Computing and Exploiting Tree-Decompositions for Solving Constraint Networks

Philippe Jégou, Samba Ndojh Ndiaye, and Cyril Terrioux

LSIS - UMR CNRS 6168,
Université Paul Cézanne (Aix-Marseille 3),
Avenue Escadrille Normandie-Niemen,
13397 Marseille Cedex 20, France
{philippe.jegou, samba-ndojh.ndiaye, cyril.terrioux}@univ.u-3mrs.fr

Abstract. Methods exploiting tree-decompositions seem to provide the best approach for solving constraint networks w.r.t. the theoretical time complexity. However, they have not shown a real practical interest yet. In this paper, we study several methods for computing a rough optimal tree-decomposition and assess their relevance for solving CSPs.

1 Introduction

The CSP formalism (Constraint Satisfaction Problem) offers a powerful framework for representing and solving efficiently many problems. A CSP instance is defined by a tuple (X, D, C). X is a set $\{x_1, \ldots, x_n\}$ of n variables. Each variable x_i takes its values in a finite domain from D. The variables are subject to the constraints from C which express restrictions between the different possible assignments. Given an instance (X, D, C), the CSP problem, which consists in determining whether a solution (i.e. an assignment of each variable which satisfies each constraint) exists, is NP-complete. In this paper, without loss of generality, we only consider binary constraints (i.e. constraints which involve two variables). So, the structure of a CSP can be represented by the graph (X, C), called the *constraint graph*.

The usual method for solving CSPs is based on backtracking search. This approach, often efficient in practice, has an exponential theoretical time complexity in $O(e.d^n)$ for an instance having n variables and e constraints and whose largest domain has d values. Several works have been developed to improve this theoretical complexity bound thanks to particular features of the instance. The best known bound is given by the "tree-width" w of a CSP. This parameter, related to a tree-decomposition of the constraint graph, leads to a time complexity in $O(n.d^{w+1})$. As $w+1 \leq n$, depending on the instances, we can expect a significant gain w.r.t. enumerative approaches. So several methods have been proposed to reach this bound like *Tree-Clustering* [1] or BTD [2]. Yet, the space complexity, often linear for enumerative methods, is in $O(n.s.d^s)$ with s the size of the largest minimal separators of the graph and so may make such an approach unusable in practice. Hence, most of works based on this approach only present theoretical

P. van Beek (Ed.): CP 2005, LNCS 3709, pp. 777–781, 2005.

results. Except [2,3], no practical results have been provided. Moreover, finding an optimal tree-decomposition (i.e. a tree-decomposition with width w) is NP-Hard [4]. So approximate optimal tree-decompositions (with width w^+ s.t. $w \leq w^+ \leq n-1$) are often exploited. Yet, although this choice is first induced by runtime reasons, we will show that it seems sensible in practice. This paper deals with the computation of a suitable tree-decomposition w.r.t. CSP solving.

An algorithmic way to compute a tree-decomposition relies on *triangulated* graphs [5]. As any graph G is not necessarily triangulated, we can triangulate G. A *triangulation* of G consists in adding a set C' of edges to G s.t. $G' = (X, C \cup C')$ is triangulated. The width of G' is equal to the maximal size of cliques minus one in graph G'. The tree-width of G is then equal to the minimal width over all triangulations. Hence, a rough tree-decomposition can be computed by using a non-optimal triangulation. Many algorithms exist for computing such a triangulation. So, in order to make structural methods efficient, this paper studies and compares some of them w.r.t. to CSP solving. This work is performed by using the BTD method [2] (one of few structural methods which have been implemented and used successfully for practical CSP solving).

By lack of place, the ideas we present are not fully developed and explained. A more complete version of this paper can be found in [6].

2 Triangulation Algorithms

This section raises the problem of computing "good" tree-decompositions thanks to "good" triangulations. Several approaches and algorithms have been proposed for triangulations. In any case, they aim to minimize either the number of added edges, or the size of the cliques in the triangulated graph. We can distinguish four classes of approaches. First, computing an **optimal triangulation** is NP-hard. So no polynomial algorithm is known yet and the proposed algorithms have an exponential time complexity. Secondly, we can exploit **approximation algorithms** which approximate the optimum by a constant factor and whose complexity is often polynomial in the tree-width [7]. Unfortunately, implementing these two first approaches do not have much interest from a practical viewpoint (e.g. the latter is time expensive while obtaining results of poor quality). On the other hand, we can exploit **minimal triangulations**. A minimal triangulation computes a set C' s.t. $(X, C \cup C')$ is triangulated and, for every subset $C" \subset C'$, $(X, C \cup C")$ is not triangulated. Note that a minimal triangulation is not necessarily optimal. The main interest of this approach is related to the existence of polynomial algorithms (e.g. LEX-M [8] and LB [9] whose time complexity is $O(ne')$ with e' the number of edges in the triangulated graph). Finally, the fourth approach, namely **heuristic triangulations**, generally add some edges to the initial graph until the graph is triangulated. They often achieve this work in polynomial time (between $O(n + e')$ and $O(n(n + e'))$) but they do not provide any minimality warranty. Nonetheless, in practice, they can be easily implemented and their interest seems justified. In effect, these heuristics appear to obtain triangulations reasonably close to the optimum [10]. In the following,

we consider two such heuristics: MCS and min-fill. MCS relies on the algorithm of [11] which recognizes the triangulated graphs. Min-fill orders the vertices from 1 to n by choosing as next vertex one which leads to add a minimum of edges when completing the subgraph induced by its unnumbered neighbors.

3 Experimental Study

3.1 Comparison Based on Graphical Criteria

According to the experiments presented in the literature, the two first approaches do not appear very interesting as a first step of a CSP solving method due to a too expensive runtime w.r.t. the weak improvement of the value w^+. Hence, we only assess the interest of the two other approaches by experimenting them on graphs from real-world problems and random structured graphs (partial k-trees).

Table 1 presents empirical results for some graphs of the CALMA archive (real-world frequency assignment problems). We compare four triangulation algorithms, namely n-LEX-M, n-LB, n-min-fill and n-MCS, defined respectively from LEX-M, LB, min-fill and MCS. Precisely, each algorithm n-X fixes the choice of the first vertex and then uses the method X for the remaining vertices. It repeats this process by choosing each vertex as the first vertex. We note that the best results w.r.t. tree-width are performed by n-min-fill and n-LB. However, n-LB offers a more promising trade-off between the runtime and the quality of w^+. These results appear generally better than ones obtained by the MSVS heuristic [12] (based on network flow techniques instead of triangulation). Note that we have observed similar trends on partial k-trees [6].

As an indication, a random choice of the first vertex leads, of course, to worse results. However, these results are often very close to the previous ones. They are obtained in a time divided by n w.r.t. the times provided in table 1. For instance, for CALMA problems, the time does not exceed 2 s.

Table 1. Tree-width obtained after triangulation and triangulation time (in s) for graphs from CALMA archive

Instance	n	e	n-LEX-M		n-LB		n-min-fill		n-MCS	
			w^+	time	w^+	time	w^+	time	w^+	time
CELAR02	100	311	10	0.42	10	0.36	10	0.53	10	0.33
CELAR03	200	721	17	4.71	17	3.71	14	5.78	17	4.32
CELAR06	100	350	11	0.42	11	0.37	11	0.58	11	0.37
CELAR07	200	817	19	4.42	18	3.80	16	6.44	18	4.20
CELAR08	458	1655	20	55.85	19	82.73	16	73.57	19	51.74
CELAR09	340	1130	18	39.96	18	38.89	16	31.43	19	36.36
GRAPH05	100	416	28	1.00	26	0.68	25	1.34	31	0.97
GRAPH06	200	843	58	15.56	53	7.92	54	19.64	58	15.65
GRAPH11	340	1425	106	146.16	90	39.63	91	162.90	104	150.13
GRAPH12	340	1256	99	140.09	85	45.19	85	148.28	96	142.62
GRAPH13	458	1877	146	558.38	120	115.43	126	710.06	131	640.62

Table 2. Runtime (in s) for a value s respectively unlimited and limited to 10. T and M mean that some instances cannot be solved for time reason or for a lack of memory.

Instance	d	t	Time for unlimited s				Time for s limited to 10			
			LEX-M	LB	min-fill	MCS	LEX-M	LB	min-fill	MCS
CELAR02	50	1216	2.72	2.81	2.73	2.80	2.74	2.82	2.72	2.80
CELAR03	30	373	2.22	57.71	M	1.95	2.23	2.60	1.45	1.51
CELAR06	50	1155	3.40	3.52	3.41	3.50	3.43	3.53	3.41	3.48
CELAR07	25	209	12.79	M	13.23	4.66	4.92	4.83	4.86	4.69
CELAR09	25	209	12.47	T	11.82	6.72	4.69	4.88	4.66	4.76

Table 3. Runtime (in s) and value of w^+ for class $(150, 25, 15, t, 5, 15)$ after removing p% edges (with s limited to 5)

p	t	LEX-M		LB		min-fill		MCS	
		w^+	time	w^+	time	w^+	time	w^+	time
10%	215	18.50	5.53	14.00	3.95	15.97	10.66	14.03	4.06
20%	237	22.00	4.07	14.00	4.37	16.33	6.74	14.00	3.53
30%	257	23.30	82.79	14.00	2.85	17.20	5.49	15.03	3.81
40%	285	24.90	78.22	14.00	1.11	15.33	1.21	15.33	5.88

Here, the quality of a decomposition is only assessed w.r.t. the value of w^+. Nonetheless, from the viewpoint of CSP solving, the most relevant criterion is related to the solving efficiency obtained thanks to the computed tree-decomposition. Of course, this computation must be achieved in reasonable time.

3.2 Comparison Based on CSP Solving Efficiency

In the frame of CSP solving, the quality of a decomposition mostly depends on the practical efficiency we obtain by exploiting it. So we compare LEX-M, LB, min-fill and MCS w.r.t. CSP solving. We consider random CSPs whose graph is one of some CALMA instances (see table 2). Surprisingly, the most interesting decompositions are computed by MCS. Moreover, when we limit the maximal size of separators in the decomposition, the gap between the triangulations significantly decreases. Note that, for efficiency reasons, it is our interest to reduce the value of s, by aggregating the clusters which share a large intersection.

Then, we experiment on partial random structured CSPs (see table 3). For each instance, we randomly produce a random structured CSP [2] and then we remove p% edges. The least promising method, namely MCS, obtains interesting results. Only LB obtains similar or better results w.r.t. the value of w^+ or the CSP solving. However, on the whole, MCS seems the most robust heuristic since it often provides the best approximation of w^+ while offering a limited value of s and solving efficiently CSP. The value of s seems to be an important criterion for the practical solving efficiency. Indeed, with an unlimited value of s, LB or min-fill cannot success in solving some classes (see table 2). Yet, when s is bounded,

LB and min-fill may obtain results close to ones of MCS. Likewise, bounding the value of s significantly improves the results obtained by TM.

4 Discussion and Conclusion

We have considered several approaches for computing a tree-decomposition by triangulating the constraint graph. First, we have observed that, for solving CSPs, the heuristic triangulations in polynomial time might be sufficient to produce a suitable decomposition. Indeed, the optimal or approximate triangulations turn to be too expensive in time w.r.t. the improvement we can expect for the solving. Besides, the criterion w^+ is not relevant enough for CSP solving. Finally, we have noted that limiting the size s of the largest minimal separator allows us to improve the solving runtime, what contradicts the theory (i.e. the time complexity) which requires to minimize w^+ rather than s.

This study must be carried on. The first way consists in improving the computation of tree-decompositions by computing a decomposition which optimizes the solving instead of minimizing the value w^+. Then, another way relies on the strategies to achieve the best depth-first traversal of the associated cluster tree w.r.t. CSP solving, what corresponds to variable heuristics for enumerative methods. Finally, our study must be extended to Valued CSP.

References

1. R. Dechter and J. Pearl. Tree-Clustering for Constraint Networks. *Artificial Intelligence*, 38:353–366, 1989.
2. P. Jégou and C. Terrioux. Hybrid backtracking bounded by tree-decomposition of constraint networks. *Artificial Intelligence*, 146:43–75, 2003.
3. G. Gottlob, M. Hutle, and F. Wotawa. Combining hypertree, bicomp and hinge decomposition. In *Proceedings of ECAI*, pages 161–165, 2002.
4. S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal of Discrete Mathematics*, 8:277–284, 1987.
5. M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press. New-York, 1980.
6. P. Jégou, S. N. Ndiaye, and C. Terrioux. Computing and exploiting tree-decomposition for (Max-)CSP. Technical Report LSIS.RR.2005.005 (www.lsis.org), 2005.
7. E. Amir. Efficient approximation for triangulation of minimum treewidth. In *Proceedings of UAI*, pages 7–15, 2001.
8. D. Rose, R. Tarjan, and G. Lueker. Algorithmic Aspects of Vertex Elimination on Graphs. *SIAM Journal on computing*, 5:266–283, 1976.
9. A. Berry. A Wide-Range Efficient Algorithm for Minimal Triangulation. In *Proceedings of SODA*, january 1999.
10. U. Kjaerulff. Triangulation of Graphs - Algorithms Giving Small Total State Space. Technical report, Judex R.R. Aalborg., Denmark, 1990.
11. R. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13 (3):566–579, 1984.
12. A. M. C. A. Koster, H. L. Bodlaender, and C. P. M. van Hoesel. Treewidth: Computational Experiments. Technical Report 01–38, Berlin, Germany, 2001.