Intelligent Splitting for Disjunctive Numerical CSPs

Thomas Douillard, Christophe Jermann, Frédéric Benhamou

Intelligent Splitting for Disjunctive Numerical CSPs

Thomas Douillard, Christophe Jermann, and Frédéric Benhamou

LINA (CNRS FRE 2729) - University of Nantes
2 rue de la Houssinière, F-44300 Nantes
<first-name>.<last-name>@univ-nantes.fr

Abstract. Disjunctions in numerical CSPs appear in applications such as Design, Biology or Control. Generalized solving techniques have been proposed to handle these disjunctions in a Branch&Prune fashion. However, they focus essentially on the pruning operation. In this paper, we present experimental evidences that significant performance gains can be expected by exploiting the disjunctions in the branching operation.

1 First-order numerical CSPs

A numerical CSP (NCSP) is composed of variables whose domains are subsets of $\mathbb{R}$ (typically intervals) and constraints expressed as equations and inequalities on these variables. A solution to a NCSP is an assignment of real values (in practice, *small* intervals) to the variables that satisfies the constraints, i.e. they are considered *in conjunction*. However, in applications like Design, Biology or Control [8], it is not rare that constraints are considered *in disjunction*.

Consider the design of an industrial mixer [2]. The mixer M is composed of a vessel V and an agitator A; the agitator is itself composed of an engine E, an impeller I and a shaft S, while the vessel can optionally include a cooler C and a condenser D. The dimensioning of each component can be expressed as a NCSP. Hence, the mixer consists of 7 NCSPs (each denoted by the same letter as the corresponding component) linked together by the compositional structure of the mixer: $M = (A \wedge (E \wedge I \wedge S)) \wedge (V \wedge (C \vee D \vee True))$.

Formalisms derived from CSP have been proposed to handle this kind of problems [5, 9, 2]. However, it is often possible to handle them directly as *first-order NCSPs*, i.e. first-order logical formulas whose atoms are numerical constraints. Ratschan proposed a generalized framework for solving first-order NCSPs [6, 7] which we will use as a basis for our work. This framework defines first-order consistency as a generalization of classical numerical consistency and proposes a general Branch&Prune algorithm for solving first-order NCSPs. The main contribution is centered on the pruning operation, and more specifically on handling the logical quantifiers in first-order NCSPs.

In this paper, building upon the idea of constructive disjunction [3] much exploited in finite domains CSPs [4], we propose a general principle for intelligent splitting in disjunctive, a subset of first-order NCSPs without quantifiers, and we present experimental evidences that much can be gained following this principle.

2 Solving disjunctive NCSPs

In finite domains CSPs, constructive disjunction has been used to perform efficient pruning of disjunctions of constraints [4]. Intuitively, the consistent domain of a variable in a disjunction is the union of its consistent domains for all the alternatives in the disjunction. This can be efficiently computed by exploiting two properties : first if in a disjunction a variable is not constrained in all the alternatives, then its domain can not be pruned; second if a value in a domain is supported in one alternative, there is no need to search a support for this value in the other alternatives.

Ratschan has defined a general solving framework following a Branch&Prune approach for first-order NCSPs [6, 7]. We will not detail the complete framework but suffice it to say that pruning disjunctions is done in a similar way, taking the *hull* of the alternatives consistent domains. See Fig. 1 for an example.

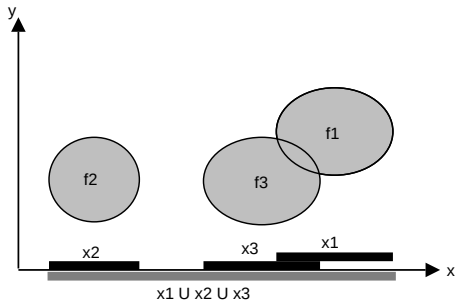

Fig. 1. A formula $f_1 \vee f_2 \vee f_3$; the consistent domain x_i of variable x for each alternative f_i; the consistent domain $Hull(x_1 \cup x_2 \cup x_3)$ of the disjunctive formula

Our contribution is based on the following two remarks:

1. The *holes* corresponding to incompatible alternatives are not exploited
2. The formula is not simplified during branching

This is due to taking the hull of the alternatives as the domain of a disjunction. The reason for taking the hull is well-known: handling the union of disjoint intervals as the domain of a single variable is too computationally expensive. However, when performing the branching operation in a Branch&Prune algorithm, it is possible to exploit these disjunctions.

3 Disjunctive multi-split principle

We propose to overcome these two weaknesses by defining interesting splitting points so that the holes are exploited and the formula is simplified. These interesting splitting points are defined by the bounds of each individual alternative

projection : each bound is a splitting point. Hence, the domain of a variable is split into at most $2 \times k$ subdomains if the disjunction contains k alternatives. In our example (see Fig. 1), this splitting results in 4 subdomains for x: x_2, $x_3 - x_1$, $x_1 \cap x_3$, $x_1 - x_3$. The hole is eliminated. Moreover, in each of these subdomains, the problem can be simplified: in x_2, the problem is reduced to formula f_2; in $x_3 - x_1$ to f_3; in $x_1 \cap x_3$ to $f_1 \vee f_3$; and in $x_1 - x_3$ to f_1. Hence, in this example, the disjunction has been completely eliminated in 3 of the 4 produced subdomains.

Selecting the variable domain to split and the disjunction to use for this operation is certainly the key problem to obtain good results using this disjunctive multi-split principle. For this reason, we propose as a first approximation to consider the conjunctive normal form (CNF) of a formula[1], i.e. a conjunction of disjunctions of numerical constraints. Hence, to apply the disjunctive multi-split principle, we have to select a clause (a disjunction) and a variable in this clause.

4 Experimental results

To assess the merits of the disjunctive multi-split principle, we have implemented a generator of first-order NCSPs in CNF. Indeed, not much first-order NCSPs benchmarks are available, and most of the available ones involve only very few disjunctions [8]. We suspect this is due to the lack of disjunction handling techniques that yield people to model otherwise naturally disjunctive problems in a purely conjunctive way[2].

Our generator produces only satisfiable problems composed of polynomial equations. Without detailing, it is based on algebraic principles and the use of ideals which are the prescribed solutions. The generator takes as parameters the number of variables, their domains, the number of solutions, the number of clauses and the number of constraints per clause. In our benchmarks, we have used only problems with initial domains $[-50; 50]$, hence a class of problem is fully defined by a quadruple (s, v, c, d) where s is the number of solutions, v is the number of variables, c is the number of constraints per clause and d is the number of clauses.

We have implemented the pruning operation as described in Ratschan's framework, and integrated it with three different branching operations: bisection, trisection and disjunctive multi-split. Bisection, the classical branching operation, is used in a round-robin strategy. Trisection has been implemented to testify that the gains observed using the disjunctive multi-split do not come solely from the fact that the produced subdomains are smaller; it is also used in a round-robin strategy. Finally, the multi-split principle is used in a random selection mode, i.e., the clause and the variable are selected at random. Indeed, it is not so trivial to define a round-robin strategy for the disjunctive multi-split since not all variables may occur in all clauses. The results are depicted in

[1] Since any (quantifier-free) first-order logical formula can be rewritten in CNF, the principle however applies to any first-order NCSP.

[2] Expressing the disjunctions as integer or boolean variables for instance.

Tab. 1 where each case contains the average of five run on different problems in the same class.

(s, v, c, d)	bisection	trisection	multi-split	multi-split2
$(3, 5, 3, 10)$	TO	TO	44	99
$(3, 7, 3, 30)$	TO	TO	205	TO
$(5, 5, 5, 30)$	TO	TO	39	139
$(10, 3, 10, 30)$	33	119	2	3
$(10, 5, 10, 30)$	126	TO	126	TO

Table 1. times in second for the complete solving on an Intel Pentium 2.80GHz with 1024 Mb ram, TO stands for computation times interrupted after 10 minutes

The results show that the disjunctive multi-split principle pays-off in average: the computation times are reduced by a factor of 10 or more in many cases. This gain is not due solely to the smaller size of the subdomains since trisection is often worse than bisection.

The last columns in Tab. 1, entitled *multi-split2*, correspond to the use of a heuristic that selects the clause and the variable maximizing the cumulated width of the eliminated holes. This heuristic does not fare better than selecting the clause and variable randomly. This indicates that the exploitation of the holes does not dominate the importance of simplifying the formula. This is in line with related work that tried to exploit holes in conjunctive NCSPs [1].

Heuristics that allow to select accurately a good pair (clause, variable) are still to be defined. We expect that classical variable selection heuristics may not be applicable straight-forward due to the disjunctive nature of the problem. Putting the principle in practice, and not only for CNF formulas, is our next objective.

References

1. H. Batnini, C. Michel, and M. Rueher. Mind the gaps: A new splitting strategy for consistency techniques. In *CP*, 2005.
2. Esther Gelle and Boi Faltings. Solving mixed and conditional constraint satisfaction problems. *Constraints*, 8(2):107–141, April 2003.
3. P. Van Hentenryck. *Constraint satisfaction in logic programming.* MIT Press, 1989.
4. Olivier Lhomme. An efficient filtering algorithm for disjunction of constraints. In *CP*, pages 904–908, 2003.
5. S. Mittal and B. Falkenhaimer. Dynamic constraint satisfaction problems. In *AAAI*, pages 25–32, 1990.
6. Stefan Ratschan. Continuous first-order constraint satisfaction. In *Artificial Intelligence, Automated Reasoning, and Symbolic Computation*, number 2385 in LNCS. Springer, 2002.
7. Stefan Ratschan. Continuous first-order constraint satisfaction with equality and disequality constraints. In Pascal van Hentenryck, editor, *Proc. 8th International Conference on Principles and Practice of Constraint Programming*, number 2470 in LNCS, pages 680–685. Springer, 2002.

8. Stefan Ratschan. Applications of quantified constraint solving - bibliography and benchmark problems, 2006.
9. D. Sabin and E. Freuder. Configuration as composite constraint satisfaction. In *Artificial Intelligence and Manufacturing Research Planning Workshop*, pages 153–161, 1996.