Treewidth: Computational Experiments

Arie M.C.A. Koster [a], Hans L. Bodlaender [b,1],
Stan P.M. van Hoesel [c]

[a] *Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustraße 7, D-14195 Berlin, Germany. E-Mail: koster@zib.de.*

[b] *Institute of Information and Computing Sciences, Universiteit Utrecht, P.O.Box 80.089, 3508 TB Utrecht, The Netherlands.*

[c] *Department of Quantitative Economics, Universiteit Maastricht, P.O.Box 616, 6200 MD Maastricht, The Netherlands.*

In their fundamental research on graph minors, Robertson and Seymour defined the graph parameter treewidth [5] together with the associated graph structure tree decomposition. This notion (and the related notions pathwidth and branchwidth) proved to be important in algorithm theory. Many ***NP***-complete graph problems can be solved in polynomial time for graphs of bounded treewidth. Among these problems are many well-known combinatorial optimisation problems such as the maximum independent set problem, the Hamiltonian circuit problem, and the Steiner tree problem. Moreover, tree decompositions are used to solve the probabilistic inference problem for probabilistic (or Bayesian) networks (which is a central result in the theory and practice of a lively research area from expert systems) [4]. Also, several ***PSPACE***-complete problems can be solved in linear time for inputs restricted to graphs of bounded treewidth (cf. [1]).

Many of these results first seemed to be of theoretical interest only. In recent years, however, the results were applied successfully in a number of computational studies [2–4,8]. These studies show that algorithms based on a tree/path/branch decomposition of the graph can be an alternative to integer programming and other techniques to solve hard (combinatorial) optimisation problems.

The procedure to solve an optimisation problem with bounded treewidth, for instance, involves two steps: (i) computation of a (good) tree decomposition and (ii) application of an algorithm that solves instances of bounded treewidth in polynomial time. Whereas the second step is application-dependent, the calculation of a good tree decomposition of a graph can be done independently

[1] This research was partially supported by EC contract IST-1999-14186: Project ALCOM-FT (Algorithms and Complexity - Future Technologies).

of the application. The need for a tree decomposition with optimal, or second best, good width becomes clear by the running time of the algorithm in the second step, which typically is exponentially in the width.

The treewidth of a graph, however, is ***NP***-hard to compute in general. Much research has been carried out to determine the treewidth (or related notions) for classes of graphs. For fixed k, linear time algorithms to determine whether a graph has treewidth k have been developed. Their use in practice is very limited since the running time contains a large constant with k in the exponent. Approximation with an $O(\log |V|)$ approximation ratio can be done in polynomial time; finding a polynomial time constant approximation ratio algorithm remains open. We refer to [1] for a survey.

Hence, heuristics without theoretical quality guarantee remain as a practical alternative. In this paper, we compare four such heuristics. The quality of the upper bounds is benchmarked with two lower bounds. Due to space limitations, preliminaries and algorithmic details are omitted in this extended abstract, and only few computational results are presented.

Heuristics

Graph Triangulation Heuristics. Given a graph $G = (V, E)$, there exists a triangulation $G_t = (V, E \cup E_t)$ with maximum clique of size $k + 1$ if and only if the treewidth of G is k. As the maximum clique size of a triangulated graph can be computed in linear time, triangulations of graphs are particularly suited to determine an upper bound on the treewidth.

Three algorithms have been designed for testing whether a graph is triangulated (or 'chordal') that also yield triangulations for inputs that are not triangulated. None of these algorithms guarantees a minimum of the maximum clique size. In [6], two algorithms based on the *lexicographic breadth first search* principle are given: LEX_P (using $O(|V| + |E|)$ time) and LEX_M (using $O(|V||E|)$ time). In [7], the *maximum cardinality search* (MCS) algorithm is given, which also uses $O(|V| + |E|)$ time. Only LEX_M guarantees that the triangulation is *minimal* in the following sense: it does not contain as subgraph another triangulation with fewer edges.

Minimum Separating Vertex Sets Heuristic. The third heuristic, developed in the context of solving hard frequency assignment problems [3], uses another characterisation of treewidth based on separating vertex sets. Every tree decomposition can be transformed to a tree decomposition in which the vertex-set associated to an internal node separates the graph into at least two components, i.e., the with the node associated vertices form a separating vertex set.

A minimum separating vertex set can be found efficiently by Menger's theorem. This result forms the basic for the Minimum Separating Vertex Sets

(MSVS) heuristic. The heuristic starts with a (trivial) tree decomposition. To reduce the width, we recursively replace nodes of the tree decomposition with associated vertex set of maximum cardinality by new nodes of smaller cardinality. The new nodes are obtained by looking for a minimal separating vertex set in a graph defined on the vertex set in such a way that after replacement of the node, all conditions for a tree decomposition are still satisfied. The new nodes are defined by the separating vertex set and the connected components of the graph without this set. The refinement steps can be regarded as 'optimal' in case the separating vertex set induces a clique in the original graph. For further details, we refer to [3].

Lower bounds

Maximum Clique Bound (MC). By the definition of treewidth, the maximum clique size of the graph minus one is a lower bound on the treewidth. Although computing the maximum clique in a graph is ***NP***-hard as well, it can be computed within reasonable time for many instances.

Maximum Minimum Degree Bound (MMD). A trivial lower bound on the treewidth is given by the minimum degree of the graph. The minimum degree of a subgraph provides a lower bound as well, since taking minors of a graph can not increase the treewidth. Hence, the maximum of the minimum degree over all subgraphs bounds the treewidth from below. Since the maximum clique is a subgraph, this bound is at least as good as the maximum clique bound. However, the max-min degree bound can be computed in polynomial time. Let $v \in V$ be a vertex in G of minimum degree. The minimum degree of all subgraphs G' that contain v is at most the minimum degree of G. Thus, the max-min degree of G, $MMD(G)$, is given by $\max\{\delta(v), MMD(G[V-v])\}$ for all $v \in V$ of minimum degree. This directly provides an algorithm to compute the max-min degree: remove subsequently a vertex of minimum degree of the graph and keep track of the maximum of these minimum degrees.

Computational Results

In Table 1, we present preliminary computational results for the (preprocessed) instances of the CALMA project on frequency assignment [3] (preprocessed for frequency assignment). Results for graphs from other applications will be presented in the talk and in the final paper. Our experiments show that in many cases, the MSVS heuristic performs better than the triangulation heuristics; of these, LEX_M and MCS do often better than LEX_P; on some instances LEX_M outperforms MCS, on some instances MCS does better than LEX_M. The three triangulation heuristics allow for a choice of the vertex to begin the algorithm with: tests show that often a big difference exists between the best, the worst, and the average width over all choices of a starting vertex. The presented computation times therefore represent $|V|$ runs of the algorithms,

one run for every vertex to begin with. The computation times for the MSVS heuristic can be substantially reduced by the use of graph decomposition techniques.

instance	$\|V\|$	$\|E\|$	lower bounds		upper bounds				CPU time (seconds on a Pentium III, 800 Mhz)					
			MC	MMD	MCS	LEX_P	LEX_M	MSVS	MC	MMD	MCS	LEX_P	LEX_M	MSVS
CELAR06	100	350	10	10	11	11	11	11	0.13	0.00	0.07	0.08	0.53	9.30
CELAR06pp	82	327	10	10	11	11	11	11	0.10	0.00	0.05	0.06	0.38	5.67
CELAR07	200	817	10	11	18	19	19	17	0.45	0.01	0.45	0.56	4.20	101.62
CELAR07pp	162	764	10	11	18	19	19	17	0.40	0.01	0.34	0.43	2.92	58.65
CELAR08	458	1655	10	11	21	21	21	18	1.48	0.04	2.52	3.51	34.30	603.19
CELAR08pp	365	1539	10	11	20	21	21	18	1.28	0.03	1.91	3.04	20.98	412.07
CELAR09	340	1130	10	11	19	21	21	18	0.70	0.02	1.37	2.05	21.95	580.30
CELAR09pp	67	165	7	7	7	7	7	7	0.01	0.00	0.02	0.02	0.07	0.03
GRAPH05	100	416	8	8	29	29	29	27	0.06	0.00	0.62	0.79	1.60	34.98
GRAPH06	200	843	8	8	54	60	60	56	0.19	0.01	15.23	19.39	25.93	323.19
GRAPH06pp	119	348	5	5	19	21	21	18	0.02	0.00	0.43	0.48	1.72	22.81
GRAPH07	200	843	8	8	54	60	60	56	0.18	0.01	15.25	19.51	25.85	323.37
GRAPH11	340	1425	7	7	105	105	105	102	0.32	0.01	239.64	255.07	286.90	2290.80
GRAPH12	340	1255	5	5	97	99	99	93	0.17	0.02	203.48	223.30	239.42	2124.58
GRAPH12pp	61	1255	4	4	5	5	5	4	0.00	0.00	0.02	0.02	0.11	2.04
GRAPH13	458	1877	6	6	136	147	147	133	0.48	0.04	931.51	1054.18	1130.84	7507.91
GRAPH13pp	456	1874	6	6	133	147	147	133	0.48	0.04	924.34	1098.28	1117.53	7602.44

Table 1
Computational results for CALMA frequency assignment instances.

References

[1] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11(1-2):1–21, 1993.

[2] W. Cook and P. D. Seymour. An algorithm for the ring-router problem. Technical report, Bellcore, 1994.

[3] A. M. C. A. Koster. *Frequency Assignment - Models and Algorithms*. PhD thesis, Maastricht University, 1999. Available at http://www.zib.de/koster/.

[4] S. J. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *The Journal of the Royal Statistical Society. Series B (Methodological)*, 50:157–224, 1988.

[5] N. Robertson and P. D. Seymour. Graph minors. II. algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[6] D. J. Rose, R. E. Tarjan, and G. S. Leuker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5:266–283, 1976.

[7] R. E. 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.

[8] B. Verweij. *Selected Applications of Integer Programming: A Computational Study*. PhD thesis, Universiteit Utrecht, Utrecht, The Netherlands, 2000.