Meta-Constraints on Violations for Over Constrained Problems

Thierry Petit
ILOG-LIRMM
les Taissounières, 1681 route des Dolines,
06560 Valbonne, France
tpetit@ilog.fr, tpetit@lirmm.fr

Jean-Charles Régin
ILOG
les Taissounières, 1681 route des Dolines,
06560 Valbonne, France
regin@ilog.fr

Christian Bessière
LIRMM (UMR 5506 CNRS)
161 rue Ada,
34392 Montpellier Cedex 5, France
bessiere@lirmm.fr

Abstract

Constraint programming techniques are widely used to solve real-world problems. It often happens that such problems are over-constrained and do not have any solution. In such a case, the goal is to find a good compromise. A simple theoretical framework is the Max-CSP, where the goal is to minimize the number of constraint violations. However, in real-life problems, complex rules are generally imposed with respect to violations. Solutions which do not satisfy these rules have no practical interest. Therefore, many frameworks derived from the Max-CSP have been introduced. In this paper, we classify the most usual types of rules, and we show that some of them are not expressible in existing frameworks. We introduce a new paradigm in which all these rules can be encoded, through meta-constraints. Moreover, we show that most of existing frameworks can be included into our model.

1. Introduction

Constraint programming techniques are used more and more to solve real-word applications, such as time tabling, configuration, car sequencing, frequency allocation, scheduling, etc. A constraint network $\mathcal{N}$ is defined as a set of n variables $X = \{x_1, \ldots, x_n\}$, a set of current domains $\mathcal{D} = \{D(x_1), \ldots, D(x_n)\}$ where $D(x_i)$ is the finite set of possible values for variable x_i, and a set $\mathcal{C}$ of m constraints between variables. A constraint C on the set of variables $X(C)$ specifies the allowed combinations of values for the variables. Each combination of values is called a tuple on $X(C)$.

Given a constraint network, the search for an assignment to all variables that satisfies all the constraints is an NP-Hard problem called the Constraint Satisfaction Problem (CSP). Such an assignment is a solution of the CSP.

In many cases, real-life applications are over-constrained. This observation means that when they are coded in terms of CSP, such CSPs have no solution. In this situation, it is necessary to relax the problem in order to obtain a solution. The goal is to find a good compromise, that is, a total assignment that best respects the set of constraints, even if some of them are violated. This assignment is a solution of the over-constrained problem.

The most well-known approach is to find an assignment which minimizes the number of violated constraints. The theoretical framework used to do so is called the Maximal Constraint Satisfaction Problem (Max-CSP). Many works have been carried out in order to solve instances of Max-CSP [5, 12, 6, 7]. Unfortunately, this framework is not realistic enough: in many real-life applications the goal is much more complex than a minimization of the number of violations.

For instance, let us consider an example derived from a real problem [3]. The task is to schedule managers for a directory assistance center, with a set of activities and a set of persons, over a week. We assume that each person has to perform several activities per day. Each activity has to be performed a certain number of times, which is bounded by a minimum and a maximum. Activities have different durations, and some of them require material resources or specific skills. Furthermore, persons can express preferences with respect to nature of their activities.

The number of constraints often makes such a problem over-constrained. In this case it is necessary to violate some constraints, while keeping in mind that the obtained schedule must be realistic. Therefore, some rules are generally imposed on violated constraints.

We focus on a subset of constraints and the associated rules. A denotes an activity that requires a resource: the video-conference room. We assume that this activity must be performed several times per day.

Constraints	
C_1	Carrying outs of A must be separated one another
C_2	Paul does not want to perform A
C_3	Silvia should not perform A
C_4	Each day, all activities of a person should be different
C_5	A person should not end working after 8 p.m.
C_6	A person should not start working before 8 a.m.

C_1 must be set because only one video-conference room is available; C_2 is a preference; C_3 is related to a person who does not have skills necessary to perform A; C_4 is related to distribution of activities; C_5 and C_6 fix maximal and minimal bounds of a workday.

Assuming that some constraints have to be violated, we define the following rules in order to obtain realistic solutions. Some of these rules are local to one constraint whereas others involve several constraints:

1. C_1 must not be violated because it is related to a physical aspect of the problem: there is only one video-conference room.

2. A violation of C_2 may be prefered to a violation of C_3, because its consequences are less serious.

3. When C_4 is violated, we aim at minimizing the number of repetitions of each activity, in order to respect at least partially its semantic.

4. A schedule such that some persons have all their preferences satisfied and others have all their preferences violated is not realistic. There is no reason to favour randomly certain persons.

5. Assume that a person ends working after 8 p.m. on Monday. It is not realistic to ask to this person to start before 8 a.m. on Tuesday. Consequently, a rule of dependence has to be defined for each person: if C_5 is violated, then the next day C_6 must not be violated for this person.

6. Assume that C_5 is violated for Claudia's schedule on Monday and Tuesday. We may need to compensate these two succesive violations by imposing that, on Wednesday, a new constraint must be satisfied: Claudia must not end working after 4 p.m. This constraint was not defined in the initial problem.

In order to deal with different priorities or degrees of violation of constraints, various frameworks derived from Max-CSP have been suggested. They provide different kinds of valuation structure associated with constraints or tuples and various kinds of objective functions. However, these frameworks are sometimes not well suited to express global rules about violated constraints, and such rules are necessary to get realistic solutions in real-life problems.

Throughout this paper, we introduce a new framework that provides the possibility to add a set of meta-constraints to the problem, in order to express such rules.

This paper is organized as follows: first, we classify the different classes of rules about violated constraints that were highlighted in the example of the introduction. We present the related work for over-constrained problems, and we show that some of the rules are not expressible in existing frameworks. Then, we introduce our framework, and we explain how these rules can be expressed. Finally, we discuss solving, and we show how CSP paradigms can be encoded in our framework.

2. Rules imposed on violated constraints

The example in the introduction highlights the fact that some rules about violations generally have to be satisfied by the solutions. It is interesting to try to classify them.

First, two sets of constraints have to be distinguished: hard constraints that must be satisfied in any solution, and soft constraints that may possibly be violated. In the example, C_1 is hard because it is related to a physical aspect of the problem (rule 1.). In this paper, we will use the following notation:

Notation 1 Let $\mathcal{N} = (X, \mathcal{D}, \mathcal{C})$ be a constraint network, we define the following notation:

- $\mathcal{C}_h \subseteq \mathcal{C}$ is the set of hard constraints, $m_h = |\mathcal{C}_h|$.
- $\mathcal{C}_s \subseteq \mathcal{C}$ is the set of soft constraints, $m_s = |\mathcal{C}_s|$.
- $\mathcal{C}_h$ and $\mathcal{C}_s$ are disjoint and $\mathcal{C} = \mathcal{C}_h \cup \mathcal{C}_s$

No rule has to be defined about hard constraints because in any case they must be satisfied. Concerning soft constraints, we suggest enumerating the more frequent rules that are imposed in order to get realistic solutions:

2.1. Priorities

Generally, not all the soft constraints have the same importance in a problem. In this case the goal is to favour the satisfaction of the most important ones. For instance, in the example, a violation of C_2 will be prefered to a violation of C_3, because its consequences are less serious (rule 2.).

2.2. Degree of violation

Sometimes there are different ways to violate a constraint. For example, C_4 may be violated more or less strongly: a violation such that only two among all the activities are the same may be prefered to one such that the same activity is performed all the day (rule 3.).

2.3. Homogeneity

Violations generally have to be homogeneously distributed in solutions. In the example, the distribution of violations of preferences of each person must be fair (rule 4.).

2.4. Dependences

Particular configurations of violations entail the necessity to satisfy some other constraints: "if $C_i \in \mathcal{C}_s$ is violated then $C_j \in \mathcal{C}_s$ must not be violated" (rule 5.). Notice that a dependence can involve more than two constraints. This remark is important because it highlights the fact that the problem is about expressing interactions between subsets of constraints.

2.5. Dependences with new constraints

It is sometimes interesting to add new constraints to the network when certain violations appear. This observation can be linked to the works of Mittal and Falkenhainer about "Dynamic CSPs" [8]. These new constraints are used to compensate for some violations. In rule 6. of the example, the new hard constraint activated when C_5 is violated twice consecutively does not belong to the initial problem. This is a new constraint derived from C_5 (and more restrictive). It could be a constraint which has nothing to do with C_5.

3. Related work

3.1. Max-CSP

In a Max-CSP [5], no distinction is made between hard and soft constraints. All the constraints have the same importance. The objective is to find an assignment that minimizes the number of violated constraints.

3.2. Extended frameworks

Various frameworks derived from Max-CSP have been suggested, and most of them can be encoded through two generic paradigms called valued CSP [11] and semiring CSP [2]. We will not explain here these generic paradigms, because our aim is just to summarize the principles of the most well-known frameworks.

In all these frameworks, a valuation is associated with each constraint or each tuple, and the set of valuations is integrated into an optimization function, based on indempotent or monotonic operators.

- Weighted CSP: the Weighted CSP is an extension of the Max-CSP such that a weight is associated with each constraint[1] in order to express different levels of priority. It is preferable to violate constraints which have a low weight than ones with a greater weight. The goal is to minimize the total sum of weights of violated constraints.

• Possibilistic CSP: in Possibilistic CSPs [10] a preference level is associated with each constraint. Such a level is always between 0 and 1, where 1 represents the maximal level of importance of a constraint and 0 the minimal. The goal is to minimize the maximal level among violated constraints. Indeed, instead of using a strict monotonic operator in order to compute the cost of a solution, as '+' for a sum of valuations, possibilistic CSPs use an idempotent operator 'max'.

• Fuzzy CSP: in Fuzzy CSPs [4] a preference level is associated with each tuple of each constraint. Such a level is always between 0 and 1, where 1 represents the best value (that is, the tuple is allowed) and 0 the worst one. The minimal value associated with a tuple corresponds to the more strongly violated constraint. The goal is to maximize this minimal value. Indeed, similarly to possibilistic CSPs, fuzzy CSPs use an idempotent operator 'min'. Notice that by decomposing each constraint by a set of constraints, such that each of them corresponds to a tuple, we may turn the problem into a Possibilistic CSP.

4. Expression of rules in existing frameworks

Hard and soft constraints can be distinguished in all these frameworks, except Max-CSP. In weighted and possibilistic CSPs, a hard constraint is expressed by a huge valuation. In Fuzzy CSPs, it is expressed by associating a null value with all the tuples that violate it. Moreover, most existing frameworks are well suited to express priorities and/or different degrees of violation of constraints. Various optimization criteria that take valuations into account can be used, in order to deal with different classes of problems.

On the other hand, a valuation is local to one constraint, or local to one tuple. Therefore, these frameworks are not well suited to express rules about violations that take several constraints into account, such rules as homogeneity or dependence. Indeed, in all frameworks derived from Max-CSP, the only entity involving all the valuations is the objective function.

[1] A particular kind of weighted CSP consists of associating a weight with each tuple of each constraint [2].

Type of rule	Semantic	Related work
Priorities	Expresses that some constraints have more priority than others	Weighted and Possibilistic CSP
Degree of violation	Expresses that a given constraint can be more or less strongly violated	Fuzzy CSP
Homogeneity	Expresses that violations must be homogeneously distributed	-
Dependence with respect to violations	A soft constraint can become hard according to a specific rule: "if C_i and C_j are violated then C_k must not be violated"	-
Dependence involving new constraints	A new hard constraint can be added according to a specific rule: "if C_i is violated then a new hard constraint R_j must be added to the network"	nothing specific to over-constrained problems

5. Meta-constraints for expressing rules about violated constraints

Existing frameworks provide the possibility to express certain types of rules about violations, but others are not expressible. We propose a new model in which meta-constraints can be added, in order to express all the kinds of rule described in the previous table. By "meta-constraints" we mean "constraints on constraints". This denotation seems to be the most usual in constraint programming [1].

Let $\mathcal{N} = (X, \mathcal{D}, \mathcal{C})$ be a constraint network. In our framework, $\mathcal{N}$ is turned into a new constraint network $\mathcal{N}'$. According to the previous section, three kinds of meta-constraints are distinguished:

1. Meta-constraints used to add homogeneity criteria with respect to violations in $\mathcal{C}_s$. This set of meta-constraints is denoted by $\mathcal{C}_{hom}$.
2. Meta-constraints used to express dependences with respect to violations in $\mathcal{C}_s$. This set of meta-constraints is denoted by $\mathcal{C}_{dep}$.
3. Meta-constraints related to dependences between soft constraints $\mathcal{C}_s$ and new constraints $\mathcal{R}$, that are activated only when particular configurations of violations appear in $\mathcal{C}_s$. This set of meta-constraints is denoted by C_{act}.

This list is not exhaustive. More specific meta-constraints may be defined for each application.

5.1. A new set of variables

For each soft constraint, a distance is taken into account, similarly to many existing frameworks for over-constrained problems. However, we suggest expressing such a distance through a new variable, which is added to the new constraint network. The set of these variables is denoted by S. A one-to-one mapping is defined between S and $\mathcal{C}_s$. For each $s \in S$ linked to $C \in \mathcal{C}_s$, a constraint Cs_{dst} imposing that the value of s must be equal to the value in E returned by a valuation function γ is defined:

$$Cs_{dst} : [s = \gamma(X(C))]$$

The set of these constraints is denoted by $\mathcal{C}_{dst}$. E is totally ordered and has a minimal element that expresses a state of "satisfaction" of constraints $\mathcal{C}_s$. For the sake of clarity, we will assume that E is the set of natural numbers with the minimal element 0 and the total order '>'.

A major interest of this model is that new constraints involving variables S are meta-constraints useful to control violations in solutions. Indeed, a value assigned to a variable $s \in S$ indicates whether the corresponding constraint of $\mathcal{C}_s$ is violated or satisfied. Therefore, a new constraint defined on a subset of S naturally expresses a rule concerning satisfaction and violation of the constraints of $\mathcal{C}_s$ linked to the variables of this subset. This point will be fully detailed later.

5.2. Disjunctive constraints

We suggest defining an optimization problem that satisfies the set of hard constraints $\mathcal{C}_h$, possibly a set of meta-constraints, and also a set of disjunctive constraints in order to integrate the variables S. This set of disjunctive constraints is denoted by $\mathcal{C}_{disj}$ and replaces the set of soft constraints $\mathcal{C}_s$ in the new constraint network $\mathcal{N}'$.

Let $C \in \mathcal{C}_s$ be associated with a variable $s \in S$. A value of s corresponds to a state of the constraint: if $s = 0$ then C must be satisfied, and if $s > 0$ then C must be violated. This semantic is expressed in $\mathcal{N}'$ through the following disjunctive constraint $Cs_{disj} \in \mathcal{C}_{disj}$:

$$Cs_{disj} : [C \wedge [s = 0]] \vee [\neg C \wedge [s > 0]]$$

The problem defined by $\mathcal{N}'$ is not over-constrained: the set of soft constraints has been replaced by the disjunctions. The optimization criterion is related to values assigned to variables S, and some of them semantically correspond to violations of initial constraints $\mathcal{C}_s$. An interesting consequence is that this model can be integrated in a constraint solver without any restriction.

5.3. Optimization criterion

The optimization criterion is expressed through a variable, denoted by *cost*. Its initial domain is denoted by $D(cost)$. A constraint C_{cost} imposes that *cost* must be equal to the value of an optimization function $\varphi : S^{m_s} \rightarrow E$ including all the variables S:

$$C_{cost} : [cost = \varphi(S)]$$

The goal is to solve a sequence of decision problems such that the solution of each decision problem must provide a better value of *cost*: a new constraint imposing that cost must be better than its assigned value in the solution of the previous problem is added to each new decision problem. This is a very classical schema for solving optimization problems in constraint programming.

The main interest of using a variable for the objective and valuations is that propagation can be directly carried out in both directions. Deductions can be made from the objective to the variables of the problem and from the variables to the objective, for any kind of constraint and valuations (associated with tuples or constraints).

5.4. Meta-constraints on violations

Constraints involving variables of S are meta-constraints that can be added to the problem in order to control violations in solutions. Indeed, let us remember that a distance value also expresses the state of a constraint: a satisfied constraint in $\mathcal{C}_s$ corresponds to a null distance, whereas a violated constraint in $\mathcal{C}_s$ corresponds to a strictly positive distance.

Since the newly defined problem must not be over-constrained, meta-constraints are hard and must necessarily be satisfied. Let us describe how the different kinds of meta-constraints can be expressed:

• Homogeneity constraints $\mathcal{C}_{hom}$: Homogeneity can be expressed through a cardinality constraint defined on subsets of S. For each subset, such a constraint is defined in order to impose that at least k variables of S must have a null distance, where k is a natural positive number lower than the size of the subset. For more complex homogeneity rules, global cardinality constraints [9] can be used.

• Dependences $\mathcal{C}_{dep}$: Dependences with respect to violation and satisfaction of constraints $\mathcal{C}_s$ can be also expressed through meta-constraints on variables S. For instance, consider the simplest dependence between two constraints that corresponds in the example to the rule related to hours of start and end of work of a person, during successive days: "If the constraint $C_i \in \mathcal{C}_s$ is violated then $C_j \in \mathcal{C}_s$ must not be violated." The constraint used to define such a rule is:

$$C : [[s_i > 0] \Rightarrow [s_j = 0]]$$

• Dependences with new constraints $\mathcal{C}_{act}$: It is interesting to add new constraints when specific configurations of violations appear in $\mathcal{C}_s$. For example, it may be necessary to express: "If $C_i \in \mathcal{C}_s$ is violated then a new constraint R_j on variables of X must be added to the network." In order to define such a kind of constraint, we add a new set of variables denoted by $S_{\mathcal{R}}$. These variables are similar to variables S, but they are related to a new set of constraints $\mathcal{R}$. The important difference is that these variables have nothing to do with the objective function φ. This distinction is logical, because the corresponding constraints $\mathcal{R}$ do not belong to the initial over-constrained problem. Activation of constraints of $\mathcal{R}$ depends on the values that are assigned to variables $S_{\mathcal{R}}$. The domain of each b in $S_{\mathcal{R}}$ is $\{0, 1\}$: 0 indicates that the corresponding contraint is satisfied, whereas 1 indicates that the constraint is violated. A set of disjunctive constraints $\mathcal{R}_{disj}$ is defined in order to express this semantic. Given $R \in \mathcal{R}$, the disjunction in $\mathcal{R}_{disj}$ is:

$$Rb_{disj} : [R \wedge [b = 0]] \vee [\neg R \wedge [b = 1]]$$

Meta-constraints that specify conditions of activation of constraints of $\mathcal{R}$ can be defined: given $s_i \in S$, corresponding to $C_i \in \mathcal{C}_s$, the previous rule is expressed by:

$$C : [[s_i > 0] \Rightarrow [b_j = 0]]$$

5.5. A new constraint network

The new network $\mathcal{N}' = (X', \mathcal{D}', \mathcal{C}')$ is not over-constrained, $\mathcal{C}' = \mathcal{C}'_h$ and $\mathcal{C}'_p = \emptyset$. A solution is an assignment with an optimal value assigned to $cost$, and must satisfy all constraints $\mathcal{C}'$.

Definition 1 $\mathcal{N}'$ is the constraint network defined by:

- $X' = X \cup S \cup S_{\mathcal{R}} \cup \{cost\}$
- $\mathcal{D}' = \mathcal{D} \cup \mathcal{D}_S \cup \{0, 1\}^{m_{\mathcal{R}}} \cup D(cost)$
- $\mathcal{C}' = \mathcal{C}_h \cup \mathcal{C}_{dst} \cup \mathcal{C}_{disj} \cup \mathcal{R}_{disj} \cup \mathcal{C}_{hom} \cup \mathcal{C}_{dep} \cup \mathcal{C}_{act} \cup \{C_{cost}\}$

5.6. Encoding existing frameworks

The two following tables show how Max-CSP and its extensions can be expressed through the constraint network $\mathcal{N}'$. In all cases $S_{\mathcal{R}}$, $\mathcal{R}$, $\mathcal{R}_{disj}$, $\mathcal{C}_{hom}$, $\mathcal{C}_{dep}$ and $\mathcal{C}_{act}$ are empty, because no meta-constraints are defined. Different approaches are obtained by selecting different functions γ and φ.

1. function γ:

	E	Domain of γ
Max-CSP	integers	$\{0, 1\}$
Weighted CSP	reals	$\{0, r\}$
Possibilistic CSP	reals	$\{0, r\}, r \leq 1$
Fuzzy CSP	reals	$[0, 1]$

2. function φ and goal:

	φ	Goal
Max-CSP	$\sum_{i=1}^{m_s}(s_i))$	minimize *cost*
Weighted CSP	$\sum_{i=1}^{m_s}(s_i))$	minimize *cost*
Possibilistic CSP	$max_{i=1}^{m_s}(s_i))$	minimize *cost*
Fuzzy CSP	$min_{i=1}^{m_s}(1-s_i))$	maximize *cost*

6. Discussion

In this section, we discuss the efficiency of solving an over-constrained problem through our framework; we also discuss the particular case where the new constraint network is itself over-constrained.

6.1. Solving

In order to compare our framework with existing ones, an important point is to know whether an increase of complexity is induced by the introduction of meta-constraints and state variables. Let us consider the case of a Max-CSP. We have the following property:

Property 1 Let $x \in X$ be involved in a constraint $C \in \mathcal{C}$. Let s be the variable of S associated with C. If all values of $D(x)$ are not consistent with C (resp. $\neg C$) then $s = 1$ (resp. 0).

A direct consequence is that we can solve a Max-CSP only by assigning successively variables X. For instance, a Branch and Bound Based search algorithm can be used. Values will be automatically assigned to variables S by applying property 1. The same kind of propagation can be performed on variables $S_{\mathcal{R}}$.

In this case, the search tree has exactly the same depth as one of the other solving methods. The cost of the simple filtering stemming from the property above is not significant, providing that it is easy to check the consistency of values with constraints (a situation which is almost always the case). Consequently, if the solving method only assigns values to variables X, the increase of complexity due to meta-constraints and state variables will not be restrictive.

Moreover, when a value is assigned to a state variable, a specific filtering of the corresponding constraint $C \in \mathcal{C}$ (or its oppsite) can be activated. This practice could lead to a gain of efficiency compared with existing frameworks, in which it is not possible to use specific filtering algorithms of constraints.

However, regarding future work, the study of solving algorithms associated with this new framework requires more investigation, and more experiments have to be performed.

6.2. Satisfaction of the new CSP

The newly problem can be itself over-constrained because of conflicting meta-constraints. In this case, there is no solution satisfying the rules about violations. In other words, no acceptable compromise exists. This case is similar to the one where the set of hard constraints is itself over-constrained (a case which can appear also in existing frameworks). The rules which define the features of acceptable solutions are mandatory. Therefore, the answer “no solution exists” seems to be justified when they cannot be satisfied.

7. Conclusion

We have suggested a new framework that provides the ability to define meta-constraints on violations in order to get realistic solutions. Moreover, our model can be directly integrated into a constraint solver and deal with any type of constraint and any solving algorithm. We have also proved that the Maximal Constraint Satisfaction Problem and most of the existing approaches can be encoded in our framework, through particular parameters.

8. Aknowledgments

The work of ILOG authors was partially supported by the IST Programme of the Commission of the European Union through the ECSPLAIN project (IST-1999-11969).

References

[1] P. Berlandier. The use and interpretation of meta level constraints. EPIA'93, 1993.

[2] S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier. Semiring-based csps and valued csps: Frameworks, properties, and comparison. Constraints, 4:199–240, 1999.

[3] Y. Caseau, P.-Y. Guillo, and E. Levenez. A deductive and object-oriented approach to a complex scheduling problem. DOOD'93, 1993.

[4] D. Dubois, H. Fargier, and H. Prade. The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction. FUZZ-IEEE'93, 1993.

[5] E. Freuder and R. Wallace. Partial constraint satisfaction. Artificial Intelligence, 58:21–70, 1992.

[6] J. Larrosa, P. Meseguer, and T. Schiex. Maintaining reversible dac for max-csp. Artificial Intelligence, 107:149–163, 1999.

[7] J. Larrossa and P. Meseguer. Partition-based lower bound for max-csp. Proceedings CP, pages 303–315, 1999.

[8] S. Mittal and F. Falkenhainer. Dynamic constraint satisfaction problems. Proceedings AAAI, pages 25–32, 1990.

[9] J.-C. Régin. Generalized arc consistency for global cardinality constraint. Proceedings AAAI, pages 209–215, 1996.

[10] T. Schiex. Possibilistic constraint satisfaction problems, or "how to handle soft constraints?". Proceedings of 8th Conf. of Uncertainty in AI, 1992.

[11] T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: hard and easy problems. Proceedings IJCAI, 1995.

[12] G. Verfaillie, M. Lemaître, and T. Schiex. Russian doll search for solving constraint optimisation problems. Proceedings AAAI, pages 181–187, 1996.