Contraction and Treewidth Lower Bounds*

Hans L. Bodlaender[1], Arie M.C.A. Koster[2], and Thomas Wolle[1]

[1] Institute of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
{hansb,thomasw}@cs.uu.nl
[2] Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7, D-14194 Berlin, Germany
koster@zib.de

Abstract. Edge contraction is shown to be a useful mechanism to improve lower bound heuristics for treewidth. A successful lower bound for treewidth is the degeneracy: the maximum over all subgraphs of the minimum degree. The degeneracy is polynomial time computable. We introduce the notion of contraction degeneracy: the maximum over all graphs that can be obtained by contracting edges of the minimum degree. We show that the problem to compute the contraction degeneracy is NP-hard, but for fixed k, it is polynomial time decidable if a given graph G has contraction degeneracy at least k. Heuristics for computing the contraction degeneracy are proposed and experimentally evaluated. It is shown that these can lead to considerable improvements to the lower bound for treewidth. A similar study is made for the combination of contraction with Lucena's lower bound based on Maximum Cardinality Search [12].

1 Introduction

It is about two decades ago that the notion of treewidth and the equivalent notion of partial k-tree were introduced. Nowadays, these play an important role in many theoretic studies in graph theory and algorithms, but also their use for practical applications is growing, see, e.g.: [10,11]. A first step when solving problems on graphs of bounded treewidth is to compute a tree decomposition of (close to) optimal width, on which often a dynamic programming approach is applied. Such a dynamic programming algorithm typically has a running time that is exponential in the treewidth of the graph. Since determining the treewidth of a graph is an NP-hard problem, it is rather unlikely to find efficient algorithms for computing the treewidth. Therefore, we are interested in lower and upper bounds for the treewidth of a graph.

This paper focuses on lower bounds for the treewidth. Good lower bounds can serve to speed up branch and bound methods, inform us about the quality

* This work was partially supported by the DFG research group "Algorithms, Structure, Randomness" (Grant number GR 883/9-3, GR 883/9-4), and partially by the Netherlands Organisation for Scientific Research NWO (project *Treewidth and Combinatorial Optimisation*).

S. Albers and T. Radzik (Eds.): ESA 2004, LNCS 3221, pp. 628–639, 2004.

of upper bound heuristics, and in some cases, tell us that we should not use tree decompositions to solve a problem on a certain instance. A large lower bound on the treewidth of a graph implies that we should not hope for efficient dynamic programming algorithms that use tree decompositions for this particular instance.

In this paper, we investigate the combination of contracting edges with two existing lower bounds for treewidth, namely the degeneracy (or MMD [9]) and the MCSLB (the Maximum Cardinality Search Lower Bound), introduced by Lucena [12]. The definitions of these lower bounds can be found in Section 2. Contracting an edge is replacing its two endpoints by a single vertex, which is adjacent to all vertices at least one of the two endpoints was adjacent to. Combining the notion of contraction with degeneracy gives the new notion of *contraction degeneracy* of a graph G: the maximum over all graphs that can be obtained by contracting edges of G of the minimum degree. It provides us with a new lower bound for the treewidth of graphs. While unfortunately, computing the contraction degeneracy of a graph is NP-complete (as is shown in Section 3.2), the fixed parameter cases are polynomial time solvable (see Section 3.3), and there are simple heuristics that provide us with good bounds on several instances taken from real life applications (see Section 5).

We experimented with three different heuristics for the contraction degeneracy, based on repeatedly contracting a vertex of minimum degree with (a) a neighbor of smallest degree, (b) a neighbor of largest degree, or (c) a neighbor that has as few as possible common neighbors. We see that the last one outperforms the other two, which can be explained by noting that such a contraction 'destroys' only few edges in the graph. Independently, the heuristic with contraction with neighbors of minimum degree has been proposed as a heuristic for lower bounds for treewidth by Gogate and Dechter [7] in their work on branch and bound algorithms for treewidth.

The lower bound provided by MCSLB is never smaller than the degeneracy, but can be larger [3]. The optimisation problem to find the largest possible bound of MCSLB for a graph obtained from G by contracting edges is also NP-hard, and polynomial time solvable for the fixed parameter case (Section 4). We also studied some heuristics for this bound. In our experiments, we have seen that typically, the bound by MCSLB is equal to the degeneracy or slightly larger. In both cases, often a large increase in the lower bound is obtained when we combine the method with contraction. See Section 5 for results of our computational experiments. They show that contraction is a very viable idea for obtaining treewidth lower bounds.

A different lower bound for treewidth was provided by Ramachandramurthi [13,14]. While this lower bound appears to generally give small lower bound values, it can also be combined with contraction. Some first tests indicate that this gives in some cases a small improvement to the lower bound obtained with contraction degeneracy.

An interesting and useful method for getting treewidth lower bounds was given by Clautiaux et al. [4]. The procedure from [4] uses another treewidth

lower bound as subroutine. The description in [4] uses the degeneracy, but one can also use another lower bound instead. Our experiments (not given in this extended abstract for space reasons) showed that the contraction degeneracy heuristics generally outperform the method of [4] with degeneracy. Very recently, we combined the method of [4] with the contraction degeneracy heuristics, using a 'graph improvement step as in [4]' after each contraction of an edge. While this increases the time of the algorithm significantly, the algorithm still takes polynomial time, and in many cases considerable improvements to the lower bound are obtained. More details of this new heuristic will be given in the full version of this paper.

2 Preliminaries

Throughout the paper $G = (V, E)$ denotes a simple undirected graph. Most of our terminology is standard graph theory/algorithm terminology. As usual, the degree in G of vertex v is $d_G(v)$ or simply $d(v)$. $N(S)$ for $S \subseteq V$ denotes the open neighbourhood of S, i.e. $N(S) = \bigcup_{s \in S} N(s) \setminus S$. We define: $\delta(G) := \min_{v \in V} d(v)$.

Subgraphs and Minors. After deleting vertices of a graph and their incident edges, we get an *induced subgraph*. A *subgraph* is obtained, if we additionally allow deletion of edges. If we furthermore allow edge-contractions, we get a *minor*. The treewidth of a minor of G is at most the treewidth of G (see e.g. [2]).

Edge-Contraction. Contracting edge $e = \{u, v\}$ in the graph $G = (V, E)$, denoted as G/e, is the operation that introduces a new vertex a_e and new edges such that a_e is adjacent to all the neighbours of v and u and delete vertices u and v and all edges incident to u or v: $G/e := (V', E')$, where $V' = \{a_e\} \cup V \setminus \{u, v\}$ and $E' = \{\, \{a_e, x\} \mid x \in N(\{u, v\})\} \cup E \setminus \{e' \in E \mid e' \cap e \neq \emptyset\}$. A *contraction-set* is a cycle free set $E' \subseteq E(G)$ of edges. Note that after each single edge-contraction the names of the vertices are updated in the graph. Hence, for two adjacent edges $e = \{u, v\}$ and $f = \{v, w\}$, edge f will be different after contracting edge e, namely in G/e we have $f = \{a_e, w\}$. Thus, f represents the same edge in G and in G/e. For a contraction-set $E' = \{e_1, e_2, \ldots, e_p\}$, we define $G/E' := G/e_1/e_2/\ldots/e_p$. Furthermore, note that the order of edge-contractions to obtain G/E' is not relevant. A *contraction* H of G is a graph such that there exists a contraction-set E' with: $H = G/E'$.

Treewidth. A *tree decomposition* of $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, with $\{X_i \mid i \in I\}$ a family of subsets of V and T a tree, such that $\bigcup_{i \in I} X_i = V$, for all $\{v, w\} \in E$, there is an $i \in I$ with $v, w \in X_i$, and for all $i_0, i_1, i_2 \in I$: if i_1 is on the path from i_0 to i_2 in T, then $X_{i_0} \cap X_{i_2} \subseteq X_{i_1}$. The *width* of tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i| - 1$. The *treewidth* $tw(G)$ of G is the minimum width among all tree decompositions of G.

Degeneracy/MMD. We also use the term MMD (Maximum Minimum Degree) for the degeneracy. The degeneracy δD of a graph G is defined to be: $\delta D(G) := \max_{G'}\{\delta(G') \mid G' \text{ is a subgraph of } G\}$. The minimum degree of a graph is a lower bound for its treewidth and the treewidth of G cannot increase by taking subgraphs. The treewidth of G is at least its degeneracy. (See also [9].)

Maximum Cardinality Search. MCS is a method to number the vertices of a graph. It was first introduced by Tarjan and Yannakakis for the recognition of chordal graphs [18]. We start by giving some vertex number 1. In step $i = 2, \ldots, n$, we choose an unnumbered vertex v that has the largest number of already numbered neighbours, breaking ties as we wish. Then we associate number i to vertex v.

An *MSC-ordering* ψ can be defined by mapping each vertex to its number: $\psi(v) :=$ number of v. For a fixed MCS-ordering, let $v_i := \psi^{-1}(i)$. The visited degree $vd_\psi(v_i)$ of v_i is defined as follows: $vd_\psi(v) := d_{G[v_1,\ldots,v_i]}(v_i)$. The visited degree $MCSLB_\psi$ of an MCS-ordering ψ is defined as follows: $MCSLB_\psi := \max_{i \in V(G)} vd_\psi(v_i)$

In [12], Lucena has shown that for every graph G and MCS-ordering ψ of G, $MCSLB_\psi \leq tw(G)$. Thus, an MCS numbering gives a lower bound for the treewidth of a graph.

3 Contraction Degeneracy

We define the new parameter of contraction degeneracy and the related computational problem. Then we will show the NP-completeness of the problem just defined, and consider the complexity of the fixed parameter cases.

3.1 Definition of the Problem

Definition 1. *The* contraction degeneracy δC *of a graph* G *is defined as follows:*

$$\delta C(G) := \max_{G'}\{\delta(G') \mid G' \text{ is a minor of } G\}$$

It is also possible to define δC as the maximum over all contractions of the minimum degree of the contraction. Both definitions are equivalent, since deleting edges or vertices does not help to obtain larger degrees. The corresponding decision problem is formulated as usual:

Problem: Contraction Degeneracy
Instance: Graph $G = (V, E)$ and integer $k \geq 0$.
Question: Is the contraction degeneracy of G at least k, i.e. is $\delta C(G) \geq k$, i.e. is there a contraction-set $E' \subseteq E$, such that $\delta(G/E') \geq k$?

Lemma 1. *For a graph* G*, it holds:* $\delta C(G) \leq tw(G)$

Proof. Note that for any minor G' of G, we have: $tw(G') \leq tw(G)$. Furthermore, for any graph G': $\delta(G') \leq tw(G')$. The lemma follows now directly. □

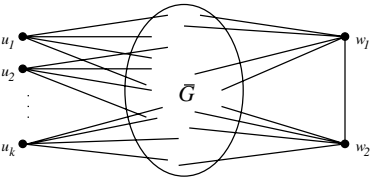

Fig. 1. Intermediate graph G' constructed for the transformation.

3.2 *NP*-Completeness

Theorem 1. *The* Contraction Degeneracy *problem is NP-complete for bipartite graphs.*

Proof. Because of space constraints, we will not give the full proof here. Instead, we only show the construction for the transformation. Clearly, the problem is in NP. The hardness proof is a transformation from the Vertex Cover problem. Let be given a Vertex Cover instance (G, k), with $G = (V, E)$.

Construction: Starting from the complement of G, we add vertices and edges to construct a new graph G', which is an intermediate step in the transformation. $G' = (V', E')$ is formally defined as follows, see Figure 1:

$$\begin{aligned} V' &= V \cup \{w_1, w_2\} \cup \{u_1, \ldots, u_k\} \\ E' &= (V \times V \setminus E) \cup \{\ \{w_1, w_2\}\ \} \cup \{\ \{w_i, v\} \mid i \in \{1, 2\} \wedge v \in V\ \} \cup \\ &\quad \{\ \{u_i, v\} \mid i \in \{1, \ldots, k\} \wedge v \in V\ \} \end{aligned}$$

The final graph $G^* = (V^*, E^*)$ in our construction is obtained by subdividing any edge in G', i.e. replacing each edge in G' by a path with two edges.

$$V^* = V' \cup \{\ v_e \mid e \in E'\ \} \text{ and } E^* = \{\ \{u, v_e\}, \{v_e, w\} \mid e = \{u, w\} \in E'\ \}$$

Let $n := |V|$, the constructed instance of the Contraction Degeneracy problem is $(G^*, n + 1)$.

Now, it is easy to see that G^* is a bipartite graph. It is also possible to prove that there is a vertex cover for G of size at most k if, and only if $\delta C(G^*) \geq n+1$. □

3.3 Fixed Parameter Cases

Now, we consider the fixed parameter case of the Contraction Degeneracy problem. I.e., for a fixed integer k, we consider the problem to decide for a given graph G if G has a contraction with minimum degree k. Graph minor theory gives a fast answer to this problem. Those unfamiliar with this theory are referred to [6].

Theorem 2. *The* Contraction Degeneracy *problem can be solved in linear time when k is a fixed integer with $k \leq 5$, and can be solved in $O(n^3)$ time when k is a fixed integer with $k \geq 6$.*

Proof. Let k be a fixed integer. Consider the class of graphs $\mathcal{G}_k = \{G \mid G$ has contraction degeneracy at most $k-1\}$. $\mathcal{G}_k$ is closed under taking minors: if H is a minor of G and H has contraction degeneracy at least k, then G has also contraction degeneracy at least k. As every class of graphs that is closed under minors has an $O(n^3)$ algorithm to test membership by Robertson-Seymour graph minor theory (see [6]), the theorem for the case that $k \geq 6$ follows.

Suppose now that $k \leq 5$. There exists a planar graph G_k with minimum degree k. Hence, $G_k \notin \mathcal{G}_k$. A class of graphs that is closed under taking minors and does not contain all planar graphs has a linear time membership test (see [6]), which shows the result for the case that $k \leq 5$. □

It can be noted that the cases that $k = 1$ and $k = 2$ are very simple: a graph has contraction degeneracy at least 1, if and only if it has at least one edge, and it has contraction degeneracy at least 2, if and only if it is not a forest. The result is non-constructive when $k \geq 6$; when $k \leq 5$, the result can be made constructive by observing that the property that G has contraction degeneracy k can be formulated in monadic second order logic for fixed k. Thus, we can solve the problem as follows: the result of [17] applied to G_k gives an explicit upper bound c_k on the treewidth of graphs in $\mathcal{G}_k$. Test if G has treewidth at most c_k, and if so, find a tree decomposition with width at most c_k with the algorithm of [1]. If G has treewidth at most c_k, use the tree decomposition to test if the MSOL formula holds for G [5]; if not, we directly know that G has contraction degeneracy at least k. The constant factors hidden in the O-notation are very large, thus these results have only theoretical importance. We summarise the different cases in the following table.

Table 1. Complexity of Contraction Degeneracy

k	Time	Reference
1	$O(n)$	Trivial
2	$O(n)$	G is not a forest
3, 4, 5	$O(n)$	[1,5,17],MSOL
fixed $k \geq 6$	$O(n^3)$	[16,15]
variable k	NP-complete	Theorem 1

4 Maximum Cardinality Search with Contraction

From a maximum cardinality search ordering, we can derive a parameter that is a lower bound for the treewidth of a graph (see Section 2). We define four problems related to this parameter. Each of these is NP-complete or NP-hard, respectively. We consider the following problem, and variants.

Problem: MCSLB With Contraction
Instance: Graph $G = (V, E)$, integer k.
Question: Does G have a contraction H, and H an MCS-ordering ψ with the visited degree of ψ at least k?

In the variant MCSLB With Minors we require instead that H is a minor of G. In the variants MinMCSLB With Contraction and MinMCSLB With Minors we ask that *every* MCS-ordering ψ of H has visited degree at least k.

Theorem 3. MCSLB With Contraction *and* MCSLB With Minors *are NP-complete.* MinMCSLB With Contraction *and* MinMCSLB With Minors *are NP-hard.*

The hardness proofs are transformations from Vertex Cover, with a somewhat more involved variant of the proof of Theorem 1. Membership in NP is trivial for MCSLB With Contraction and MCSLB With Minors, but unknown for the other two problems.

The fixed parameter case of MCSLB With Minors can be solved in linear time with help of graph minor theory. Observing that the set of graphs $\{G \mid G$ does not have a minor H, such that H has an MCS-ordering ψ with the visited degree of ψ at least $k\}$ is closed under taking of minors, and does not include all planar graphs (see [3]), gives us by the Graph Minor theorem of Robertson and Seymour and the results in [1,17] the following result.

Theorem 4. MCSLB With Minors *and* MinMCSLB With Minors *are linear time solvable for fixed k.*

The fixed parameter cases of MinMCSLB With Minors give an $O(n^3)$ algorithm (linear when $k \leq 5$), similar as for Theorem 2. Again, note that these results are only of theoretical interest, due to the high constant factors hidden in the O-notation, and the non-constructiveness of the proof.

5 Experimental Results

In this section, we report on the results of computational experiments we have carried out. We tested our algorithms on a number of graphs, obtained from two application areas where treewidth plays a role in algorithms that solve the problems at hand. The first set of instances are probabilistic networks from existing decision support systems from fields like medicine and agriculture. The second set of instances are from frequency assignment problems from the EUCLID CALMA project (see e.g. [8]). We have also used this set of instances in earlier experiments. We excluded those networks for which the MMD heuristic already gives the exact treewidth. Some of the graphs can be obtained from [19]. All algorithms have been written in C++, and the computations have been carried out on a Linux operated PC with a 3.0 GHz Intel Pentium 4 processor.

Table 2. Graph sizes, upper bounds, and MMD/MMD+ lower bounds

instance	size		UB	MMD		MMD+					
						min-d.		max-d.		least-c.	
	$\|V\|$	$\|E\|$		LB	CPU	LB	CPU	LB	CPU	LB	CPU
barley	48	126	7	5	0.00	6	0.00	5	0.00	6	0.00
diabetes	413	819	4	3	0.00	4	0.01	4	0.00	4	0.00
link	724	1738	13	4	0.00	8	0.02	5	0.01	11	0.03
mildew	35	80	4	3	0.00	4	0.00	3	0.00	4	0.00
munin1	189	366	11	4	0.00	8	0.01	5	0.00	10	0.00
munin2	1003	1662	7	3	0.01	6	0.01	4	0.01	6	0.02
munin3	1044	1745	7	3	0.01	7	0.01	4	0.02	7	0.02
munin4	1041	1843	8	4	0.01	7	0.01	5	0.01	7	0.02
oesoca+	67	208	11	9	0.00	9	0.00	9	0.00	9	0.00
oow-trad	33	72	6	3	0.00	4	0.00	4	0.00	5	0.00
oow-bas	27	54	4	3	0.00	4	0.00	3	0.00	4	0.00
oow-solo	40	87	6	3	0.00	4	0.00	4	0.00	5	0.00
pathfinder	109	211	6	5	0.00	6	0.00	5	0.00	6	0.01
pignet2	3032	7264	135	4	0.01	29	0.11	10	0.07	38	0.20
pigs	441	806	10	3	0.00	6	0.01	4	0.00	7	0.01
ship-ship	50	114	8	4	0.00	6	0.00	4	0.00	6	0.00
water	32	123	10	6	0.00	7	0.00	7	0.00	8	0.00
wilson	21	27	3	2	0.00	3	0.00	3	0.00	3	0.00
celar01	458	1449	17	8	0.00	12	0.01	9	0.00	14	0.02
celar02	100	311	10	9	0.00	9	0.00	9	0.00	10	0.00
celar03	200	721	15	8	0.00	11	0.00	9	0.00	13	0.01
celar04	340	1009	16	9	0.00	12	0.01	9	0.00	13	0.01
celar05	200	681	15	9	0.01	11	0.00	9	0.00	13	0.01
celar06	100	350	11	10	0.00	11	0.00	10	0.00	11	0.01
celar07	200	817	18	11	0.00	13	0.00	12	0.01	15	0.00
celar08	458	1655	18	11	0.00	13	0.01	12	0.01	15	0.01
celar09	340	1130	18	11	0.01	13	0.01	12	0.00	15	0.01
celar11	340	975	15	8	0.00	11	0.00	9	0.00	13	0.01
graph01	100	358	25	8	0.00	9	0.01	9	0.00	14	0.00
graph02	200	709	51	6	0.00	11	0.00	7	0.01	20	0.02
graph03	100	340	22	5	0.00	8	0.00	6	0.01	14	0.00
graph04	200	734	55	6	0.00	12	0.01	7	0.00	19	0.02
graph05	100	416	26	8	0.00	9	0.00	9	0.00	15	0.00
graph06	200	843	53	8	0.00	12	0.01	9	0.00	21	0.02
graph07	200	843	53	8	0.00	12	0.01	9	0.00	21	0.02
graph08	340	1234	91	7	0.00	16	0.02	8	0.01	26	0.04
graph09	458	1667	118	8	0.01	17	0.03	9	0.01	29	0.06
graph10	340	1275	96	6	0.00	15	0.01	7	0.01	27	0.04
graph11	340	1425	98	7	0.00	17	0.01	8	0.01	27	0.05
graph12	340	1256	90	5	0.00	16	0.01	6	0.01	25	0.04
graph13	458	1877	126	6	0.00	18	0.02	7	0.01	31	0.07
graph14	458	1398	121	4	0.00	20	0.02	8	0.02	27	0.05

Table 3. MCSLB/MCSLB+ lower bounds

instance	MCSLB		MCSLB+ LBs						
			min-deg.		last-mcs		max-mcs		average
	LB	CPU	min-d.	least-c.	min-d.	least-c.	min-d.	least-c.	CPU
barley	5	0.01	6	6	6	6	6	5	0.06
diabetes	4	0.92	4	4	4	4	4	4	5.23
link	5	3.09	8	10	8	11	8	6	43.08
mildew	3	0.00	4	4	4	4	4	4	0.03
munin1	4	0.17	8	10	9	10	9	7	0.95
munin2	4	5.46	6	6	6	6	5	6	31.30
munin3	4	5.87	6	7	7	7	6	7	33.80
munin4	5	6.06	7	7	7	8	6	7	48.30
oesoca+	9	0.02	9	9	9	9	9	9	0.15
oow-trad	4	0.00	5	5	5	5	5	4	0.03
oow-bas	3	0.00	4	4	4	4	4	4	0.02
oow-solo	4	0.01	4	5	4	5	5	5	0.05
pathfinder	6	0.05	6	6	6	6	6	6	0.34
pignet2	5	59.60	28	39	30	39	16	18	509.60
pigs	3	1.01	7	7	7	7	6	6	5.12
ship-ship	5	0.01	6	6	6	6	6	6	0.06
water	8	0.00	8	8	8	8	8	8	0.04
wilson	3	0.00	3	3	3	3	3	3	0.01
celar01	10	1.20	12	13	12	14	12	13	17.08
celar02	9	0.06	9	10	9	10	9	9	0.30
celar03	9	0.23	11	12	11	13	12	12	1.54
celar04	11	0.66	11	13	12	13	12	13	4.85
celar05	9	0.23	12	13	12	13	12	12	1.80
celar06	11	0.06	11	11	11	11	11	11	0.33
celar07	12	0.24	13	15	13	15	13	15	1.66
celar08	12	1.45	13	15	14	15	13	15	17.04
celar09	12	0.82	13	15	14	15	14	15	4.62
celar11	10	0.66	11	13	12	13	12	12	4.43
graph01	9	0.06	9	15	11	14	12	13	0.45
graph02	8	0.24	12	19	12	19	14	13	2.54
graph03	6	0.06	9	13	10	14	9	13	0.49
graph04	8	0.25	12	20	13	20	14	16	1.95
graph05	9	0.06	10	15	11	16	11	14	0.47
graph06	9	0.26	12	22	14	22	14	15	2.22
graph07	9	0.26	12	22	14	22	14	15	2.15
graph08	9	0.76	17	26	18	26	18	20	5.89
graph09	9	1.43	17	30	20	28	22	23	11.51
graph10	8	0.77	18	26	17	26	19	22	6.25
graph11	8	0.80	15	27	18	27	17	26	6.53
graph12	7	0.76	15	25	16	25	17	21	5.92
graph13	8	1.48	18	32	19	31	20	28	12.89
graph14	5	1.35	19	28	20	28	21	25	10.11

5.1 The Heuristics

MMD computes exactly the degeneracy, by deleting a vertex of minimum degree and its incident edges in each iteration.

MMD+ is a derivative of MMD. Instead of deleting a vertex v of minimum degree, we contract it to a neighbour u. We consider three strategies how to select a neighbour:

- *min-d.* selects a neighbour with minimum degree, motivated by the idea that the smallest degree is increased as fast as possible in this way.
- *max-d.* selects a neighbour with maximum degree, motivated by the idea that we end up with some vertices of very high degree.
- *least-c.* selects a neighbour u of v, such that u and v have the least number of common neighbours, motivated by the idea to delete as few as possible edges in each iteration to get a high minimum degree.

MCSLB computes $|V|$ MCS-orderings ψ – one for each vertex as start vertex. It returns the maximum over these orderings of $MCSLB_\psi$, cf. [3].

MCSLB+ initially uses MCSLB to find a start vertex w with largest $MCSLB_\psi$. After each computation of an MCS-ordering, we select a vertex which we will contract, and then we apply MCSLB+ again. To reduce the CPU time consumption, $MCSLB_\psi$ is computed only for start vertex w instead of all possible start vertices. Three strategies for selecting a vertex v to be contracted are examined:

- *min-deg.* selects a vertex of minimum degree.
- *last-mcs* selects the last vertex in the just computed MCS-ordering.
- *max-mcs* selects a vertex with maximum visited degree in the just computed MCS-ordering.

Once a vertex v is selected, we select a neighbour u of v using the two strategies *min-d.* and *least-c.* that are already explained for MMD+.

5.2 Tables

Table 2 and 3 show the computational results for the described graphs. Table 2 shows the size of the graphs and the best known upper bound UB for treewidth, cf. [9,19]. Furthermore, Table 2 presents the lower bounds LB and the CPU times in seconds for the MMD and MMD+ heuristic with different selection strategies. Table 3, shows the same information for the MCSLB and MCSLB+ heuristics with different combinations of selection strategies. Because of space constraints and similarity, we only give the average running times.

In both tables, the graphs for which the lower and upper bound coincide are highlighted bold. In total, the treewidth of 8 graphs could be determined by MMD+, and one more by MCSLB+.

The MMD+ results show that the *min-d.* and in particular the *least-c.* strategy are very successful. The *max-d.* strategy is only marginal better than the degeneracy. For the MCSLB+ we observe that *min-deg.* and *last-mcs* in combination with *least-c.* outperform the other strategies. The added value of MCSLB+ to MMD+ is marginal, also because of the relatively large computation times.

The success of the *least-c.* strategy is explained as follows. When we contract an edge $\{v, w\}$, for each common neighbor x of v and w, the two edges $\{v, x\}$ and $\{w, x\}$ become the same edge in the new graph. Thus, with the *least-c.* strategy, we have more edges in the graph after contraction, which often leads to a larger minimum degree.

6 Discussion and Concluding Remarks

In this article, we examined two new methods MMD+ and MCSLB+ for treewidth lower bounds which are derivatives of the MMD and MCSLB heuristics. We showed the two corresponding decision problems to be NP-complete.

The practical experiments show that contracting edges is a very good approach for obtaining lower bounds for treewidth as it considerably improves known lower bounds. When contracting a vertex, it appears that the best is to contract to a neighbor that has as few as possible common neighbors. We can see that in some cases the MCSLB+ gives a slightly better lower bound than the MMD+. However, the running times of the MMD+ heuristic are almost always negligible, while the running times of the MCSLB+ heuristic can reach a considerable order of magnitude. Furthermore, we see that the strategy for selecting a neighbour u of v with the least number of common neighbours of u and v often performs best.

Finally, notice that although the gap between lower and upper bound could be significantly closed by contracting edges within the algorithms, the absolute gap is still large for many graphs (pignet2, graph*), which asks for research on new and better heuristics.

MMD+ is a simple method usually performing very well. However, it has its limits, e.g., on planar graphs. A planar graph always has a vertex of degree ≤ 5, and therefore MMD and MMD+ can never give lower bounds larger than 5. Similar behaviour could be observed in experiments on 'almost' planar graphs.

Apart from its function as a treewidth lower bound, the contraction degeneracy appears to be an attractive and elementary graph measure, worth further study. For instance, interesting topics are its computational complexity on special graph classes or the complexity of approximation algorithms with a guaranteed performance ratio. While we know the fixed parameter cases are polynomial time solvable, it is desirable to have algorithms that solve e.g., contraction degeneracy for small values of k efficiently in practice.

References

1. H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25:1305–1317, 1996.
2. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comp. Sc.*, 209:1–45, 1998.
3. H. L. Bodlaender and A. M. C. A. Koster. On the maximum cardinality search lower bound for treewidth, 2004. Extended abstract to appear in proceedings WG 2004.
4. F. Clautiaux, J. Carlier, A. Moukrim, and S. Négre. New lower and upper bounds for graph treewidth. In J. D. P. Rolim, editor, *Proceedings International Workshop on Experimental and Efficient Algorithms, WEA 2003*, pages 70–80. Springer Verlag, Lecture Notes in Computer Science, vol. 2647, 2003.
5. B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.
6. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1998.
7. V. Gogate and R. Dechter. A complete anytime algorithm for treewidth. To appear in proceedings UAI'04, Uncertainty in Artificial Intelligence, 2004.
8. A. M. C. A. Koster. *Frequency assignment - Models and Algorithms*. PhD thesis, Univ. Maastricht, Maastricht, the Netherlands, 1999.
9. A. M. C. A. Koster, H. L. Bodlaender, and S. P. M. van Hoesel. Treewidth: Computational experiments. In H. Broersma, U. Faigle, J. Hurink, and S. Pickl, editors, *Electronic Notes in Discrete Mathematics*, volume 8. Elsevier Science Publishers, 2001.
10. A. M. C. A. Koster, S. P. M. van Hoesel, and A. W. J. Kolen. Solving partial constraint satisfaction problems with tree decomposition. *Networks*, 40:170–180, 2002.
11. 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.
12. B. Lucena. A new lower bound for tree-width using maximum cardinality search. *SIAM J. Disc. Math.*, 16:345–353, 2003.
13. S. Ramachandramurthi. A lower bound for treewidth and its consequences. In E. W. Mayr, G. Schmidt, and G. Tinhofer, editors, *Proceedings 20th International Workshop on Graph Theoretic Concepts in Computer Science WG'94*, pages 14–25. Springer Verlag, Lecture Notes in Computer Science, vol. 903, 1995.
14. S. Ramachandramurthi. The structure and number of obstructions to treewidth. *SIAM J. Disc. Math.*, 10:146–157, 1997.
15. N. Robertson and P. D. Seymour. Graph minors. XX. Wagner's conjecture. To appear.
16. N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory Series B*, 63:65–110, 1995.
17. N. Robertson, P. D. Seymour, and R. Thomas. Quickly excluding a planar graph. *J. Comb. Theory Series B*, 62:323–348, 1994.
18. R. E. Tarjan and M. Yannakakis. Simple linear time algorithms to test chordiality of graphs, test acyclicity of graphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13:566–579, 1984.
19. Treewidthlib. http://www.cs.uu.nl/people/hansb/treewidthlib, 2004-03-31.