An extension of complexity bounds and dynamic heuristics for tree-decompositions of CSP

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-cezanne.fr

Abstract. This paper deals with methods exploiting tree-decomposition approaches for solving constraint networks. We consider here the practical efficiency of these approaches by defining five classes of variable orders more and more dynamic which guarantee time complexity bounds from $O(exp(w+1))$ to $O(exp(2(w+k)))$, with w the "tree-width" of a CSP and k a constant. Finally, we assess practically their relevance.

1 Introduction

The CSP formalism (Constraint Satisfaction Problem) offers a powerful framework for representing and solving efficiently many problems. An instance of CSP is defined by a tuple (X, D, C) where X is a set of n variables, taking their values in finite domains from D, and being subject to constraints from C. Given an instance, the CSP problem consists in determining if there is an assignment of each variable which satisfies each constraint. This problem 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 *constraint graph* $G = (X, C)$. The usual approach for solving CSP (Backtracking), has an exponential theoretical time complexity in $O(exp(n))$. To improve this bound, structural methods like *Tree-Clustering* [1] were proposed (see [2] for a survey and a theoretical comparison of these methods). They are based on particular features of the instance like the "tree-width" of the constraint graph (denoted w). The *tree-width* w of G is the minimal width over all the tree-decompositions of G [3]. A *tree-decomposition* of G is a pair (E, T) where $T = (I, F)$ is a tree with nodes I and edges F and $E = \{E_i : i \in I\}$ a family of subsets of X, such that each subset (called cluster) E_i is a node of T and verifies: (i) $\cup_{i \in I} E_i = X$, (ii) for each edge $\{x, y\} \in C$, there exists $i \in I$ with $\{x, y\} \subseteq E_i$, and (iii) for all $i, j, k \in I$, if k is in a path from i to j in T, then $E_i \cap E_j \subseteq E_k$. The width of a tree-decomposition (E, T) is equal to $max_{i \in I} |E_i| - 1$. Recent studies (e.g. [4]) integrate as quality parameter for a decomposition, its efficiency for solving the considered CSP. This paper deals with the question of an efficient use of the considered decompositions. We focus on the BTD method (Backtracking on tree-decomposition [5]) which seems to be the most effective method proposed until now within the framework of these structural methods. Indeed, most of works based on this approach only present theoretical results, except [6, 5]. BTD proceeds by an enumerative search guided by a static pre-established partial order induced by a tree-decomposition of the constraint graph. This permits to bound its time complexity by $O(exp(w+1))$, while its space complexity is $O(n.s.d^s)$ with s the size of the largest minimal separators of the graph. Since the efficiency of dynamic variable orders is known, we propose five classes of orders which exploit dynamically the tree-decomposition and guarantee time complexity bounds. Then we define several heuristics for each class.

In section 2, we define the classes and heuristics to compute their orders. Section 3 is devoted to experimental results and conclusions.

2 Classes of orders and heuristics

Even though, the basic version of BTD uses a compatible static variable ordering, we prove here by defining the following classes that it is possible to consider more dynamic orders without loosing the complexity bounds. These orders are in fact provided by the cluster order and the variable ordering inside each cluster. Firstly, we give the definition of a generalized tree-decomposition [7]. The set of directed k-covering tree-decompositions of a tree-decomposition (E, T) of G with E_1 its root cluster and k a non nil positive integer, is defined by the set of tree-decompositions (E', T') of G that verify: (i) $E_1 \subset E'_1$, E'_1 the root cluster of (E', T'), (ii) $\forall E'_i \in E'$, $E'_i \subset E_{i_1} \cup E_{i_2} \cup \ldots \cup E_{i_K}$, with $E_{i_1} \ldots E_{i_K}$ a path in T, and (iii) $|E'_i| \leq w^+ + k$, where $w^+ = max_{E_i \in E}|E_i|$. Given a CSP and a tree-decomposition of its constraint graph, we define:

- **Class 1.** Enumerative static compatible order.
- **Class 2.** Static compatible cluster order and dynamic variable order in the clusters.
- **Class 3.** Dynamic compatible cluster order and dynamic variable order in the clusters.
- **Class 4.** *Class 3* order on a directed k-covering tree-decomposition of the tree-decomposition.
- **Class 5.** *Class 3* order on a set of directed k-covering tree-decompositions of the tree-decomposition.
- **Class ++.** Enumerative dynamic order.

The defined classes form a hierarchy since we have: *Class 1* $\subset$ *Class 2* $\subset$ *Class 3* $\subset$ *Class 4* $\subset$ *Class 5*$\subset$ *Class ++*. The *Class ++* gives a complete freedom, but it does not guarantee time complexity bounds, contrary to the *Class 3*.

Theorem 1 *Let the enumerative order be in the Class 3, the time complexity of BTD is* $O(exp(w+1))$.

The properties of the *Class 3* offer the possibility to choose any cluster to visit next since the variables on the path from the root cluster to that cluster are already assigned. And in each cluster, the variable ordering is totally free. The definitions of the *Class 4* and *Class 5* enforce the order of one assignment to be in the *Class 3*. So we derive natural theorems:

Theorem 2 *Let the enumerative order be in the Class 4 with constant k, the time complexity of BTD is* $O(exp(w^+ + k))$.

Theorem 3 *[7] Let the enumerative order be in the Class 5, the time complexity of BTD is* $O(exp(2(w^+ + k)))$.

We define many heuristics to compute orders in the Classes proposed here and, by lack of place, we only present the more efficient ones:

- *minexp*(A): this heuristic is based on the expected number of partial solutions of clusters [8] and on their size. It chooses as root cluster one which minimizes the ratio between the expected number of solutions and the size of the cluster. It allows to start the exploration with a large cluster having few solutions.
- *size*(B): we have here a local criteria: we choose the cluster of maximum size as root cluster
- $minexp_s(C)$: this heuristic is similar to *minexp* and orders the son clusters according to the increasing value of their ratio.
- $minsep_s(D)$: we order the son clusters according to the increasing size of their separator with their parent.
- $nv(E)$: we visit first the son cluster where appears the next variable in the variable order among the variables of the unvisited son clusters.
- $minexp_{sdyn}(F)$: the next cluster to visit minimizes the ratio between the current expected number
- $nv_{sdyn}(G)$: We visit first the son cluster where appears the next variable in the variable order among the variables of the unvisited son clusters.

Inside a cluster, we use min domain/degree heuristic for choosing the next variable (static version mdd_s for class 1 and dynamic mdd_{dyn} for the other classes).

3 Experimental study and discussion

Applying a structural method on an instance generally assumes that this instance presents some particular topological features. So, our study is performed on random partial structured CSPs described in [7]. All these experimentations are performed on a Linux-based PC with a Pentium IV 3.2GHz and 1GB of memory. For each class, the presented results are the average on instances solved over 50. We limit the runtime to 30 minutes. Above, the solver is stopped and the involved instance is considered as unsolved. In the table, the letter M means that at least one instance cannot be solved because it requires more than 1GB of memory. We use MCS [9] to compute tree-decompositions because it obtains

Table 1. Parameters w^+ and s of the tree-decomposition and runtime (in s) on random partial structured CSPs with *mdd* for class 1 and mdd_{dyn} for classes 2, 3 and 4.

CSP	w^+	s	Class 1		Class 2		Class 3		Class 4			
			B	A	B	A	A	B	B	A	A	B
(n, d, w, t, s, n_c, p)			D	C	D	C	F	G	D	C	F	G
$(150, 25, 15, 215, 5, 15, 10)$	13.0	12.2	9.31	28.12	3.41	2.52	2.45	5.34	2.75	2.17	2.08	2.65
$(150, 25, 15, 237, 5, 15, 20)$	12.5	11.9	9.99	5.27	5.10	2.47	1.99	5.47	2.58	1.76	1.63	2.97
$(150, 25, 15, 257, 5, 15, 30)$	12.1	11.4	13.36	27.82	3.38	5.06	4.97	3.55	1.41	1.05	1.13	1.30
$(150, 25, 15, 285, 5, 15, 40)$	11.5	10.6	3.07	8.77	1.13	0.87	1.27	1.17	1.67	0.39	0.63	1.75
$(250, 20, 20, 107, 5, 20, 10)$	17.8	16.9	54.59	57.75	15.92	12.39	12.14	14.93	10.18	7.75	7.34	10.26
$(250, 20, 20, 117, 5, 20, 20)$	17.2	16.5	55.39	79.80	23.38	14.26	13.25	24.14	10.05	8.81	8.39	10.34
$(250, 20, 20, 129, 5, 20, 30)$	16.8	15.8	26.21	21.14	7.23	5.51	6.19	7.84	33.93	4.61	4.41	34.20
$(250, 20, 20, 146, 5, 20, 40)$	15.9	15.2	44.60	30.17	26.24	3.91	4.51	17.99	11.38	3.17	3.17	10.63
$(250, 25, 15, 211, 5, 25, 10)$	13.0	12.3	28.86	38.75	15.33	11.67	13.37	18.12	5.86	7.71	6.65	6.44
$(250, 25, 15, 230, 5, 25, 20)$	12.8	11.9	20.21	34.47	8.60	7.12	14.84	19.47	4.19	3.94	3.36	6.81
$(250, 25, 15, 253, 5, 25, 30)$	12.3	11.8	11.36	16.91	5.18	11.13	5.14	5.26	2.80	3.71	3.52	3.06
$(250, 25, 15, 280, 5, 25, 40)$	11.8	11.1	7.56	32.74	3.67	16.32	17.49	4.91	4.03	1.40	1.26	3.55
$(250, 20, 20, 99, 10, 25, 10)$	17.9	17.0	M	M	M	M	M	M	66.94	63.15	62.99	66.33
$(500, 20, 15, 123, 5, 50, 10)$	13.0	12.5	12.60	13.63	7.01	8.08	7.31	7.54	5.48	4.50	4.41	5.86
$(500, 20, 15, 136, 5, 50, 20)$	12.9	12.1	47.16	19.22	25.54	23.49	27.01	15.11	4.86	4.92	3.94	5.24

the best results in the study performed in [4] on triangulation algorithms to compute a good tree-decomposition w.r.t. CSP solving. FC and MAC are often unable to solve several instances of each class within 30 minutes.
Table 1 shows the runtime of BTD with several heuristics of Classes 1, 2, 3 and 4. For Class 5, we cannot get good results and then, the results are not presented. Also it presents the width of the computed tree-decompositions and the maximum size of the separators. Clearly, we observe that Class 1 orders obtain poor results. This behaviour is not surprising since static variable orders are well known to be inefficient compared to dynamic ones. A dynamic strategy allows to make good choices by taking in account the modifications of the problem during search. That explains the good results of Classes 2 and 3 orders. The results show as well the crucial importance of the root cluster choice since each heuristic of the Classes 2 and 3 has a dramatic runtime on an average of 4 instances over all instances of all classes because of a bad choice of root cluster. The memory problems marked by M can be solved by using a *Class 4* order with the *sep* heuristic for grouping variables (we merge cluster whose intersection is greater than a value s_{max}). Table 1 gives the runtime of BTD for this class with $s_{max} = 5$. When we analyze the value of the parameter k, we observe that in general, that its value is limited (between 1 to 6). Yet, for the CSPs $(250, 20, 20, 99, 10, 25, 10)$, the value of k is near 40, but this high value allows to solve them.
The heuristics improve very significantly their results obtained for the Classes 2 and 3. The impact of the dynamicity is obvious. *minexp* and *nv* heuristics solve all the instances except one due to a bad root cluster choice, *size* solves all the instances. Except this unsolved instance, *minexp* obtains very promising results. The son cluster ordering has a limited effect because the instances considered have a few son clusters reducing the possible choices and so their impact. The best results are obtained by $minexp + minexp_{sdyn}$, but $size + minsep_s$ obtains often similar results and succeed in solving all instances in the *Class 4*. The calculus of the expected number of solution assumes that the problem

constraints are independent, what is the case for the problems considered here. Thus, $size + minsep$ may outperform $minexp + minexp_{sdyn}$ on real-world problems which have dependent constraints.
These experiments highlight the importance of dynamic orders and make us conclude that the Class 4 gives the best variable orders w.r.t CSP solving with a good value of k. Merging clusters with k less than 5 decreases the maximal size of separator and leads to an important reduction of the runtime.

To summarize, we aim to improve the practical interest of the CSP solving methods based on tree-decompositions. This study takes now on importance for solving hard instances with suitable structural properties since they are seldom solved by enumerative methods like FC or MAC. We defined classes of variable orders which guarantee good theoretical time complexity bounds. A comparison of these classes with relevant heuristics w.r.t. CSP solving, points up the importance of a dynamic variable ordering. Indeed the best results are obtained by *Class 4* orders because they give more freedom to the variable ordering heuristic while their time complexity is $O(exp(w+k))$ where k is a constant to parameterize. Note that for the most dynamic class (the Class 5), we get a time complexity in $O(exp(2(w+k)))$ which should be too large to expect a practical improvement. Then, for *Class 4*, we aim to exploit better the problem features to improve the computing of k. This study will be pursued on the optimization problem.

References

1. R. Dechter and J. Pearl. Tree-Clustering for Constraint Networks. *Artificial Intelligence*, 38:353–366, 1989.
2. G. Gottlob, N. Leone, and F. Scarcello. A Comparison of Structural CSP Decomposition Methods. *Artificial Intelligence*, 124:343–282, 2000.
3. N. Robertson and P.D. Seymour. Graph minors II: Algorithmic aspects of tree-width. *Algorithms*, 7:309–322, 1986.
4. P. Jégou, S. N. Ndiaye, and C. Terrioux. Computing and exploiting tree-decompositions for solving constraint networks. In *Proceedings of CP*, pages 777–781, 2005.
5. P. Jégou and C. Terrioux. Hybrid backtracking bounded by tree-decomposition of constraint networks. *Artificial Intelligence*, 146:43–75, 2003.
6. G. Gottlob, M. Hutle, and F. Wotawa. Combining hypertree, bicomp and hinge decomposition. In *Proceedings of ECAI*, pages 161–165, 2002.
7. P. Jégou, S. N. Ndiaye, and C. Terrioux. Strategies and heuristics for exploiting tree-decompositions of constraint networks. In *Proceedings of WIGSK*, 2006.
8. B. Smith. The Phase Transition and the Mushy Region in Constraint Satisfaction Problems. In *Proceedings of ECAI*, pages 100–104, 1994.
9. 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.