Max-CSP competition 2006: `toolbar/toulbar2` solver brief description

S. Bouveret[3], S. de Givry[1], F. Heras[2], J. Larrosa[2], E. Rollon[2], M. Sanchez[1], T. Schiex[1], G. Verfaillie[3], and M. Zytnicki[1]

[1] INRA, Toulouse, France
[2] Dep. LSI, UPC, Barcelona, Spain
[3] ONERA, Toulouse, France

Abstract. This document gives a brief description of the key techniques used in four different versions of `toolbar/toulbar2` solvers submitted to the Max-CSP competition 2006.

All the solvers exploit an initial upper bound found by a local search solver : `maxwalksat` [13] (with 5 tries) for `toolbar/MaxSAT` and `INCOP`[4] [11] for the other solvers. The solvers are implemented in C code, except for `toulbar2` in C++[5].

`toolbar`

The search procedure is *MEDAC** [3], a branch and bound algorithm which maintains the state-of-the-art soft local consistency property *EDAC** during the search. The local consistency enforcement procedure is *à la AC-2001* as described in [9] and it uses specific data structures for efficient binary constraint updating as introduced in [1]. The usual *min domain / max degree* dynamic variable ordering heuristic is employed during the search. Domain values are dynamically ordered by increasing associated unary costs for value enumeration at each node of the search tree.

No particular options are used, except in a preprocessing step where constraints of arity smaller than 10 are projected on binary constraints. Notice that non-binary constraints are delayed from propagation during the search until they become binary.

`toolbar/BTD`

The search procedure is *EDAC-BTD+* [2], a branch and bound algorithm which exploits the problem structure given by a tree decomposition. EDAC-BTD+ extends *BTD* [5] by exploiting local initial upper bounds inside the clusters

[4] The command line parameters are `narycsp result problem.wcsp 0 1 5 idwa 100000 cv v 0 200 1 0 0`.

[5] The solvers are available at the *AlgorithmS* section of `http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/SoftCSP`

and maintaining EDAC* during the search instead of partial forward checking (EDAC* is restricted to the current cluster subtree in order to guarantee time complexity proportional to the tree width).

The *min-degree* heuristic is used to compute a tree decomposition. The root of the tree decomposition is the cluster that minimizes the tree height. Ties are broken by selecting the cluster whose product of domain sizes is maximal. Cluster separators larger than 5 are removed, by merging clusters. The same preprocessing as in `toolbar` for non-binary constraints is performed. The variable ordering heuristic is dynamic (*min domain / max degree*) inside the clusters and follows a compatible order with respect to the cluster tree decomposition. All the variables of a cluster are assigned before its children are examined (in lexicographic order). A hash-table with initial size of 2^{20} is used to memorize cluster lower bounds for pruning and partial solutions for recovering an optimal solution. The value ordering heuristic chooses the last value in the best solution found so far before sorting values by increasing unary costs.

`toolbar/MaxSAT`

The search procedure is *Max-DPLL* [7], a branch and bound algorithm dedicated to Weighted Max-SAT. Max-DPLL is enhanced by several inference rules : *neighborhood resolution* (equivalent to soft AC* in Weighted CSPs), *chain resolution* restricted to binary clauses (equivalent to soft DAC* in Weighted CSPs but with a dynamic DAC ordering), and *cycle resolution* with cycles of triplets of variables, initially proposed in [6]. *Two-sided Jeroslow* dynamic variable ordering heuristic is used during the search.

The usual direct encoding (one Boolean variable per value for non-Boolean variables) is used to convert Weighted CSP instances into Weighted Max-SAT.

`toulbar2`

The search procedure extends the one in `toolbar`, i.e. MEDAC*, in several ways:

- EDAC* also propagates soft ternary constraints as defined in [12].
- The variable ordering heuristic combines a basic form of conflict back-jumping [10] with the usual *min domain / max degree.*
- A limited form of variable elimination (for variables with a degree less than or equal to 2) is applied during the search as proposed in [8].
- The search procedure exploits a binary branching scheme instead of value enumeration. Two different kinds of branching schema are used depending on the domain size of the current chosen variable. If the domain size is greater than 10, then the domain is split into two equal-size parts, creating two new search nodes. Otherwise, the chosen variable is assigned to its *fully supported value* (maintained by EDAC*) or this value is removed from its domain.
- A Limited Discrepancy Search [4] scheme is performed by iteratively running MEDAC* with a power-of-two increasing limit in the number of discrepancies to the value ordering heuristic until optimality proof has been obtained (when no limit occurred).

The first extension yields better lower bounds, especially on the *pedigree* benchmark. The second and third extensions exploit the problem structure better. The forth extension improves propagation and heuristics, especially on problems with large domains as in the *celar* benchmark. The last extension allows to find better upper bounds more rapidly. However, this last option may be counter-productive when a good initial upper bound has been already found by local search as it slows down the time to prove optimality (mainly by a factor of two approximatively).

References

[1] M. Cooper and T. Schiex. Arc consistency for soft constraints. *Artificial Intelligence*, 154:199–227, 2004.

[2] S. de Givry, T. Schiex, and G. Verfaillie. Exploiting Tree Decomposition and Soft Local Consistency in Weighted CSP. In *Proc. of AAAI-06*, Boston, MA, 2006.

[3] S. de Givry, M. Zytnicki, F. Heras, and J. Larrosa. Existential arc consistency: Getting closer to full arc consistency in weighted CSPs. In *Proc. of IJCAI-05*, pages 84–89, Edinburgh, Scotland, 2005.

[4] W. D. Harvey and M. L. Ginsberg. Limited discrepency search. In *Proc. of the 14^{th} IJCAI*, Montréal, Canada, 1995.

[5] P. Jégou and C. Terrioux. Decomposition and good recording. In *Proc. of ECAI-2004*, pages 196–200, Valencia, Spain, 2004.

[6] J. Larrosa and F. Heras. New Inference Rules for Efficient Max-SAT Solving. In *Proc. of AAAI-06*, Boston, MA, August 2006.

[7] J. Larrosa, F. Heras, and S. de Givry. A logical approach to efficient max-sat solving. *Artificial Intelligence*, 172(2–3):204–233, 2008.

[8] J. Larrosa, E. Morancho, and D. Niso. On the practical applicability of bucket elimination: still-life as a case study. *Journal of Artificial Intelligence Research*, 23:421–440, 2005.

[9] J. Larrosa and T. Schiex. Solving Weighted CSP by Maintaining Arc-consistency. *Artificial Intelligence*, 159(1-2):1–26, 2004.

[10] C. Lecoutre, L. Sais, S. Tabary, and V. Vidal. Last conflict based reasoning. In *Proc. of ECAI-2006*, pages 133–137, Trento, Italy, 2006.

[11] B. Neveu and G. Trombettoni. INCOP: An Open Library for INcomplete Combinatorial OPtimization. In *Proc. of CP-03*, pages 909–913, Cork, Ireland, 2003.

[12] M. Sanchez, S. de Givry, and T. Schiex. Mendelian error detection in complex pedigrees using weighted constraint satisfaction techniques. *Constraints*, 13(1), 2008. Special issue on Bioinformatics and Constraints.

[13] B. Selman, H. Kautz, and B. Cohen. Noise Strategies for Improving Local Search. In *Proc. of AAAI-94*, pages 337–343, Seattle, WA, 1994.