VALCSP solver : a combination of Multi-Level Dynamic Variable Ordering with Constraint Weighting

Assef Chmeiss, Lakdar Saïs, Vincent Krawczyk

CRIL - University of Artois - IUT de Lens
Rue Jean Souvraz - SP 18
62307 LENS Cedex 3, France
{chmeiss, sais, krawczyk}@cril.univ-artois.fr

Abstract. The usual way for solving constraint satisfaction problems is to use a backtracking algorithm. One of the key factors in its efficiency is the rule it will use to decide on which variable to branch next (namely, the variable ordering heuristics). In this paper, we propose one binary CSPs solver: VALCSP. He uses and combines two powerful and complementary heuristics. The first one [BCS01] is a look-ahead based heuristic called multi-level variable ordering heuristic, a variable is selected according to a measure that takes into account the properties of the neighborhood of the given variable. The second one [BHLS04] is based on constraint weighting (*Wdeg*). More precisely, a higher weight is given to constraints violated at some previous steps of the search process. Such weighted constraints are used to guide the dynamic variable ordering heuristics of a backtrack search-like algorithms. Our solver is based on the well known MAC algorithm. Arc-consistency is maintained using the AC8 algorithm. In this paper, we give a description of our solver presented to the second International CSP Solver Competition.

1 Introduction

Constraint satisfaction problems (*CSPs*) are widely used to solve combinatorial problems appearing in a variety of application domains. The usual technique to solve CSPs is the systematic backtracking. It repeatedly chooses a variable, attempts to assign it one of its values, and then goes to the next variable, or backtracks in case of failure. This technique is at the basis of almost all the CSP solving engines. But if we want to tackle highly combinatorial problems, we need to enhance this basic search procedure with clever improvements.

A crucial improvement to be added is look-ahead value filtering, which consists in removing from future domains values that cannot belong to a solution extending the current partial instantiation. Many works have studied the different levels of filtering that can be applied at each node of the search tree. Two famous algorithms maintaining different levels of consistency at each node are forward checking (FC), and maintaining arc consistency (MAC). Several papers

have discussed their performances [SF94,BR96,GS96]. Our solver uses MAC as a search algorithm with the AC8 [CJ98] arc consistency algorithm.

A second kind of improvement that can be added to a backtrack search procedure is to use the knowledge obtained from deadends to avoid future failures coming from the same reason. Backjumping based algorithms [Pro93] are the most famous of these "look back" techniques. In [BHLS04], a simple and efficient criterion is used to direct the search on the most hard and probably inconsistent subpart of the CSP is proposed. It selects the next variable to assign according to its occurrence in the most violated constraint during search. This heuristic is originally proposed in [BGS99] for solving the satisfiability problem.

Another improvement that has been shown to be of major importance is the ordering of the variables (VO), 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 concentrated on the characteristics inherent to the variable to be ordered, and not too much on the influence its neighborhood could have. Those that used 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.

Our VALCSP solver aims to use the influence of the neighborhood in the criterion of choice of a variable [BCS01], while remaining free of any constraint check. It, also, combines this heuristic with constraint weight.

2 Definitions and notations

A constraint network is defined by a set of variables $\mathcal{X}$, each taking values in its finite domain $D_i \in \mathcal{D}$, and a set of constraints $\mathcal{C}$ restricting the possible combinations of values between variables. The set of variables implied in a constraint c will be denoted by Vars(c).

Any constraint network can be associated with a *constraint graph* in which the nodes are the variables of the network, and an edge links nodes if and only if there is a constraint on the corresponding variables.

$\Gamma_{init}(x_i)$ denotes the set of nodes sharing an edge with the node x_i (its initial neighbors). We define the set $\Gamma(x_i)$ as the current neighborhood of x_i, namely, the neighbors remaining uninstantiated once a backtracking search procedure has instantiated the set $Y = \{X_{i_1}, \ldots, X_{i_k}\}$ of variables, i.e., $\Gamma(x_i) = \Gamma_{init}(x_i) - Y$. The size of $\Gamma(x_i)$ (resp. $\Gamma_{init}(x_i)$) is called the *degree* (resp. initial degree) of x_i.

3 Search heuristic

In the VALCSP solver, we combine the Multi-Level ordering heuristic and the Wdeg heuristic. This solver is based on the MAC algorithm which uses the following skills:

– the AC8 [CJ98] arc consistency algorithm is used to maintain arc consistency during search.
– The multi-level variable ordering [BCS01] is the basic heuristic used to choose the next variable to assign (defined in 3.1)

Let us recall the notion of Multi-Level ordering heuristic proposed in [BCS01] and the Wdeg heuristic proposed in [BHLS04].

3.1 Multi-Level variable ordering heuristics

Let us first define $W(C_{ij})$ as the weight of the constraint C_{ij} and,
(1) $W(x_i) = \frac{\sum_{x_j \in \Gamma(x_i)} W(C_{ij})}{|\Gamma(x_i)|}$
as the mean weight of the constraints involving x_i. In order to maximize the number of constraint 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.

For complexity reasons, 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 $W(C_{ij}) = \alpha(x_i) \circledcirc \alpha(x_j)$, where $\alpha(x_i)$ is instantiated to a simple syntactical property of the variable such as $|D_i|$ or $\frac{|D_i|}{|\Gamma(x_i)|}$, and $\circledcirc \in \{+, \times\}$.

We obtain the new formulation of (2):
(3) $H^{\circledcirc}_{\alpha}(x_i) = \frac{\sum_{x_j \in \Gamma(x_i)} \alpha(x_i) \ \circledcirc \ \alpha(x_j)}{|\Gamma(x_i)|^2}$

Multi-level generalization In the formulation of the dynamic variable orderings (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 $H^{\circledcirc}_{(k-1,\alpha)}$. 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) $H^{\circledcirc}_{(k,\alpha)}(x_i) = \frac{\sum_{x_j \in \Gamma(x_i} \alpha(x_i)) \circledcirc H^{\circledcirc}_{(k-1,\alpha)}(x_i)}{|\Gamma(x_i)|^2}$

3.2 Wdeg heuristic

The main goal behind the `Wdeg` heuristic is to exploit informations about previous step of the search and to direct the search to the most constrained sub-

problem. More precisely, a counter, called $W(C_{ij})$, with any constraint C_{ij} of the problem. These counters will be updated during search whenever a dead-end (domain wipeout) occurs. As systematic solvers such as FC or MAC involve successive revisions of variables in order to remove values that are no more consistent with the current state, it suffices to introduce a test at the end of each revision. If the constraint under test is violated, its counter is increased by one.

Using these counters, it is possible to define a new variable ordering heuristic, denoted `Wdeg`, that gives an evaluation $H_{wdeg}(x_i)$, called weighted degree, of any variable x_i as follows:

$$H_{wdeg}(x_i) = \sum_{x_j \in \Gamma(x_i)} W(C_{ij})$$

3.3 VALCSP solver : combining multi-level DVO with Constraint weighting

Our `VALCSP` solver combines both Multi-Level and Constraint weighting heuristics. It defined as following:

$$H_{lw}(x_i) = \frac{|D_i| + \sum_{x_j \in \Gamma(x_i)} |D_j|}{W(C_{ij})}.$$

To compute $H_{lw}(x_i)$, we consider for each constraint C_{ij}, the ratio between the sum of the domains size of all x_j (and x_i) and the weight $W(C_{ij})$. A sum is calculated on all the constraints involving a given variable x_i.

4 Filtering algorithm

A preprocessing step achieves arc consistency on the CSP. We also maintain arc consistency during search with the MAC algorithm.

The main arc consistency algorithm used in our solvers is AC8. It's proposed by Chmeiss and Jegou in [CJ98].
AC8 is based on supports but without recording them. When a value $a \in D_i$ is removed from its domain, AC8 records the reference of the variable x_i, that is the number i, in the list of propagation denoted *List-AC*. Propagations will be realized with respect to variables in this list. Suppose that a variable x_i is removed from the list. Then, all neighboring variables will be considered, i.e. for all $x_j \in \mathcal{X}$ such that $C_{ji} \in \mathcal{C}$, and for each value $b \in D_j$, AC8 will ensure that there is a value $a \in D_i$ such that $(a, b) \in R_{ij}$ holds. Unlike AC6, AC8 has to start again the search from the first value of the domains. If no support a of b is found in D_i, then b must be deleted, and the number of the variable, namely j must be inserted in *List-AC*. To ensure that j is not duplicated in *List-AC*, we must maintain an array of booleans, denoted *Status-AC*, recording the status of variables. So, the data structures used to AC8 are the *List-AC* containing variables which have lost some values in their domain and not propagated yet, and the boolean table *Status-AC* that always verifies $\{i \in$ *List-AC* $<=>$ *Status-AC*$[i]\}$.

5 Summary: submitted solver

In this paper, we have given a description of the solver which we submit to the second International CSP Solver Competition. This solver is slightly different. It uses the AC8 algorithm to maintain arc-consistency during search. It combines Multi-Level and constraint weighting heuristics (see sections 4.2 and 4.3)

Finally, we mention that this solver is implemented using the C ANSI programming language.

References

[BCS01] C. Bessière, A. Chmeiss, and L. Sais. Neighborhood-based variable ordering heuristics for the constraint satisfaction problem. In *Proceedings of CP'01*, pages 565–569, 2001.

[BGS99] L. Brisoux, E. Gregoire, and L. Sais. Improving backtrack search for sat by means of redundancy. In *Proceedings of ISMIS'99*, pages 301–309, 1999.

[BHLS04] F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais. Boosting systematic search by weighting constraints. In *Proceedings of ECAI'04*, pages 146–150, 2004.

[BR96] C. Bessiere and J-C. Regin. MAC and combined heuristics: Two reasons to forsake FC (and CBJ?) on hard problems. In *Principles and Practice of Constraint Programming*, pages 61–75, 1996.

[BvB98] F. Bacchus and P. van Beek. On the conversion between non-binary and binary constraint satisfaction problems. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, pages 311–318, Menlo Park, 26–30 1998. AAAI Press.

[CJ98] A. Chmeiss and P. Jégou. Efficient path-consistency propagation. *International Journal on Artificial Intelligence Tools*, 7(2):121–142, 1998.

[Dec90] R. Dechter. On the expressiveness of networks with hidden variables. In *Proceedings of AAAI'90*, pages 556–562, Boston MA, 1990.

[GS96] S.A. Grant and B.M. Smith. The phase transition behavior of maintaining arc consistency. In *Proceedings of ECAI'96*, pages 175–179, 1996.

[Pie33] C.S. Pierce. *Collected Papers, Vol III.* Harward University Press, Cambridge, 1933.

[Pro93] P. Prosser. Hybrid algorithms for the constraint satisfaction problems. *Computational Intelligence*, 9(3):268–299, 1993.

[SF94] D. Sabin and E.C. Freuder. Contradicting Conventional Wisdom in Constraint Satisfaction. In Alan Borning, editor, *Proceedings of the Second International Workshop on Principles and Practice of Constraint Programming, PPCP'94, Rosario, Orcas Island, Washington, USA*, volume 874, pages 10–20, 1994.