sullivan, B.: Robust Solutions for Combinatorial Auctions. In: 6th ACM Conference on Electronic Commerce, EC 2005, pp. 183–192. ACM (2005)
21. Holland, A., O'Sullivan, B.: Weighted Super Solutions for Constraint Programs. In: 20th National Conference on Artificial Intelligence and 17th Innovative Applications of Artificial Intelligence Conference, AAAI/IAAI 2005, pp. 378–383. AAAI Press / The MIT Press (2005)

22. Hoos, H.H., Boutilier, C.: Solving Combinatorial Auctions Using Stochastic Local Search. In: 17th National Conference on Artificial Intelligence and 12th Innovative Applications of Artificial Intelligence Conference, AAAI/IAAI 2000, pp. 22–29. AAAI Press / The MIT Press (2000)
23. IBM ILOG CPLEX Optimizer. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/` (2011)
24. Leyton-Brown, K., Pearson, M., Shoham, Y.: Towards a Universal Test Suite for Combinatorial Auction Algorithms. In: ACM Conference on Electronic Commerce, EC 2000, pp. 66–76. ACM (2000)
25. Li, C.M., Manyà, F.: Handbook of Satisfiability, chap. MaxSAT, Hard and Soft Constraints, pp. 613–631. IOS Press (2009)
26. Liu, L., Truszczynski, M.: Satisfiability Testing of Boolean Combinations of Pseudo-Boolean Constraints using Local-search Techniques. Constraints **12**(3), 345–369 (2007)
27. Manquinho, V.M., Silva, J.P.M., Planes, J.: Algorithms for weighted boolean optimization. In: 12th International Conference on Theory and Applications of Satisfiability Testing, SAT 2009, *LNCS*, vol. 5584, pp. 495–508. Springer (2009)
28. Muñoz, V.: Robustness in resource allocation problems. Ph.D. thesis, Department d'Enginyeria Elèctrica, Electrònica i Automàtica, Universitat de Girona, Spain (2011)
29. Rossi, F., van Beek, P., Walsh, T. (eds.): Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier Science (2006)
30. Roussel, O., Manquinho, V.: Handbook of Satisfiability, chap. Pseudo-Boolean and Cardinality Constraints, pp. 695–734. IOS Press (2009)
31. Roy, A.: Fault Tolerant Boolean Satisfiability. Journal of Artificial Intelligence Research **25**, 503–527 (2006)
32. Soos, M.: CryptoMiniSat — a SAT solver for cryptographic problems. `http://planete.inrialpes.fr/~soos/CryptoMiniSat2/index.php` (2009)
33. Walsh, T.: SAT v CSP. In: 6th International Conference on Principles and Practice of Constraint Programming, CP 2000, *LNCS*, vol. 1894, pp. 441–456. Springer (2000)
34. Weigel, R., Bliek, C.: On Reformulation of Constraint Satisfaction Problems. In: 13th European Conference on Artificial Intelligence, ECAI 1998, pp. 254–258 (1998)
35. Williams, H.P.: Model Building in Mathematical Programming, 4th Edition. Wiley (1999)

Appendix (detailed proofs)

Proof of Lemma 1. 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 $I_{\overline{S}}(l) = I_{\overline{S \cup \{p\}}}(l)$. 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 $I \models F \Leftrightarrow I_{\overline{S \cup \{p\}}} \models F_{\overline{S \cup \{p\}}}$. □

Proof of Theorem 1.

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 $I \models \wedge_{S \subseteq S_1, 1 \leq |S| \leq a} (\vee_{T \subseteq (S_2 \setminus S) \cup S_3, |T \cap S_2| \leq b} (C_{\overline{S \cup T}} \wedge (B_{\overline{S \cup T}} \leq \beta)))$. 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 $I \models B_{\overline{S \cup T}} \leq \beta$. □

Proof of Lemma 2.

First of all notice that from the definition of D it follows that the domain of I is $Var(F) \cup Var(F^{\mathsf{ren}_R})$. Moreover, since F^{ren_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^{ren_R}.

Now, let l be any literal occurring in $F_{\overline{D}}$, and l' its corresponding literal in F^{ren_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 $I(x) = I(\mathsf{ren}_R(x))$ then $x \notin D$ and, hence, l occurs in $F_{\overline{D}}$ with the same polarity with which l' occurs in F^{ren_R}. Then, since $I(x) = I(\mathsf{ren}_R(x))$, we have that $I(l) = I(l')$. If, otherwise, $I(x) \neq I(\mathsf{ren}_R(x))$ then $x \in D$ and, hence, l occurs in $F_{\overline{D}}$ with the opposite polarity with which l' occurs in F^{ren_R}. Then, since $I(x) \neq I(\mathsf{ren}_R(x))$, 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 $I \models F^{\mathsf{ren}_R} \Leftrightarrow I \models F_{\overline{D}}$. □

Proof of Lemma 3.

For the left-to-right implication, 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 $Var(F) \cup Var(F^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}})$ 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 $x^S \in Var(F^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}})$. 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 $I^S \models F_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}}$. Concerning $(\nabla^{\mathsf{ren}_{S_2 \setminus S,\, S}} \leq b)$, by Definition 6 and construction of I^S we have $I^S(\nabla^{\mathsf{ren}_{S_2 \setminus S,\, S}}) = |T \cap (S_2 \setminus S)| \leq |T \cap S_2| \leq b$. 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 $I' \models F_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}} \wedge (\nabla^{\mathsf{ren}_{S_2 \setminus S,\, S}} \leq b)$ 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 $I' \models (\nabla^{\mathsf{ren}_{S_2 \setminus S,\, S}} \leq b)$, 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 $I' \models F_{\overline{S}}^{\mathsf{ren}_{(S_2 \setminus S) \cup S_3,\, S}}$, 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. □

Proof of Theorem 2

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 $F = C \wedge (B \leq \beta)$. □