An Original Constraint Based Approach for Solving Over Constrained Problems

J-C. RÉGIN[1], T. PETIT[1,2], C. BESSIÈRE[2], and J-F. PUGET[3]

[1]ILOG, 1681, route des Dolines, 06560 Valbonne, FRANCE
[2]LIRMM (UMR 5506 CNRS), 161, rue Ada, 34392 Montpellier Cedex 5, FRANCE
[3]ILOG S.A., 9, rue de Verdun, BP 85, 94253 Gentilly Cedex, FRANCE
E-mail: regin@ilog.fr, tpetit@ilog.fr, tpetit@lirmm.fr, bessiere@lirmm.fr, puget@ilog.fr
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).

Abstract. In this paper we present a new framework for over constrained problems. We suggest to define an over-constrained network as a global constraint. We introduce two new lower bounds of the number of violations, without making any assumption on the arity of constraints.

1 Introduction

Encoding real-world problems often leads to define over constrained networks, which do not have any solution that satisfies all the constraints. In this situation the goal is to find the best compromise. One of the most well-known theoretical frameworks for over constrained problems is the Maximal Constraint Satisfaction Problem (Max-CSP). In a Max-CSP, the goal is to minimize the number of constraint violations. Best algorithms for solving Max-CSP [4, 5] are based on computation of lower bounds of the number of violations. All these algorihms are related to binary constraint networks. On the other hand, solving real-life problems requires the use of non binary constraints [6].

In this paper we present a new framework for over constrained problems. No hypothesis is made on the arity of constraints. We introduce two new lower bounds of the number of violations. The first one is a generalization of the previous studies for binary Max-CSP to the non binary case, through a variable-based partitioning of the constraint set. The second one is an original lower bound based on computation of disjoint conflict sets. Moreover, one advantage of our framework is that filtering algorithms associated with constraints can be used in a way similar to classical CSPs.

2 Background

A *constraint network* $\mathcal{N}$ is defined as a set of n *variables* $X = \{x_1, \ldots, x_n\}$, a set of *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 *constraints* between variables. A *constraint*

C on the ordered set of variables $X(C) = (x_{i_1}, \ldots, x_{i_r})$ is a subset $T(C)$ of the Cartesian product $D(x_{i_1}) \times \cdots \times D(x_{i_r})$ that specifies the *allowed* combinations of values for the variables $x_{i_1}, \ldots, x_{i_r}$. An element of $D(x_{i_1}) \times \cdots \times D(x_{i_r})$ is called a *tuple on* $X(C)$. $|X(C)|$ is the *arity* of C. A value a for a variable x is often denoted by (x, a). A tuple τ on $X(C)$ is *valid* if $\forall (x, a) \in \tau, a \in D(x)$. C is *consistent* iff there exists a tuple τ of $T(C)$ which is valid. A value $a \in D(x)$ is *consistent with* C iff $x \notin X(C)$ or there exists a valid tuple τ of $T(C)$ in which a is the value assigned to x.

Definition 1 Let x be a variable, a be a value of $D(x)$, $\mathcal{C}$ be a set of constraints, $\#inc((x, a), \mathcal{C}) = |\{C \in \mathcal{C} \text{ s.t. } (x, a) \text{ is not consistent with } C\}|$.

3 Satisfiability Sum Constraint

Let $\mathcal{N} = (X, \mathcal{D}, \mathcal{C})$ be a constraint network. We suggest to integrate $\mathcal{C}$ into a single constraint, called the Satisfiability Sum Constraint (ssc):

Definition 2 Let $\mathcal{C} = \{C_i, i \in \{1, \ldots, m\}\}$ be a set of constraints, and $S[\mathcal{C}] = \{s_i, i \in \{1, \ldots, m\}\}$ be a set of variables and $unsat$ be a variable, such that a one-to-one mapping is defined between $\mathcal{C}$ and $S[\mathcal{C}]$. A **Satisfiability Sum Constraint** is the constraint $ssc(\mathcal{C}, S[\mathcal{C}], unsat)$ defined by:

$$[unsat = \sum_{s_i=1}^{m} s_i] \wedge \bigwedge_{i=1}^{m} [(C_i \wedge (s_i = 0)) \vee (\neg C_i \wedge (s_i = 1))]$$

Notation 1 Given a $ssc(\mathcal{C}, S[\mathcal{C}], unsat)$, a variable x, a value $a \in D(x)$ and $\mathcal{K} \subseteq \mathcal{C}$:
$max(D(unsat))$ is the highest value of "current" domain of $unsat$;
$minUnsat(\mathcal{C}, S[\mathcal{C}])$ is the minimum value of $unsat$ consistent with $ssc(\mathcal{C}, S[\mathcal{C}], unsat)$;
$minUnsat((x, a), \mathcal{C}, S[\mathcal{C}])$ is equal to $minUnsat(\mathcal{C}, S[\mathcal{C}])$ when $x = a$;
$S[\mathcal{K}]$ is the subset of $S[\mathcal{C}]$ equals to the projection of variables $S[\mathcal{C}]$ on $\mathcal{K}$;
$X(\mathcal{C})$ is the union of $X(C_i), C_i \in \mathcal{C}$.

The variables $S[\mathcal{C}]$ are used to express which constraints of $\mathcal{C}$ must be violated or satisfied: a null value assigned to $s \in S[\mathcal{C}]$ expresses that its attached constraint C is satisfied, whereas 1 expresses that C is violated[1]. The variable $unsat$ represents the objective, that is, the number of violations in $\mathcal{C}$, equal to the number of variables of $S[\mathcal{C}]$ whose value is 1.

Through this formulation, a solution of a Max-CSP is an assignment that satisfies the ssc with the minimal possible value of $unsat$. A lower bound of the objective of a Max-CSP corresponds to a necessary consistency condition of the ssc. The different domain reduction algorithms established for Max-CSP correspond to specific filtering algorithms associated with the ssc. This point of view has some advantages in regards to the previous studies:
1. Any search algorithm can be used. Since we propose to define a constraint we

[1] The extension to valued CSPs [1] can easily be performed by defining larger domains for variables in $S[\mathcal{C}]$.

can easily integrate our framework into existing solvers.
2. In all this paper, no hypothesis is made on the arity of constraints $\mathcal{C}$.
3. When a value is assigned to $s_i \in S[\mathcal{C}]$, a filtering algorithm associated with $C_i \in \mathcal{C}$ (resp. $\neg C_i$) can be used in a way similar to classical CSPs.

4 Variable Based Lower Bound

The results presented in this section are a generalization to non binary constraints of previous works for Max-CSP [2, 7, 4].

4.1 Necessary Condition of Consistency

If $minUnsat(\mathcal{C}, S[\mathcal{C}]) > max(D(unsat))$ then $ssc(\mathcal{C}, S[\mathcal{C}], unsat)$ is not consistent. Therefore, a lower bound of $minUnsat(\mathcal{C}, S[\mathcal{C}])$ provides a necessary condition of consistency of a ssc. A possible way for computing it is to perform a sum of independant lower bounds, one per variable:

Definition 3 Given a variable x a constraint set $\mathcal{K}$,
$\#inc(x, \mathcal{K}) = min_{a \in D(x)}(\#inc((x, a), \mathcal{K}))$.

The sum of these minima with $\mathcal{K} = \mathcal{C}$ cannot lead to a lower bound of the total number of violations, because some constraints can be taken into account more than once. For instance, given a constraint C and two variables x and y involved in C, C can be counted in $\#inc(x, \mathcal{C})$ and also in $\#inc(y, \mathcal{C})$. In this case, the lower bound can be overestimated, and an inconsistency could be detected while the *ssc* is consistent. Consequently, for each variable, an independent set of constraints must be considered. In the binary case, the constraint graph[2] has been used in order to guarantee this independence [4]. Each edge is oriented and for each variable x only the constraints out-going x are taken into account. This idea can be generalized to the non binary case, by associating with each constraint one and only one variable involved in the constraint: the constraints are *partionned* w.r.t the variables that are associated with.

Definition 4 Given a set of constraints $\mathcal{C}$, a **var-partition** of $\mathcal{C}$ is a partition $\mathcal{P}(\mathcal{C}) = \{P(x_1), ...P(x_k)\}$ of $\mathcal{C}$ in $|X(\mathcal{C})|$ sets such that $\forall P(x_i) \in \mathcal{P}(\mathcal{C}) : \forall C \in P(x_i), x_i \in X(C)$.

Given a var partition $\mathcal{P}(\mathcal{C})$, the sum of all $\#inc(x_i, P(x_i))$ is a lower bound of the total number of violations, because all the sets belonging to $\mathcal{P}(\mathcal{C})$ are disjoint:

Definition 5 $LB(\mathcal{P}(\mathcal{C})) = \sum_{x_i \in X(\mathcal{C})} \#inc(x_i, P(x_i)), P(x_i) \in \mathcal{P}(\mathcal{C})$

Property 1 $\forall \mathcal{P}(\mathcal{C})$ a var-partition of $\mathcal{C}$. $LB(\mathcal{P}(\mathcal{C})) \leq minUnsat(\mathcal{C}, S[\mathcal{C}])$

Corollary 1 If $LB(\mathcal{P}(\mathcal{C})) > max(D(unsat))$ then $ssc(\mathcal{C}, S[\mathcal{C}], unsat)$ is not consistent.

[2] The vertex set of the constraint graph is the variable set and there is an edge between two vertices when there is a constraint involving these two variables.

4.2 Filtering Algorithm

From definition of $minUnsat((x,a), \mathcal{C}, S[\mathcal{C}])$ we have the following theorem:

Theorem 1 $\forall x \in X(\mathcal{C}), \forall a \in D(x)$: if $minUnsat((x,a), \mathcal{C}, S[\mathcal{C}]) > max(D(unsat))$ then (x,a) is not consistent with $ssc(\mathcal{C}, S[\mathcal{C}], unsat)$.

Therefore, any lower bound of $minUnsat((x,a), \mathcal{C}, S[\mathcal{C}])$ can be used in order to check the consistency of (x,a). A first lower bound is $\#inc((x,a), \mathcal{C})$:

Property 2 $\#inc((x,a), \mathcal{C}) \leq minUnsat((x,a), \mathcal{C}, S[\mathcal{C}])$

This property leads to a first filtering algorithm. However, it can be improved by including the lower bound of Property 1. We suggest to split $\mathcal{C}$ into two disjoint sets $P(x)$ and $\mathcal{C} - P(x)$, where $P(x)$ is the subset of constraints associated with x in a var-partition $P(\mathcal{C})$ of $\mathcal{C}$. Consider the following corollary of Theorem 1:

Corollary 2 Let $P(\mathcal{C})$ be a var-partition of $\mathcal{C}$, x a variable and $a \in D(x)$, if $minUnsat((x,a), P(x), S[P(x)])$
$+ minUnsat((x,a), \mathcal{C} - P(x), S[\mathcal{C} - P(x)]) > max(D(unsat))$
then (x,a) is not consistent with $ssc(\mathcal{C}, S[\mathcal{C}], unsat)$.

Note that $minUnsat(\mathcal{C} - P(x), S[P(x)]) \leq minUnsat((x,a), \mathcal{C} - P(x), S[P(x)])$. From this remark and Properties 1 and 2 we deduce the theorem:

Theorem 2 $\forall \mathcal{P}(\mathcal{C})$ a var-partition of $\mathcal{C}, \forall x \in X(\mathcal{C}), \forall a \in D(x)$, if $\#inc((x,a), P(x)) + LB(\mathcal{P}(\mathcal{C} - P(x))) > max(D(unsat))$ then a can be removed from its domain.

5 Constraint Based Lower Bound

An original lower bound of the number of violations in $\mathcal{C}$, corresponding to a lower bound of $minUnsat(\mathcal{C}, S[\mathcal{C}])$, can be obtained by successive computations of disjoint conflict sets of $\mathcal{C}$.

Definition 6 A conflict set is a subset $\mathcal{K}$ of $\mathcal{C}$ which satisfies: $minUnsat(\mathcal{K}, S[\mathcal{K}]) > 0$.

We know that a conflict set leads to at least one violation in $\mathcal{C}$. Consequently, if we are able to compute q disjoint conflict sets in $\mathcal{C}$ then q is a lower bound of $minUnsat(\mathcal{C}, S[\mathcal{C}])$. They must be disjoint to guarantee that all violations taken into account are independent. For each $C_i \in \mathcal{C}$ such that $D(s_i) = 1$, the set $\{C_i\}$ is a conflict set. Moreover, constraints C_i of $\mathcal{C}$ with $D(s_i) = 0$ are not interesting in the determination of conflict sets. Hence we will focus on the set of constraints C_i of $\mathcal{C}$ with $D(s_i) = \{0,1\}$, denoted by $\mathcal{C}_?$.

Consider any ordering $\prec$ on $\mathcal{C}_?$. De Siqueira N. and Puget have shown that a conflict set of $\mathcal{C}_?$ can be simply computed by temporarily setting the variables of $S[\mathcal{C}_?]$ to 0 w.r.t. $\prec$ until a failure occurs. When a variable of $S[\mathcal{C}_?]$ attached to $C \in \mathcal{C}_?$ is set to 0 then values from domains of variables $X(C)$ that are not consistent with C are removed. When a failure occurs, then all the constraints C

for which s has been set to 0 form a conflict set. Given a set of constraints Q, we call $\textsc{computeConflictSet}(Q)$ the function which implements this algorithm.

A set of disjoint conflict sets can be easily computed by calling function $\textsc{computeConflictSet}(Q)$ with $Q = \mathcal{C}_?$ and by iteratively calling it with $Q \leftarrow Q - \mathcal{K}$ each time a conflict set $\mathcal{K}$ is found. The algorithm stops when Q is empty or when no conflict set is detected in Q. The lower bound depends on the number of conflict sets, and, since they are disjoint, on the size of the conflicts sets.

Definition 7 Let Q be a set of constraint. A minimal conflict set w.r.t. $\textsc{computeConflictSet}$ is a subset $\mathcal{K}$ of Q such that $\forall C \in \mathcal{K}$, $\textsc{computeConflictSet}(\mathcal{K} - \{C\})$ detects no conflict set.

De Siqueira N. and Puget have suggested a simple algorithm for finding minimal conflict set from a conflict set $\mathcal{K}$ [3]. This algorithm calls at most $|\mathcal{K}|$ times the $\textsc{computeConflictSet}$ function. $\textsc{computeMinimalConflictSet}(\mathcal{K})$ denotes the function that returns a minimal conflict set of $\mathcal{K}$. We can now propose an original algorithm for computing a lower bound LB of $minUnsat(\mathcal{C}, S[\mathcal{C}])$:

Step 1: Let $LB \leftarrow |\{C_i \in \mathcal{C} \text{ s.t. } s_i = 1\}|$ and $Q \leftarrow \mathcal{C}$;
Step 2: $\mathcal{K} \leftarrow \textsc{computeConflictSet}(Q)$;
    if $\mathcal{K} = \emptyset$ then return LB else $\mathcal{K}_{min} \leftarrow \textsc{computeMinimalConflictSet}(\mathcal{K})$;
Step 3: $LB \leftarrow LB + 1$; $Q \leftarrow Q - \mathcal{K}_{min}$; goto Step 2.

Corollary 3 If $LB > max(D(unsat))$ then $ssc(\mathcal{C}, S[\mathcal{C}], unsat)$ is not consistent.

6 Conclusion

This paper presents a new framework for over constrained problems, which can be directly integrated into existing solvers. New lower bounds of number of violations are introduced without making any assumption on the constraints.

References

1. 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.
2. E. Freuder and R. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58:21–70, 1992.
3. J.L. de Siqueira N. and J.-F. Puget. Explanation-based generalization of failures. *Proceedings ECAI*, pages 339–344, 1988.
4. J. Larrosa, P. Meseguer, and T. Schiex. Maintaining reversible dac for max-csp. *Artificial Intelligence*, 107:149–163, 1999.
5. J. Larrossa and P. Meseguer. Partition-based lower bound for max-csp. *Proceedings CP*, pages 303–315, 1999.
6. H. Simonis. Problem classification scheme for finite domain constraint solving. *CP Workshop on C.P. Applications: An Inventory and Taxonomy*, pages 1–26, 1996.
7. R. Wallace. Directed arc consistency preprocessing as a strategy for maximal constraint satisfaction. *Proceedings ECAI*, pages 69–77, 1994.