Degree-Based Treewidth Lower Bounds*

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

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

Abstract. Every lower bound for treewidth can be extended by taking the maximum of the lower bound over all subgraphs or minors. This extension is shown to be a very vital idea for improving treewidth lower bounds. In this paper, we investigate a total of nine graph parameters, providing lower bounds for treewidth. The parameters have in common that they all are the vertex-degree of some vertex in a subgraph or minor of the input graph. We show relations between these graph parameters and study their computational complexity. To allow a practical comparison of the bounds, we developed heuristic algorithms for those parameters that are NP-hard to compute. Computational experiments show that combining the treewidth lower bounds with minors can considerably improve the lower bounds.

1 Introduction

Many combinatorial optimisation problems take a graph as part of the input. If this graph belongs to a specific class of graphs, typically more efficient algorithms are available to solve the problem, compared to the general case. In case of trees for example, many NP-hard optimisation problems can be solved in polynomial time. Over the last decades, it has been shown that many NP-hard combinatorial problems can be solved in polynomial time for graphs with treewidth bounded by a constant. Until recently, it was assumed that these results were of theoretical interest only. By means of the computation of so-called exact inference in probabilistic networks [17] as well as the frequency assignment problem [15] in cellular wireless networks, it has been shown that such an algorithm to compute the optimal solution can be used in practice as well.

* 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.E. Nikoletseas (Ed.): WEA 2005, LNCS 3503, pp. 101–112, 2005.
© Springer-Verlag Berlin Heidelberg 2005

Polynomial time algorithms for solving combinatorial problems on a graph of bounded treewidth consist of two steps: (i) the construction of a tree decomposition of the graph with width as small as possible, and (ii) the application of dynamic programming on the tree decomposition to find the optimal solution of the combinatorial problem. Whereas the first step can be applied without knowledge of the application, the second step requires the development of an algorithm tailor-made for the specific application.

To exploit the full potential of tree decomposition approaches for as many combinatorial problems as possible, the first step is of fundamental importance. The smallest possible width of a tree decomposition is known as the treewidth of the graph. Computing the treewidth is however NP-hard [1]. To advance towards tree decompositions with close-to-optimal width, research in recent years has been carried out on practical algorithms for reduction and decomposition of the input graph [5, 6, 11], upper bounds [10, 9, 14], lower bounds [4, 7, 10, 18, 20], and exact algorithms (e.g. [12]).

In this paper, we research treewidth lower bounds that are based on the degree of specific vertices. Good treewidth lower bounds can be utilised to decrease the running time of branch-and-bound algorithms (see e.g. [12]). The better the lower bounds, the bigger the branches that can be pruned in a branch-and-bound method. Furthermore, treewidth lower bounds are useful to estimate the running times of dynamic programming methods that are based on tree decompositions. Such methods have running times that are typically exponential in the treewidth. Therefore, a large lower bound on the treewidth of a graph implies only little hope for an efficient dynamic programming algorithm based on a tree decomposition of that graph. In addition, lower bounds in connection with upper bounds help to assess the quality of these bounds.

Every lower bound for treewidth can be modified by taking the maximum of the lower bound over all subgraphs or minors. In [7, 8] this idea was used to obtain considerable improvements on two lower bounds: the minimum degree of a graph and the MCSLB lower bound by Lucena [18].

In this paper, we extend our research efforts to improve the quality of further known lower bounds in this way. One lower bound for treewidth is given by the second smallest degree, another one by the minimum over all non-adjacent pairs of vertices of the maximum degree of the vertices (cf. Ramachandramurthi [20]). Altogether, we examine nine parameters (defined in Section 2) and determine some relationships between them (see Section 3.1). We show that the second smallest degree over all subgraphs is computable in polynomial time, whereas the parameters for other combinations are NP-hard to compute (see Section 3.2). In this extended abstract, however, we omit full proofs. For the parameters that are NP-hard to compute, we develop several algorithms in Section 4.2 to obtain treewidth lower bounds heuristically. A computational evaluation (Section 4.3 and 4.4) of the algorithms shows that the heuristics where we combine a lower bound with edge contraction outperforms other strategies.

2 Preliminaries and Graph Parameters

Throughout the paper $G = (V, E)$ denotes a simple undirected graph. Unless otherwise stated, $n(G)$ (or simply n) denotes the number of vertices in G, i.e. $n := |V|$, and $m(G)$ (or simply m) denotes the number of edges $m := |E|$. Most of our terminology is standard graph theory/algorithm terminology. The open neighbourhood $N_G(v)$ or simply $N(v)$ of a vertex $v \in V$ is the set of vertices adjacent to v in G. As usual, the degree in G of vertex v is $d_G(v)$ or simply $d(v)$, and we have $d(v) = |N(v)|$. $N(S)$ for $S \subseteq V$ denotes the open neighbourhood of S, i.e. $N(S) = \bigcup_{s \in S} N(s) \setminus S$.

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. (We use $G' \subseteq G$ to denote that G' is a subgraph of G.) If we furthermore allow edge-contractions, we get a *minor* (denoted as $G' \preceq G$, if G' is a minor of G). Contracting edge $e = \{u, v\}$ in the graph $G = (V, E)$ is the operation that introduces a new vertex a_e and new edges such that a_e is adjacent to all the neighbours of u and v, and deletes vertices u and v and all edges incident to u or v.

Treewidth. The notions treewidth and tree decomposition were introduced by Robertson and Seymour in [21]. 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 each of the following holds: $\bigcup_{i \in I} X_i = V$; 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. The following lemma is well known and an important fact for proving the parameters, considered in this paper, to be treewidth lower bounds.

Lemma 1 (see e.g. [3]). *If G' is a minor of G, then $tw(G') \leq tw(G)$.*

Graph Parameters. We consider a number of graph parameters in this paper, all lower bounds on the treewidth of a graph, cf. Section 3. The *minimum degree* δ of a graph G is defined as usual: $\delta(G) := \min_{v \in V} d(v)$

The *δ-degeneracy* or simply the *degeneracy* δD of a graph G is defined in [2] to be the minimum number s such that G can be reduced to an empty graph by the successive deletion of vertices with degree at most s. It is easy to see that this definition of the degeneracy is equivalent (see [24]) to the following definition: $\delta D(G) := \max_{G'} \{\delta(G') \mid G' \subseteq G \wedge n(G') \geq 1\}$ The treewidth of G is at least its degeneracy (see also [14]). The *δ-contraction degeneracy* or simply the *contraction degeneracy* δC of a graph G was first defined in [7]. Instead of deleting a vertex v of minimum degree, we contract it to a neighbour u, i.e. we contract the edge $\{u, v\}$. This has been proven to be a very vital idea for obtaining treewidth lower bounds [7, 8]. The contraction degeneracy

is defined as the maximum over all minors G' of G of the minimum degree: $\delta C(G) := \max_{G'}\{\delta(G') \mid G' \preceq G \wedge n(G') \geq 1\}$

Let be given an ordering $v_1, ..., v_n$ of the vertices of G with $n \geq 2$, such that $d(v_i) \leq d(v_{i+1})$, for all $i \in \{1, ..., n-1\}$. The *second smallest degree* δ_2 of a graph G is defined as: $\delta_2(G) := d(v_2)$ Note that it is possible that $\delta(G) = \delta_2(G)$. Similar to the δ-degeneracy and δ-contraction-degeneracy we define the δ_2-degeneracy and δ_2-contraction-degeneracy. The *δ_2-degeneracy* $\delta_2 D$ of a graph $G = (V, E)$ with $n \geq 2$ is defined as follows: $\delta_2 D(G) := \max_{G'}\{\delta_2(G') \mid G' \subseteq G \wedge n(G') \geq 2\}$ The *δ_2-contraction degeneracy* $\delta_2 C$ of a graph $G = (V, E)$ with $n \geq 2$ is: $\delta_2 C(G) := \max_{G'}\{\delta_2(G') \mid G' \preceq G \wedge n(G') \geq 2\}$

In [19, 20], Ramachandramurthi introduced the parameter $\gamma_R(G)$ of a graph G and proved that this is a lower bound on the treewidth of G. $\gamma_R(G) := \min(n-1, \min_{v,w \in V, v \neq w, \{v,w\} \notin E} \max(d(v), d(w)))$ Note that $\gamma_R(G) = n - 1$ if and only if G is a complete graph on n vertices. Furthermore, note that $\gamma_R(G)$ is determined by a pair $\{v, w\} \notin E$ with $\max(d(v), d(w))$ is as small as possible. We say that $\{v, w\}$ is a non-edge determining $\gamma_R(G)$, and if $d(v) \leq d(w)$ then we say that w is a vertex determining $\gamma_R(G)$. Once again, we define the 'degeneracy' and 'contraction degeneracy' versions also for the parameter γ_R. The *γ_R-degeneracy* $\gamma_R D(G)$ of a graph $G = (V, E)$ with $n \geq 2$ is defined as follows: $\gamma_R D(G) := \max_{G'}\{\gamma_R(G') \mid G' \subseteq G \wedge n(G') \geq 2\}$ The *γ_R-contraction degeneracy* $\gamma_R C(G)$ of a graph $G = (V, E)$ with $n \geq 2$ is defined as: $\gamma_R C(G) := \max_{G'}\{\gamma_R(G') \mid G' \preceq G \wedge n(G') \geq 2\}$.

3 Theoretical Results

3.1 Relationships Between the Parameters

Lemma 2. *For a graph $G = (V, E)$ with $|V| \geq 2$, $x \in \{\delta, \delta_2, \gamma_R\}$ and $X \in \{D, C\}$, each of the following holds:*

1. $\delta(G) \leq \delta_2(G) \leq \gamma_R(G) \leq tw(G)$
2. $x(G) \leq xD(G) \leq xC(G) \leq tw(G)$
3. $\delta X(G) \leq \delta_2 X(G) \leq \gamma_R X(G) \leq tw(G)$
4. $\delta_2 X(G) \leq \delta X(G) + 1$
5. $\gamma_R X(G) \leq 2 \cdot \delta_2 X(G)$

It follows directly from Lemma 2 that all the parameters defined in Section 2 are lower bounds for treewidth. Furthermore, we see that the gap between the parameters δD and $\delta_2 D$, and between δC and $\delta_2 C$ can be at most one (see Lemma 2). In Section 3.2, we will see that $\delta_2 D$ can be computed in polynomial time. Therefore, Lemma 2 gives us a 2-approximation algorithm for computing the parameter $\gamma_R D$. Also in Section 3.2, we will see that $\gamma_R D$ is NP-hard to compute.

The next lemma shows some interesting properties of the parameter γ_R, when given a vertex sequence sorted according to non-decreasing degree.

Lemma 3. *Let be given a graph G on n vertices with $G \neq K_n$. Furthermore, let be given an ordering $v_1, ..., v_n$ of $V(G)$, such that $d(v_i) \leq d(v_{i+1})$, for all $i \in \{1, ..., n-1\}$. We define $j := \min\{i \in \{1, ..., n\} \mid \exists l \in \{1, ..., i-1\} : \{v_i, v_l\} \notin E(G)\}$. Then we have:*

1. *$d(v_j) = \gamma_R(G)$*
2. *$v_1, ..., v_{j-1}$ form a clique in G*

3.2 Computational Complexity of the Parameters

A Bucket Data Structure

In this section, we briefly describe a data structure that can be used in many of our algorithms. A more detailed description can be found in [24]. We extend the standard adjacency-list data structure of a graph $G = (V, E)$ in the following way. We store in doubly linked lists the adjacent vertices for every vertex of the graph, and we also use cross pointers for each edge $\{v_i, v_j\}$ (i.e. a pointer between vertex v_i in the adjacency-list for vertex v_j and vertex v_j in the adjacency-list for vertex v_i). In addition to this *advanced-adjacency-list*, we create $n = |V|$ buckets that can be implemented by doubly-linked lists $B_0, ..., B_{n-1}$. List B_d contains exactly those vertices with degree d. We maintain a pointer $p(v)$ for every vertex v that points to the exact position in the list B_d that contains v for the appropriate d.

Lemma 4 (see [24]). *Let be given a graph $G = (V, E)$ with $n = |V|$ and $m = |E|$. An algorithm performing a sequence of $O(n)$ vertex deletions and searches for a vertex with smallest or second smallest degree can be implemented to use $O(n + m)$ time.*

Known Results

It is easy to see that $\delta(G)$ and $\delta_2(G)$ can be computed in $O(n + m)$ time. Also the parameter $\gamma_R(G)$ can be computed in $O(n+m)$ time, see [19] or Section 4.1. Interestingly enough, the definition of the degeneracy as in [2] (see also Section 2) reflects an algorithm to compute this parameter: Successively delete a vertex of minimum degree and return the maximum of the encountered minimum degrees. Using the data structure described in this section, $\delta D(G)$ can be computed in time $O(n + m)$. Computing δC is NP-hard as is shown in [7].

$\delta_2 D$ Is Computable in Polynomial Time

A strategy to compute $\delta_2 D$ is as follows. We can fix a vertex v of which we suppose it will be the vertex of minimum degree in a subgraph G' of G with $\delta_2(G') = \delta_2 D(G)$. Starting with the original graph, we successively delete a vertex in $V(H) \setminus \{v\}$ of smallest degree, where H is the current considered subgraph of G (initially: $H = G$). Since we do not know whether our choice of v was optimal, doing this for all vertices $v \in V$ leads to a correct algorithm to compute $\delta_2 D(G)$. Using the bucket data structure, described above, this method

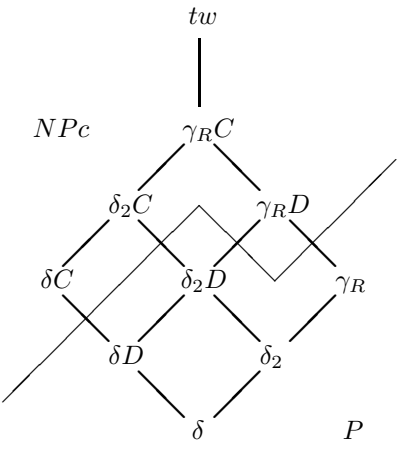

Fig. 1. An overview of some theoretical results

can be implemented to take $O(n \cdot m)$ time. We call this algorithm `Delta2D`. The following pseudo-code makes this algorithm more precise.

```
Algorithm Delta2D

 1   delta2D := 0
 2   for each v ∈ V do
 3       H := G
 4       repeat
 5           if δ2(H) > delta2D then delta2D := δ2(H) endif
 6           V* := V(H) \ {v}
 7           let u ∈ {w ∈ V* | ∄w' ∈ V* : dH(w') < dH(w)}
 8           H := H[V(H) \ {u}]
 9       until |V(H)| = 1
10   endfor
11   return delta2D
```

Lemma 5. *Algorithm* `Delta2D` *computes* $\delta_2 D(G)$ *and can be implemented to run in* $O(n \cdot m)$ *time, for a given connected graph* $G = (V, E)$ *with* $|V| \geq 2$.

NP-completeness Results

Here, we will state the computational hardness of the decision problems corresponding to the parameters $\gamma_R D$, $\gamma_R C$ and $\delta_2 C$.

Theorem 1. *Let* G *be a graph,* G' *be a bipartite graph and* k *be an integer. Each of the following is NP-complete to decide:* $\gamma_R D(G) \geq k$, $\gamma_R C(G') \geq k$ *and* $\delta_2 C(G) \geq k$.

Figure 1 represents some of the theoretical results. A thick line between two parameters indicates that the parameter below is smaller or equal to the

parameter above, as stated by Lemma 2. The thin line marks the border between polynomial computability and NP-hardness of the corresponding parameters (see Theorem 1 and other results in Section 3.2).

4 Experimental Results

In this section, we describe exact and heuristic algorithms, which we used in our experiments to compute the corresponding parameters.

4.1 Exact Algorithms

An implementation of algorithms to compute δ and δ_2 is straightforward. It is obvious that, in linear time, both parameters can be computed exactly. The parameters δD and $\delta_2 D$ were computed as described in Section 3.2. Ramachandramurthi shows in [19] that γ_R can be computed in $O(n + m)$ time. In our experiments, we use a different algorithm that does not use an adjacency matrix. See the full version of this article ([16]) for more details.

4.2 Heuristics

For the parameters that are NP-hard to compute, we have developed heuristics some of which are based on the polynomial counterparts.

γ_R-degeneracy: For the $\gamma_R D$, we developed three heuristics based on the following observation: Let $v_1, ..., v_n$ be a sorted sequence of the vertices according to non-decreasing degree in G, and let $\gamma_R(G)$ be determined by v_j for some $j > 1$ (see Lemma 3). Thus, v_j is not adjacent to some vertex v_k with $k < j$, whereas $v_1, ..., v_{j-1}$ induce a clique in G. Let V' be the set of all vertices v_i with $i < j$ and $\{v_i, v_j\} \notin E$. Now, for any subgraph $G' \subset G$ with $(\{v_j\} \cup V') \subseteq V(G')$, we have that $\gamma_R(G') \leq \gamma_R(G)$. Hence, only subgraphs without either v_j or V' are of further interest. Based on this observation, we implemented two heuristics. In the heuristic $\gamma_R D$-left, we remove the vertices in V' (the vertices that are more to the left in the sequence) from the graph and continue. Whereas in the heuristic $\gamma_R D$-right, we delete the vertex v_j (the vertex that is more to the right in the sequence) and go to the next iteration.

δ-contraction degeneracy: For computing lower bounds for δC, we have examined various strategies for contraction in [7]. The most promising one has been to recursively contract a vertex of minimum degree with a neighbour that has the least number of common neighbours (denoted as the least-c strategy).

δ_2-contraction degeneracy: For $\delta_2 C$ we implemented three heuristic algorithms. The first one, *all-v* is based on the polynomial time implementation for $\delta_2 D$. We fix all vertices once at a time and perform the δC heuristic (with least-c strategy) on the rest of the graph. The best second smallest degree recorded provides a lower bound on $\delta_2 C$. The other two $\delta_2 C$-heuristics are based on

the algorithms for δC. Instead of recording the minimum degree we also can record the second smallest degree (*M*aximum *S*econd *D*egree with contraction, abbreviated as MSD+). If we contract a vertex of minimum degree with one of its neighbours (according to the least-c strategy), we obtain the algorithm MSD+1. If the vertex of second smallest degree is contracted with one of its neighbours (also according to the least-c strategy), we obtain the algorithm MSD+2.

γ_R-contraction degeneracy: For $\gamma_R C$ the same strategies as for $\gamma_R D$ have been implemented. The only difference is that instead of removing all vertices in V' or v_j, we contract each of the vertices with a neighbour that is selected according to the least-c strategy. Inspired by the good results of the '$\delta_2 C$ all-v' heuristic, we furthermore implemented the all-v strategy as described above also for the γ_R-contraction degeneracy. The difference is that instead of computing δ_2 of each obtained minor, we now compute γ_R.

4.3 Experiments

The algorithms and heuristics described above have been tested on a large number of graphs from various application areas such as probabilistic networks, frequency assignment, travelling salesman problem and vertex colouring (see e.g. [7, 8] for details). 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. Many of the tested graphs as well as most of the experimental results on their treewidth (from, among others, [7, 8] and this article) can be obtained from [23].

In the tables below, we present the results for some selected instances only. The result of these representative instances reflect typical behaviour for the whole set of instances. The best known upper bound for treewidth (see [14]) is reported in the column with title UB. Columns headed by LB give treewidth lower bounds in the terms of the according parameter or a lower bound for the parameter. The best lower bounds in the tables are highlited in bold font. Values in columns headed by CPU are running times in seconds.

Table 1 shows the sizes of the graphs, and the results obtained for the treewidth lower bounds without contraction. These bounds are the exact parameters apart from the values for the two $\gamma_R D$-heuristics. As the computation times for δ, δ_2 and γ_R are negligible, we omit them in the table. Also the δD can be computed within a fraction of a second. The computational complexity of $\delta_2 D$ is $O(n)$ larger than the one of δD which is reflected in the CPU times for this parameter.

Table 2 shows the results for the same graphs as in Table 1. Furthermore, in Table 2, we give the treewidth lower bounds according to the parameters that involve contraction. For δC, we only give the results of the least-c strategy, as this seems to be the most promising one (see [7]). For $\delta_2 C$ and $\gamma_R C$, the results of the heuristics as described in Section 4.2 are shown.

Table 1. Graph sizes, upper bounds and lower bounds without contractions

instance	size			δ	δ_2	γ_R	δD		$\delta_2 D$		$\gamma_R D$			
											left		right	
	$\|V\|$	$\|E\|$	UB	LB	LB	LB	LB	CPU	LB	CPU	LB	CPU	LB	CPU
link	724	1738	13	0	0	0	4	0.01	4	3.67	4	0.01	4	0.01
munin1	189	366	11	1	1	1	4	0.00	4	0.23	4	0.00	4	0.00
munin3	1044	1745	7	1	1	1	3	0.01	3	6.70	3	0.02	3	0.01
pignet2	3032	7264	135	2	2	2	4	0.04	4	69.87	4	0.04	4	0.05
celar06	100	350	11	1	1	1	10	0.01	**11**	0.08	**11**	0.00	10	0.00
celar07pp	162	764	18	3	3	3	11	0.01	12	0.27	12	0.00	11	0.01
graph04	200	734	55	3	3	3	6	0.01	6	0.36	6	0.00	6	0.00
rl5934-pp	904	1800	23	3	3	3	3	0.01	3	5.33	3	0.01	3	0.01
school1	385	19095	188	1	1	1	73	0.04	74	7.89	75	0.03	73	0.03
school1-nsh	352	14612	162	1	1	1	61	0.02	62	5.69	62	0.03	61	0.02
zeroin.i.1	126	4100	50	28	29	32	48	0.00	48	0.58	**50**	0.01	**50**	0.01

Table 2. Treewidth lower bounds with contraction

instance	δC		$\delta_2 C$						$\gamma_R C$					
	least-c		all-v		MSD+1		MSD+2		left		right		all-v	
	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU
link	11	0.02	**12**	17.27	11	0.02	11	0.03	11	0.02	**12**	0.02	**12**	150.13
munin1	**10**	0.01	**10**	0.58	**10**	0.00	**10**	0.00	9	0.01	**10**	0.00	**10**	3.07
munin3	**7**	0.01	**7**	13.20	**7**	0.01	**7**	0.02	**7**	0.01	**7**	0.02	**7**	312.92
pignet2	38	0.11	**40**	369.00	39	0.12	39	0.14	38	0.12	39	0.12	**40**	11525.1
celar06	**11**	0.00	**11**	0.16	**11**	0.01	**11**	0.00	**11**	0.00	**11**	0.00	**11**	0.30
celar07pp	**15**	0.00	**15**	0.77	**15**	0.01	**15**	0.01	**15**	0.00	**15**	0.01	**15**	2.08
graph04	20	0.01	20	2.72	20	0.01	19	0.01	20	0.02	19	0.01	**21**	4.78
rl5934-pp	5	0.02	**6**	36.12	5	0.02	5	0.03	5	0.03	**6**	0.02	**6**	221.72
school1	122	0.48	124	180.30	123	0.48	122	0.51	122	0.45	122	0.49	**125**	215.35
school1-nsh	106	0.37	**108**	173.51	106	0.35	107	0.38	104	0.34	106	0.36	**108**	146.19
zeroin.i.1	**50**	0.03	**50**	6.25	**50**	0.03	**50**	0.03	**50**	0.03	**50**	0.03	**50**	5.43

4.4 Discussion

The results of algorithms and heuristics that do not involve edge-contractions (Table 1) show that the degeneracy lower bounds (i.e. the lower bounds involving subgraphs) are significantly better than the simple lower bounds, as could be expected. Comparing the results for δD and $\delta_2 D$, we see that in four cases we have that $\delta_2 D = \delta D + 1$. In the other seven cases $\delta_2 D = \delta D$. Bigger gaps than one between δD and $\delta_2 D$ are not possible (confirm Lemma 2). In some cases other small improvements (compared to δD and $\delta_2 D$) could be obtained by the heuristics for $\gamma_R D$. The two $\gamma_R D$-heuristics are all comparable in value and running times. Apart from the running times for computing $\delta_2 D$, the computation times are in all cases marginal, which is desirable for methods involving computing lower bounds many times (e.g. branch & bound). Even though the $\delta_2 D$

algorithm has much higher running times than the other algorithms in Table 1, it is still much faster than some heuristics with contraction. Furthermore, we expect that its running time could be improved by a more efficient implementation. No further investigations about parameters without contraction have been carried out as the parameters with contraction are of considerably more interest.

We can see that when using edge-contractions, the treewidth lower bounds can be significantly improved (compare Table 2 with Table 1). The results show that values for $\delta_2 C$ are typically equal or only marginal better than the value for δC. The same is true for $\gamma_R C$ with respect to $\delta_2 C$. The best results are obtained by the most time consuming algorithms: $\delta_2 C$ and $\gamma_R C$ with all-v strategy. By construction of the heuristic for $\gamma_R C$ with all-v strategy, it is clear that it is at least as good as the heuristic for $\delta_2 C$ with all-v strategy. Sometimes, it is even a little bit better. As in the case of the $\delta_2 D$ algorithm, the computation times of the $\delta_2 C$ and $\gamma_R C$ heuristics with all-v strategy could probably be improved by more efficient implementations. The other strategies for $\delta_2 C$ and $\gamma_R C$ are comparable in value and running times. No clear trend between them could be identified. In a few cases, we can observe that the gap between δC and $\delta_2 C$ is more than one. This does not contradict Lemma 2, because the considered values are not the exact values. Different strategies for heuristics can result in different values with larger gaps between them. With the same argument, we can explain that in a few cases lower bounds of one parameter that in theory is at least as good as another parameter can be smaller than lower bounds of the other parameter.

As said above, using γ_R instead of δ_2 in the degeneracy and contraction degeneracy heuristics, gives only small improvements in some cases. Therefore, the ratio of two between those parameters as stated in Lemma 2 is far from the ratios observed in our experiments. Proving a smaller ratio and/or finding a graph with ratio as large as possible, remains a topic for further research.

It was already remarked in [7] that the δ-contraction degeneracy of a planar graph can never be larger than 5. In fact, we have that $\delta C(G) \leq \delta_2 C(G) \leq \gamma(G) + 5$, where γ denotes the genus of a graph (see [24]). This behaviour can be observed in our experiments, e.g. for the graph rl5934-pp, which is expected to be nearly planar. However, the γ_R-contraction degeneracy might be larger than $\gamma(G) + 5$.

5 Conclusions

In this article, we continued our research in [7] on degree-based treewidth lower bounds, where we combined the minimum degree lower bound with subgraphs and edge-contraction/minors. Here, we applied this combination to two other treewidth lower bounds, namely the second smallest degree lower bound and the Ramachandramurthi lower bound [19].

We obtained theoretical results showing how the parameters are related to each other. We also examined the computational complexity of the parameters. Here, it is interesting to note that all contraction degeneracy problems are

NP-hard, while the degeneracy problems are polynomial, except for the γ_R-degeneracy, which has been shown to be NP-hard.

In our experiments, we could observe potent improvements when comparing the simple parameters with their degeneracy counterparts. An even bigger improvement was achieved when edge-contractions were involved. Therefore, we can conclude that the incorporation of contraction improves the lower bounds for treewidth considerably. However, the added value of $\delta_2 C$ and $\gamma_R C$ in comparison to δC is from a practical perspective marginal. The best lower bounds for $\delta_2 C$ and $\gamma_R C$ were obtained by heuristics with considerably long running times. Hence, if the lower bound has to be computed frequently, e.g. within a branch-and-bound algorithm, it is advisable to first compute a lower bound for δC, and only in tight cases using a slower but hopefully better lower bound.

It remains an interesting topic to research other treewidth lower bounds that can be combined with minors, in the hope to obtain large improvements. Furthermore, good lower bounds for graphs with bounded genus are desirable, because lower bounds based on δ, δ_2 or γ_R do not perform very well on such graphs (see [24]). However, treewidth lower bounds for planar graphs (i.e. graphs with genus zero) can be obtained e.g. by computing the branchwidth of the graph (see [13, 22]).

References

1. S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Alg. Disc. Meth.*, 8:277–284, 1987.
2. M. Behzad, G. Chartrand, and L. Lesniak-Foster. *Graphs and Digraphs.* Pindle, Weber & Schmidt, Boston, 1979.
3. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comp. Sc.*, 209:1–45, 1998.
4. H. L. Bodlaender and A. M. C. A. Koster. On the Maximum Cardinality Search lower bound for treewidth. In J. Hromkovič, M. Nagl, and B. Westfechtel, editors, *Proc. 30th International Workshop on Graph-Theoretic Concepts in Computer Science WG 2004*, pages 81–92. Springer-Verlag, Lecture Notes in Computer Science 3353, 2004.
5. H. L. Bodlaender and A. M. C. A. Koster. Safe separators for treewidth. In *Proceedings 6th Workshop on Algorithm Engineering and Experiments ALENEX04*, pages 70–78, 2004.
6. H. L. Bodlaender, A. M. C. A. Koster, F. v. d. Eijkhof, and L. C. van der Gaag. Preprocessing for triangulation of probabilistic networks. In J. Breese and D. Koller, editors, *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pages 32–39, San Francisco, 2001. Morgan Kaufmann.
7. H. L. Bodlaender, A. M. C. A. Koster, and T. Wolle. Contraction and treewidth lower bounds. In S. Albers and T. Radzik, editors, *Proceedings 12th Annual European Symposium on Algorithms, ESA2004*, pages 628–639. Springer, Lecture Notes in Computer Science, vol. 3221, 2004.
8. H. L. Bodlaender, A. M. C. A. Koster, and T. Wolle. Contraction and treewidth lower bounds. Technical Report UU-CS-2004-34, Dept. of Computer Science, Utrecht University, Utrecht, The Netherlands, 2004.

9. F. Clautiaux, S. N. A. Moukrim, and J. Carlier. Heuristic and meta-heuristic methods for computing graph treewidth. *RAIRO Oper. Res.*, 38:13–26, 2004.
10. 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.
11. F. v. d. Eijkhof and H. L. Bodlaender. Safe reduction rules for weighted treewidth. In L. Kučera, editor, *Proceedings 28th Int. Workshop on Graph Theoretic Concepts in Computer Science, WG'02*, pages 176–185. Springer Verlag, Lecture Notes in Computer Science, vol. 2573, 2002.
12. V. Gogate and R. Dechter. A complete anytime algorithm for treewidth. In proceedings UAI'04, Uncertainty in Artificial Intelligence, 2004.
13. I. V. Hicks. Planar branch decompositions I: The ratcatcher. INFORMS Journal on Computing (to appear, 2005).
14. 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.
15. 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.
16. A. M. C. A. Koster, T. Wolle, and H. L. Bodlaender. Degree-based treewidth lower bounds. Technical Report UU-CS-2004-050, Institute for Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands, 2004.
17. 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.
18. B. Lucena. A new lower bound for tree-width using maximum cardinality search. *SIAM J. Disc. Math.*, 16:345–353, 2003.
19. S. Ramachandramurthi. *Algorithms for VLSI Layout Based on Graph Width Metrics*. PhD thesis, Computer Science Department, University of Tennessee, Knoxville, Tennessee, USA, 1994.
20. S. Ramachandramurthi. The structure and number of obstructions to treewidth. *SIAM J. Disc. Math.*, 10:146–157, 1997.
21. N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of treewidth. *J. Algorithms*, 7:309–322, 1986.
22. P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.
23. Treewidthlib. http://www.cs.uu.nl/people/hansb/treewidthlib, 2004-03-31.
24. T. Wolle, A. M. C. A. Koster, and H. L. Bodlaender. A note on contraction degeneracy. Technical Report UU-CS-2004-042, Institute of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands, 2004.