Tabu Search for Generalized Hypertree Decompositions

Nysret Musliu*

*Vienna University of Technology
Favoritenstrasse 9, 1040, Vienna, Austria
musliu@dbai.tuwien.ac.at

1 Introduction

Many important real world problems can be formulated as Constraint Satisfaction Problems (CSPs). A CSP consists of a set of variables each with a domain of possible values, and a set of constraints on the allowed values for specified subsets of variables. A solution to CSP is the assignment of values to variables, such that no constraint is violated. CSPs include many NP-complete problems and are in general hard to solve. However, some classes of CSPs can be solved efficiently if they have bounded treewidth (tw) or generalized hypertree width (ghw). The process of solving problems with bounded tw/ghw includes two phases. In the first phase the tree or generalized hypertree decomposition with small upper bound for tw/htw width is generated. Second phase includes solving a problem (based on the decomposition) with a particular algorithm which runs in polynomial time on the width and the size of the problem. Given that the classes of CSPs of bounded tw/ghw are solvable in polynomial time, these two concepts are very important for efficient solving of intractable problems. In this paper we consider the generation of generalized hypertree decompositions of small width.

The concept of hypertree decomposition has been introduced by Gottlob et al [2]. To generate generalized hypertree decomposition for the given CSP the corresponding constraint hypergraph should be first constructed. The hypergraph $H = (V(H), E(H))$ consists of a nonempty set $V(H)$ of *vertices*, and a set $E(H)$ of subsets of $V(H)$. The CSP hypergraph is constructed such that $V(H)$ represents the variables of the problem, whereas $E(H)$ represent the scope of constraint of CSP problem.

A *tree decomposition* of a hypergraph H is a tree, where each node of a tree is associated to a set of vertices (variables) of hypergraph H. The vertices of each hyperedge of $E(H)$ are contained in the set of vertices of some node of the tree, and the set of nodes in the tree which are associated to each vertex form a connected subtree. The width of tree decomposition is the maximum number of vertices associated to the same node of the tree minus one. The treewidth of a hypergraph H, denoted by $tw(H)$, is the minimum width over all possible tree decompositions of H.

A *generalized hypertree decomposition (GHD) of H* is a tree decomposition of H with the following extension. GHD associates additionally to each node of the decomposition tree the set of hyperedges of H. The set of vertices associated to each node of the tree must be covered by the set of hyperedges associated to that node. The *width* of a generalized hypertree decomposition is the maximum number of hyperedges associated to a same node of the decomposition, and the *generalized hypertree-width* $ghw(H)$ of H is the minimum of the widths of all generalized hypertree decompositions of H.

In Figure 1 is given a hypergraph H, a possible tree decomposition of H, and a generalized hypertree decomposition of H. The given constraint hypergraph represents a constraint satisfaction problem with 14 variables and 8 constraints. The presented tree decomposition has width 5, whereas the width of generalized hypertree decomposition is 2.

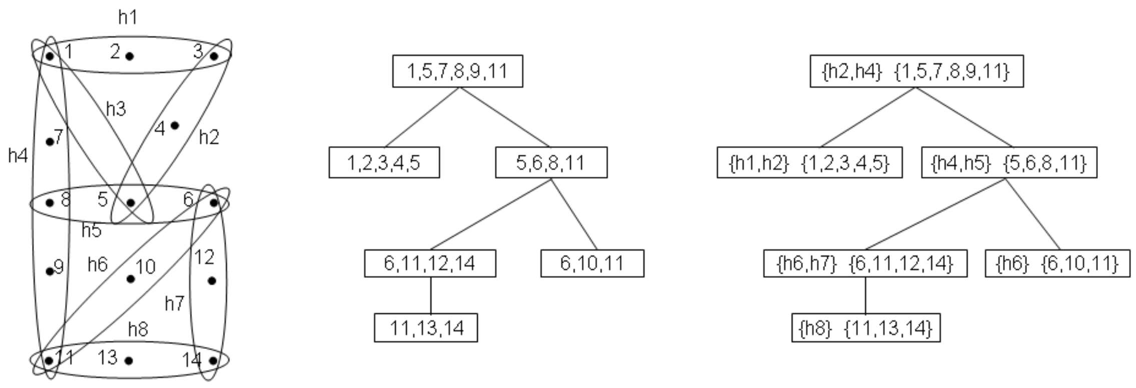

Figure 1: A hypergraph H (left), a tree decomposition of H (middle), and a generalized hypertree decomposition of H (right)

The problem of determining whether for the given k and input hypergraph H the ghw is smaller then k is NP-complete problem. Thus, different heuristic methods have been used for generation of generalized hypertree decompositions. Very good results in the literature have been obtained with heuristic methods that generate generalized hypertree decompositions based on searching for good tree decompositions [1]. These techniques search for a good perfect elimination ordering of vertices of hypergraph, and first produce good tree decomposition. The GHD is constructed then from the generated tree decomposition.

2 Tabu Search for Generalized Hypertree Decomposition

Recently, local search techniques have been shown to give state of art solutions for tree decompositions. Motivated by good results for tree decompositions obtained by Iterated Local Search [3] we extend this method and apply it for generation of generalized hypertree decompositions. First, we propose a tabu search approach as a construction phase in the iterated local search algorithm. The algorithm searches among the possible elimination orderings of vertices of hypergraph to find a elimination ordering which produces a small width. The neighbourhood is constructed based on the idea of moving only those vertices in the ordering, which cause the largest clique during the elimination process. We investigate the exploration of two types of neighbourhood. In the first case for each iteration a vertex in the elimination ordering which causes the largest clique (ties are broken randomly) is swapped with another vertex located in the randomly chosen position. In the second case, first a vertex in the elimination ordering which causes the largest clique is selected. Then, the neighbourhood of the solution is generated by swapping the selected vertex with its neighbours, i.e. all solutions are generated by swapping the selected vertex with its neighbours. To avoid cycles the tabu mechanism is applied. The swapped vertices are made tabu for a determined number of iterations. The iterated local search algorithm further includes the perturbation mechanism and the mechanism for accepting of solution for the next iteration. We experimented with different perturbation mechanisms and different techniques for acceptance of a solution.

We used two types of fitness function in searching for good elimination ordering of vertices. In the first variant, the fitness function represents the width of a tree decomposition that can be constructed from the particular elimination ordering of vertices. In this case, after the tree decomposition is generated, to obtain the generalized hypertree decomposition the set covering is applied in each node of the tree to find the minimum number of hyperedges which cover the vertices associated to that node. In the second variant, we experimented with the fitness function that directly calculates the width of generalized hypertree decomposition. In this case, to find the width of GHD the set covering is applied in cliques created during the elimination process of vertices. This method optimizes directly the width of generalized hypertree decompositions, but runs much slower compared to the first variant.

The algorithm proposed in this paper is tested on benchmark examples from industry and the literature. The preliminary results show that the proposed method can be successfully applied for problem instances of different size and it gives promising results for generation of generalized hypertree decompositions. For example, when applying the proposed algorithm to problems adder_15 (76 vertices / 106 constraints), adder_99 (496 / 694), and NASA (680 / 579), it generates generalized hypertree decompositions with width $3, 4$, and 25, respectively. The best upper bound in the literature for both adder_15, and adder_99 is 2. Considering NASA problem the best known upper bound for *ghw* is 19. This upper bound was found by an implementation of genetic algorithm in 112291 seconds, while maximal running time of our algorithm was 1000 seconds. We are considering further improvement of the proposed method, by additional tuning of iterated local search parameters, and by improving of local search procedure that includes tabu mechanism.

Acknowledgments: This paper was supported by the Austrian Science Fund (FWF) project: *Nr. P17222-N04, Complementary Approaches to Constraint Satisfaction*

References

[1] Artan Dermaku, Tobias Ganzow, Georg Gottlob, Ben McMahan, Nysret Musliu, and Marko Samer. Heuristic methods for hypertree decompositions. Technical Report DBAI-TR-2005-53, Technische Universitt Wien, 2005.

[2] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. Journal of Computer and System Sciences, *64(3):579–627, 2002.*

[3] Nysret Musliu. Generation of tree decompositions by iterated local search. In EvoCOP 2007 - Seventh European Conference on Evolutionary Computation in Combinatorial Optimisation, LNCS, *volume 4446, pages 130–141. Springer, 2007.*