Neighborhood-Based Variable Ordering Heuristics for the Constraint Satisfaction Problem

Christian Bessiere, Assef Chmeiss, Lakhdar Saïs

Neighborhood-Based Variable Ordering Heuristics for the Constraint Satisfaction Problem*

Christian Bessière[1], Assef Chmeiss[2], and Lakhdar Saïs[2]

[1] Member of the Coconut group
LIRMM-CNRS (UMR 5506), 161, rue Ada, 34392 Montpellier Cedex 5, France
bessiere@lirmm.fr
[2] CRIL - Université d'Artois - IUT de Lens
Rue de l'université - SP 16, 62307 LENS Cedex, France
{chmeiss, sais}@cril.univ-artois.fr

Abstract. One of the key factors in the efficiency of backtracking algorithms is the rule they use to decide on which variable to branch next (namely, the variable ordering heuristics). In this paper, we give a formulation of dynamic variable ordering heuristics that takes into account the properties of the neighborhood of the variable.

1 Introduction

Constraint satisfaction problems (*CSPs*) are widely used to solve combinatorial problems appearing in a variety of application domains. They involve finding solution in a *constraint network*, i.e., finding *values* for network *variables* subject to *constraints* on which combinations are acceptable.

The usual technique to solve CSPs is systematic backtracking. But if we want to tackle highly combinatorial problems, we need to enhance this basic search procedure with clever improvements. An improvement that has shown to be of major importance is the ordering of the variables, namely, the criterion under which we decide which variable will be the next to be instantiated. Many variable ordering heuristics for solving CSPs have been proposed over the years. However, the criteria used in those heuristics to order the variables are often quite simple, and concentrate on characteristics inherent to the variable to be ordered, and not too much on the influence its neighborhood could have. Those that use more complex criteria, essentially based on the constrainedness or the solution density of the remaining subproblem, need to evaluate the tightness of the constraints, and so, need to perform many constraint checks.

The goal of this paper is to propose heuristics that take into account properties of the neighborhood in the criterion of choice of a variable, while remaining free of any constraint check.

* This work was partially supported by the IUT of Lens, the Nord/PasdeCalais region, and the European community.

2 Preamble

Numerous criteria have been proposed to find good variable orderings for backtrack search procedures. Among them, dynamic[1] variable orderings (DVOs) have always shown better average performances than static ones. In [4], Haralick and Elliott introduced `dom`, the DVO choosing as next variable the one with the smallest remaining domain.

Since `dom` can be completely fooled by the structure, especially at the beginning of the search, when domains have more chances to be of equal size, other heuristics have been proposed. `dom+futdeg` is the one derived from the Brélaz heuristic (proposed for graph coloring) [3]. It breaks ties in `dom` by preferring the variable with highest future degree [6]. Smith also improved `dom+futdeg` by adding to it a second and a third tie breakers, namely, the size of the smallest neighbor, and the number of triangles in which the first chosen variable is involved. She called this DVO `BZ3`.

However, both `dom+futdeg` and `BZ3` use the domain size as the main criterion. The degree of the variables is considered only in case of ties, which can again fool the heuristic. Combined heuristics [2] do not give priority to the domain size or degree of variables, but use them equally in the criterion. `DD` chooses as the next variable, the variable X_i minimizing the ratio "size of domain/degree". `DD` has extensively been studied in [5].

To give an insight into the state of the art, we performed experiments with some of these well-known heuristics: `dom`, the oldest and most well known DVO, `DD` and `BZ3`, the best current VOs for CSPs [5,6]. On random problems with 10 values per domain, an average degree of 5 (i.e., 5/2 times more constraints than variables), and the tightness fixed at the cross-over point (which is stable at 55 forbidden tuples per constraint), we increased the number N of variables by steps of 10, and could see the following:[2]

- when $N = 110$, `DD` needs less than 1 sec., `BZ3` less than 10 sec., and `dom` less than 100 sec.,
- when $N = 120$, `dom` goes above 100 sec.,
- when $N = 150$, `DD` needs less than 10 sec., and `BZ3` less than 100 sec.,
- when $N = 160$, `BZ3` goes above 100 sec.,
- when $N = 210$, `DD` goes above 100 sec.

3 Multi-level DVOs

One of the key features for the efficiency of a backtrack search method lies in its branching strategy. At each step of the search process, a problem P is reduced into a finite number of sub-problems $(P_1, P_2, \ldots, P_{|D(X_i)|})$, where X_i is

[1] A variable ordering is dynamic when it can change the order of the variables from one branch to the other.

[2] These experiments have been run on a PC Pentium III 667 MHz under Linux. 100 instances for each value of the parameters.

the chosen variable. Following ideas developed for the DP procedure on SAT, we think that a good DVO should reduce both the *number* and the *difficulty* of such subproblems.

We propose a general formulation of DVOs which integrates in the selection function a measure of the constrainedness of the given variable. The constrainedness of a variable can be defined as a function of the constraints involving the variable. One could choose semantical constraints-based measures (e.g., number of allowed tuples) or syntactical ones (e.g., size of the Cartesian product of the domains). Choosing the most constrained variable should have a great impact on the search space, leading the search to the most constrained parts of the CSP, and thus provoking early detection of local inconsistencies.

3.1 A General Criterion Free of Constraint Checks

From now on, we will denote by $\Gamma(X_i)$ the set of variables sharing a constraint with the variable X_i. Let us first define $W(R_{ij})$ as the weight of the constraint R_{ij} and,

(1) $W(X_i) = \frac{\sum_{X_j \in \Gamma(X_i)} W(R_{ij})}{|\Gamma(X_i)|}$

as the mean weight of the constraints involving X_i. In order to maximize the number of constraints involving a given variable and to minimize the mean weight of such constraints, the next variable to branch on should be chosen according to the minimum value of

(2) $H(X_i) = \frac{W(X_i)}{|\Gamma(X_i)|}$

over all uninstantiated variables (numerator to minimize the weight, and denominator to maximize the number of constraints).

For reasons of efficiency of computation, the weight we will associate to a constraint must be something cheap to compute (e.g., free of constraint checks). It can be defined by $W(R_{ij}) = \alpha(X_i) \circledcirc \alpha(X_j)$, where $\alpha(X_i)$ is instantiated to a simple syntactical property of the variable such as $|D(X_i)|$ or $\frac{|D(X_i)|}{|\Gamma(X_i)|}$, and $\circledcirc \in \{+, \times\}$. For $\alpha(X_i) = |D(X_i)|$, and $\circledcirc = \times$, the weight associated to a given constraint R_{ij} represents an upper bound of the number of tuples allowed by R_{ij}.

We obtain the new formulation of (2):

(3) $H^{\circledcirc}_{\alpha}(X_i) = \frac{\sum_{X_j \in \Gamma(X_i)} (\alpha(X_i) \circledcirc \alpha(X_j))}{|\Gamma(X_i)|^2}$

3.2 Multi-level Generalization

In the formulation of the DVOs presented above, the evaluation function $H(X_i)$ considers only the variables at distance one from X_i (first level or neighborhood). However, when arc consistency is maintained (MAC), the instantiation of a value

to a given variable X_i could have an immediate effect not only on the variables of the first level, but also on those at distance greater than one.

To maximize the effect of such a propagation process on the CSP, and consequently to reduce the difficulty of the subproblems, we propose a generalization of the DVO $H^{\circledcirc}_{\alpha}$ such that variables at distance k from X_i are taken into account. This gives what we call a "multi-level DVO", $H^{\circledcirc}_{(k,\alpha)}$. To obtain this multi-level DVO, we simply replace $\alpha(X_j)$ in formula (3) by a recursive call to $H^{\circledcirc}_{(k-1,\alpha)}(X_j)$. This means that to compute $H^{\circledcirc}_{(k,\alpha)}$ on a given variable, we need to compute $H^{\circledcirc}_{(k-1,\alpha)}$ on all its neighbors, and so on. The recursion terminates with $H^{\circledcirc}_{(0,\alpha)}$, equal to α. This is formally stated as follows:

(4) $H^{\circledcirc}_{(0,\alpha)}(X_i) = \alpha(X_i)$

(5) $H^{\circledcirc}_{(k,\alpha)}(X_i) = \frac{\sum_{X_j \in \Gamma(X_i)} (\alpha(X_i) \circledcirc H^{\circledcirc}_{(k-1,\alpha)}(X_j))}{|\Gamma(X_i)|^2}$

In the following, $H^{\circledcirc}_{(0,dom)}$ and $H^{\circledcirc}_{(0,DD)}$ are denoted by their classical name, `dom` and `DD`, respectively. $H^{\circledcirc}_{(k,dom)}$ and $H^{\circledcirc}_{(k,DD)}$ are denoted by `H_k_dom_`$\circledcirc$ and `H_k_DD_`$\circledcirc$ respectively.

4 Experiments

We have compared experimentally the behavior of the new DVOs defined above (and others) on several classes of random CSPs and on real instances from the FullRLFAP archive[3]. We give here a brief snapshot of the results. Extensive experiments can be found in [1]. In all our experiments, we stopped search after the first solution is found. The search procedure used maintains arc consistency (MAC).

We ran the `H_1_DD_`$\circledcirc$ ($\circledcirc \in \{+, \times\}$) on the experiment described in Section 2. The gap between `H_1_DD_`$\circledcirc$ and `DD` grows with N. At $N = 230$, `H_1_DD_`$\times$ is more than 5 times faster than `DD`, which was by far the best DVO known so far. We performed other experiments, fixing the number of variables to 100, and increasing the number of constraints in the network. `H_1_DD_`$\times$ becomes better and better when density grows. (As opposed to `H_1_DD_`$+$ which was even better than `H_1_DD_`$\times$ on sparse problems, but which becomes slower on denser problems.)

If we increase the number of variables or the domain size, the gain of the $H^{\circledcirc}_{(1,\alpha)}$ heuristics continues to grow compared to `DD`.

As a synthesis of the results on different classes of random CSPs, we can say that, except `H_1_dom_`$\times$, the first level DVOs improve significantly `DD`. Furthermore, in general, `H_1_DD_`$\circledcirc$ are better than `H_1_dom_`$\circledcirc$. This is not surprising because the former take into account the connectivity of the neighborhood of the chosen variable.

[3] We thank the Centre d'Electronique de l'Armement (France).

We also compared the behavior of these DVOs on the real instances of the FullRLFAP archive. Since these are optimization problems, we built a series of satisfaction problems for each instance of optimization problem. In the table below, we report results for all instances on which a significant difference has been observed among the DVOs tested. The cpu time limit was put to one hour on a PC Pentium II 300MHz.

	scen11-01234 (sat)		scen06-012 (unsat)		scen02-24 (sat)		scen02-25 (unsat)	
	#nodes	time (sec.)	#nodes	time (sec.)	#nodes	time (sec.)	#nodes	time (sec.)
DD	6,019	8.43	—	> 1 h.	31,308,876	2,296.61	—	> 1 h.
H_1_dom_+	21,156	29.68	41	0.40	663	0.32	—	> 1 h.
H_1_DD_+	—	> 1 h.	41	0.41	—	> 1 h.	11,668	10.18
H_1_dom_×	12,517	16.99	—	> 1 h.	677	0.32	—	> 1 h.
H_1_DD_×	226,011	337.55	41	0.41	—	> 1 h.	8,529	6.99

No conclusion can be drawn on a so small number of pertinent instances. However, it seems that H_1_DD_◎ are better on inconsistent problems, and H_1_dom_◎ on satisfiable ones. But, more extensive tests should be run to draw definite conclusions.

5 Conclusion

In this paper, a general formulation of dynamic variable ordering heuristics has been proposed. It admits numerous advantages,

- the constrainedness of a given variable is computed without any constraint check, thanks to simple syntactical properties,
- it takes advantage of the neighborhood of the variable, with the notion of distance as a parameter,
- it can be instantiated to different known variable ordering heuristics,
- it is possible to use other functions to measure the weight of a given constraint.

References

1. C. Bessière, A. Chmeiss, and L. Saïs. Neighborhood-based variable ordering heuristics for the constraint satisfaction problem. Technical Report 01002, LIRMM – University of Montpelllier II, Montpellier, France, January 2001. (available at http://www.lirmm.fr/~bessiere/).
2. C. Bessière and J.C. Régin. MAC and combined heuristics: two reasons to forsake FC (and CBJ?) on hard problems. In *Proceedings CP'96*, pages 61–75, Cambridge MA, 1996.
3. D. Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22:251–256, 1979.
4. R.M. Haralick and G.L. Elliott. Increasing tree seach efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–313, 1980.
5. B. Smith and S.A. Grant. Trying harder to fail first. In *Proceedings ECAI'98*, pages 249–253, Brighton, UK, 1998.
6. B.M. Smith. The Brélaz heuristic and optimal static orderings. In *Proceedings CP'99*, pages 405–418, Alexandria VA, 1999.