Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

THOMAS WOLLE
ARIE M.C.A. KOSTER
HANS L. BODLAENDER

A Note on Contraction Degeneracy

http://www.zib.de/Optimization/Projects/DiscStruct/TACO/

ZIB-Report 04–43 (November 2004)

A Note on Contraction Degeneracy*

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

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

Abstract. The parameter contraction degeneracy — the maximum minimum degree over all minors of a graph — is a treewidth lower bound and was first defined in [3]. In experiments it was shown that this lower bound improves upon other treewidth lower bounds [3]. In this note, we examine some relationships between the contraction degeneracy and connected components of a graph, blocks of a graph and the genus of a graph. We also look at chordal graphs, and we study an upper bound on the contraction degeneracy. A data structure that can be used for algorithms computing the degeneracy and similar parameters, is also described.

1 Introduction

Since their introduction by Robertson and Seymour about twenty years ago, the notions of treewidth and treedecomposition have received a growing interest. An important reason for the popularity of these notions is the fact that many problems that are NP-hard to solve on general graphs, can be solved in linear or polynomial time (in the size of the input graph), using dynamic programming on a treedecomposition. However, this dynamic programming approach has a running time that is typically exponential in the width of the used treedecomposition. The smallest possible width of a treedecomposition of a graph is the treewidth of this graph. Therefore, lower bounds for treewidth inform us about the running times to be expected for such a dynamic programming approach.

One lower bound for treewidth is the contraction degeneracy — the maximum minimum degree over all minors of a graph. It was first defined in [3, 4] wherein also the NP-completeness of the corresponding decision problem was shown. The contraction degeneracy is not only a successful treewidth lower bound as was shown in experiments in [3, 4], but it also appears to be an interesting parameter on its own, due to its elementary formulation. In [5], we describe an algorithm to compute the contraction degeneracy in polynomial time on cographs.

* 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*).

In this note, we examine some basic properties of the parameter contraction degeneracy related to connected components and blocks, see Section 3. In Section 4, we will see that computing the contraction degeneracy of a chordal graph is easy. We already observed in [3, 4] that the contraction degeneracy is at most 5 for planar graphs. In Section 5, we generalise this to graphs with genus γ, and we obtain an upper bound on the contraction degeneracy based on the genus of a graph. Another upper bound on the contraction degeneracy based on the average degree is considered in Section 6. The running times of many algorithms that compute degree-based treewidth lower bounds (e.g. some of the algorithms in [4, 17]), might be reduced by using the data structure that we describe in Section 7.

2 Preliminaries

Throughout the paper $G = (V, E)$ denotes a simple undirected graph. Most of our terminology is standard graph theory/algorithm terminology. We use the following notation: $n(G) := |V(G)|$, $m(G) := |E(G)|$ and for a vertex v and an edge e, we write $G - v := G[V(G) \setminus v]$ and $G - e := (V, E \setminus e)$. 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. $N(S) = \bigcup_{s \in S} N(s) \setminus S$. We define:

$$\delta(G) := \min_{v \in V(G)} d(v)$$

Treewidth. A *treedecomposition* 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 treedecomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i| - 1$. The *treewidth* $tw(G)$ of G is the minimum width among all treedecompositions of G.

Edge-Contraction. A more formal approach to edge contractions as well as basic lemmas, which are summarised here, can be found in [23]. 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 deletes vertices u and v and all edges incident to u or v:

$$\begin{aligned} G/e &:= (V', E'), \text{ where} \\ V' &= \{a_e\} \cup V \setminus \{u, v\} \\ E' &= \{\ \{a_e, x\} \mid x \in N(\{u, v\})\} \cup E \setminus \{e' \in E \mid e' \cap e \neq \emptyset\} \end{aligned}$$

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, we let f represent 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: $H = G/E'$.

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* G' of G (following [9], denoted as $G' \preceq G$). We explicitely exclude the null graph (the empty graph on 0 vertices), as a subgraph or minor of a graph. It is known that the treewidth of a minor of G is at most the treewidth of G (see e.g. [2]).

Degeneracy. In [1], the degeneracy of a graph is defined as follows:

Definition 1 (See [1]). *The* degeneracy $s(G)$ *of a graph G is 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.*

We define the parameter δD of a graph G, and we show that $s(G) = \delta D(G)$.

$$\delta D(G) := \max_{G'}\{\delta(G') \mid G' \subseteq G\}$$

Lemma 1 (folklore). *For a graph G holds: $s(G) = \delta D(G)$.*

Proof. In each $G' \subseteq G$, there is a vertex with degree at most $\delta D(G)$. Therefore, we can reduce G to an empty graph by successively deleting a vertex of degree at most $\delta D(G)$. Hence, $s(G) \le \delta D(G)$.

On the other hand, let be $G' \subseteq G$, such that $\delta(G') = \delta D(G)$. Now, consider a sequence S of vertex deletion, such that the deleted vertices have degree at most $s(G)$. Let $v \in V(G')$ be the first vertex that is deleted in S. At this moment of deletion, v has degree at least $d_{G'}(v) \ge \delta(G')$, since v is the first vertex deleted in G'. Furthermore, we know that at the moment of deleting v, we have that the degree of v is at most $s(G)$. Hence, altogether, we have that $s(G) \ge d_{G'}(v) \ge \delta(G') = \delta D(G)$. $\square$

In the following, we use $\delta D(G)$ to denote the degeneracy of a graph. The minimum degree of a graph is a lower bound for its treewidth and the treewidth of G cannot increase by taking subgraphs. Hence, the treewidth of G is at least its degeneracy (see also [16]). It is interesting that Definition 1 reflects an algorithm to compute the degeneracy: Successively delete a vertex of minimum degree and return the maximum of the encountered minimum degrees.

Contraction Degeneracy. The parameter δC — the contraction degeneracy — was first defined in [3] as a lower bound for treewidth.

As mentioned above, the degeneracy lower bound for treewidth can be computed by repeatedly deleting minimum degree vertices, and keeping track of the encountered maximum minimum degree.

Instead of deleting these vertices and edges while recording the minimum degree of the occurring subgraphs, contracting edges incident to those vertices experimentally proved to be a very good idea for better treewidth lower bounds [3, 11]. Inspired by this lower bound, the parameter δC and the according decision problem were defined.

Definition 2. *The* contraction degeneracy δC *of a graph* G *is defined as follows:*

$$\delta C(G) := \max_{G'}\{\delta(G') \mid G' \preceq G \wedge |V(G')| \geq 1\}$$

Note that $\delta C(G)$ is defined as the maximum over all minors G' of G of the minimum degree of G'. Using contractions instead of minors in this definition does not lead to an equivalent notion, unless the considered graph G is connected. If G is not connected, it might be necessary to delete one or more connected components, to obtain a minor G' of G with maximum minimum degree. Apart from its role as a treewidth lower bound, it is an interesting problem also due to its elementary formulation, but little is known about it. The decision problem corresponding to δC 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?

Contraction Degeneracy is NP-complete for bipartite graphs [3]. That is why we look at special graph classes in Section 4 and in [5].

3 Connected Components and Blocks

In this section, we consider the contraction degeneracy of connected components and blocks of a graph, compared to the contraction degeneracy of the graph itself. The following lemma is easy to see.

Lemma 2. *For a graph* G *holds:*

$$\delta C(G) = \max_{G_c}\{\delta C(G_c) \mid G_c \textit{ is a connected component of } G\}$$

The maximal 2-connected subgraphs of G are the blocks of G (blocks are also called 2-connected components or biconnected components). Separating the graph into smaller subgraphs can help to solve problems or to compute graph parameters. That is why we are interested whether we can compute the contraction degeneracy of a graph, given the contraction degeneracy of the blocks of a graph. For the treewidth of a graph, we have the following lemma.

Lemma 3 (e.g. [2, 21]). *For a graph G holds:*

$$tw(G) = \max\{tw(G') \mid G' \textit{ is a block of } G\}$$

Unlike the treewidth, the contraction degeneracy of a graph does not equal the maximum contraction degeneracy over its blocks, as we can see from the example in Figure 1. It is easy to see that $\delta C(G) = 5$ and $\delta C(G_1) = \delta C(G_2) = 4$. However, we can prove the following lemma.

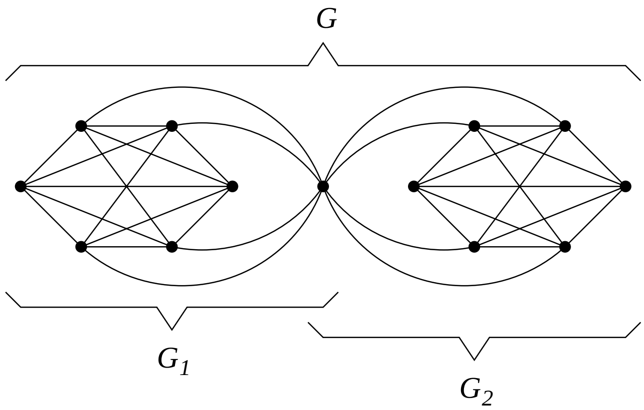

Fig. 1. Graph G with two blocks G_1 and G_2.

Lemma 4. *For a graph G holds:*

$$\delta C(G) \geq \max\{\delta C(G') \mid G' \textit{ is block of } G\}$$

Proof. This is easy to see, since all vertices and edges not belonging to the block G' with maximum contraction degeneracy, can be deleted in G'. Hence, G' is a minor of G and therefore $\delta C(G) \geq \delta C(G')$. □

This has interesting consequences for the second smallest degree of a contraction or minor. The parameter $\delta C(G)$ is defined to be the maximum over all minors G' of G of the smallest degree in G'. We can define the analogous parameter $\delta_2 C$ for the second smallest degree and prove that this is also a lower bound for the treewidth. See [17], for an alternative proof of $\delta_2 C$ being a treewidth lower bound, and also for an experimental evaluation.

Definition 3.

$$\delta_2(G) := \textit{ is the second smallest degree of } G$$
$$\delta_2 C(G) := \max_{G'}\{\delta_2(G') \mid G' \preceq G \wedge n(G') \geq 2\}$$

Lemma 5. *For a graph G, it holds that:*

$$\delta_2 C(G) \leq tw(G)$$

Proof. Consider a minor G' of G. Let v be a vertex of minimum degree in G'. If $d_{G'}(v) = 0$, then $\delta_2(G') = \delta(G' - v) \leq tw(G' - v) \leq tw(G)$, since the minimum degree of a graph is a lower bound for its treewidth.

If $d_{G'}(v) \geq 1$, then let $v_1, v_2, ...v_{|V(G')|}$ be the sequence of vertices of G' ordered according to nondecreasing degree. We take $c := \left\lceil \frac{d_{G'}(v_2)}{d_{G'}(v_1)} \right\rceil$ copies of G', and we identify all vertices corresponding to v_1 as a single vertex v. Call the resulting graph G''. Let u be a vertex in G'' corresponding to a v_2 in G'. Clearly, we have that $c \cdot d_{G'}(v_1) = d_{G''}(v) \geq d_{G''}(u) = d_{G'}(v_2)$. Therefore, u is a vertex of smallest degree in G''. From Lemma 3 follows that $tw(G') = tw(G'')$. Because u is a vertex with smallest degree, we have that $d_{G''}(u) \leq tw(G'') = tw(G') \leq tw(G)$.

Since this holds for all minors G' of G, we have $\delta_2 C(G) \leq tw(G)$. □

4 Chordal Graphs

Chordal graphs (also called triangulated graphs) form a special class of graphs that can be represented by a tree. In a chordal graph, each cycle $C = v_1, ..., v_p$ of length $p \geq 4$ has a chord, i.e. an edge connecting two non-consecutive vertices of C. In other words, chordal graphs do not contain any C_p as an induced subgraph for $p \geq 4$. All maximal cliques of a chordal graph can be arranged in a tree structure, such that a set of pairwise non-disjoint cliques forms a subtree of the tree (see e.g. [2]). This clique tree is often very useful to solve problems. However, exploiting this clique tree to compute the contraction degeneracy is not necessary. Note that chordal graphs can be recognised in linear time, and also a maximum clique of a chordal graph can be computed in that time (see e.g. [12]).

Lemma 6. *Let $G = (V, E)$ be a chordal graph and let $W \subseteq V$ be a maximum clique in G. Then we have:*

$$\delta C(G) = |W| - 1$$

Proof. For any graph G it holds: $\delta C(G) \leq tw(G)$, see [3]. If G is a chordal graph with maximum clique W, we have (see e.g. [2]): $tw(G) = |W| - 1$, and therefore: $\delta C(G) \leq |W| - 1$. Note that $G[W]$ is a minor of G and $\delta C(G) \geq \delta(G[W]) = |W| - 1$. □

As mentioned in the previous proof, it holds for a chordal graph with maximum clique W that $tw(G) = |W| - 1$ (see e.g. [2]). Therefore with Lemma 6, we have the following corollary.

Corollary 1. *For a chordal graph G holds:*

$$\delta C(G) = tw(G)$$

5 Graphs of Genus γ

Harary writes in [14] that every planar graph G with at least four vertices has at least four vertices of degree not exceeding five. Therefore, we have for a planar graph G that $\delta(G) \leq \delta_2(G) \leq 5$. As a consequence of this, we have that $\delta C(G) \leq \delta_2 C(G) \leq 5$, since planar graphs a closed under taking minors. This fact could also be observed in [3, 4, 17] in experiments computing δC. In this section, we examine a few basic observations, similar to the result above and related to the genus of a graph: the number of handles needed on a sphere to embed the graph.

A graph G is embeddable in a surface S when it can be drawn on S in such a way that no two edges intersect apart from intersecting at vertices. G is planar, iff G can be embedded in the plane or in the sphere. We can generalise this concept by changing the underling surface S. One way of changing S is to add a handle to it. A handle is a kind of bridge over which an edge can go to avoid crossing the edge below the handle, see e.g. [13, 14]. The closed orientable surfaces S_0, S_1, S_2, ... are defined as follows. S_0 is the sphere, and S_{n+1} is obtained by adding a handle to S_n. The genus of S_i is i, for all $i \geq 0$. A region of a graph embedding is a component of the surface the graph is embedded in, obtained by deleting the image of the graph from the surface. A cellular embedding of a graph G in S is an embedding such that each region is topologically equivalent to an open disk. A face of an embedding is the union of a region and its boundary. Introductory chapters to planarity and generalisations to higher genus can be found in e.g. [13, 14]. See [18, 19] for more details about graphs on surfaces.

Definition 4. *The* genus $\gamma(G)$ *of the graph* G *is defined as follows.*

$$\gamma(G) := \min\{i \mid G \textit{ can be embedded in } S_i\}$$

The corresponding decision problem: 'Given a graph G and an integer $k \geq 0$, is $\gamma(G) \leq k$?' is NP-complete (see [22]). However, if k is fixed, there is a polynomially bounded algorithm for computing the genus (see [10]). The most well known and well studied case are the planar graphs, i.e. graphs of genus 0 (see e.g. [8, 15, 6]). Given a graph G and a minor G' of G, it is easy to see that $\gamma(G') \leq \gamma(G)$.

Theorem 1 (see e.g. [13]). *Let* $G = (V, E)$ *be a connected graph with* n *vertices and* m *edges, and let* G *be cellularly embedded in* S_γ*. Let* f *be the number of faces of this embedding. Then we have that* $n - m + f = 2 - 2 \cdot \gamma$*.*

Since each face is surrounded by at least three edges, and each edge separates at most two faces, we have the edge-face inequality $3 \cdot f \leq 2 \cdot m$ (see e.g. [13]). With the last two formulas together, we obtain an upper bound on the number of edges of a graph with genus γ and n vertices (see e.g. [19]):

$$m \leq 3 \cdot (n - 2 + 2 \cdot \gamma) \tag{1}$$

Lemma 7. *Given a graph* G *with* n *vertices and genus* γ*, the following holds:*

1. $\delta(G) \leq 6 + \frac{12 \cdot \gamma - 12}{n}$
2. $\delta_2(G) \leq 6 + \frac{12 \cdot \gamma - 6}{n-1}$

Proof. (1.) It is obvious that G has at least $\frac{\delta(G) \cdot n}{2}$ edges. Hence, we have $\frac{\delta(G) \cdot n}{2} \leq 3 \cdot (n - 2 + 2 \cdot \gamma)$, from which we can derive $\delta(G) \leq 6 + \frac{12 \cdot \gamma - 12}{n}$. (2.) At least $n-1$ vertices have degree $\delta_2(G)$ in G. Therefore, G has at least $\frac{\delta_2(G) \cdot (n-1)}{2}$ edges, which implies $\delta_2(G) \leq 6 + \frac{12 \cdot \gamma - 6}{n-1}$. □

Theorem 2 (see e.g. [19]). *If* $n \geq 3$ *then*

$$\gamma(K_n) = \left\lceil \frac{(n-3) \cdot (n-4)}{12} \right\rceil$$

Lemma 8. *For any given graph* G *with at least two vertices, it holds:*

$$\delta_2 C(G) \leq 5 + \gamma(G)$$

Proof. With Lemma 7, we have:

$$\delta_2 C(G) = \max_{G' \preceq G,\ n(G') \geq 2} \delta_2(G') \leq \max_{G' \preceq G,\ n(G') \geq 2} \left(6 + \frac{12 \cdot \gamma(G') - 6}{n(G') - 1} \right)$$

We first consider minors with at least 13 vertices. The following is easy to see:

$$\max_{G' \preceq G,\ n(G') \geq 13} \left(6 + \frac{12 \cdot \gamma(G') - 6}{n(G') - 1} \right) \leq \left(6 + \frac{12 \cdot \gamma(G) - 6}{12} \right) = 5.5 + \gamma(G)$$

We will show the same result for minors G' with at most 12 vertices in a case distinction, using Lemma 7, the fact that the genus does not increase when taking minors and that $\delta(G') \leq \delta_2(G') \leq n(G') - 1$. Let G' be a minor of G with at least 2 and at most 12 vertices.

Case 0: $\gamma(G') = 0$. We have that $\delta_2(G') \leq 6 - \frac{6}{n(G')-1}$, which implies $\delta_2(G') \leq 5$ for all G'. Therefore, $\delta_2(G') \leq 5 \leq 5 + \gamma(G)$.

Case 1: $\gamma(G') = 1$. Here, we have that $\delta_2(G') \leq 6 + \frac{6}{n(G')-1}$. If $n(G') \leq 7$, then $\delta_2(G') \leq 6$; and if $8 \leq n(G') \leq 12$, then $\delta_2(G') \leq 6 + \frac{6}{n(G')-1} < 7$. Hence, $\delta_2(G') \leq 6 \leq 5 + \gamma(G)$.

Case 2: $\gamma(G') = 2$. To show $\delta_2(G') \leq 7 \leq 5 + \gamma(G)$, we distinguish subcases.

Subcase 2a: $n(G') \leq 8$. It is clear that $\delta_2(G') \leq 7$.

Subcase 2b: $n(G') = 9$. $G' \neq K_9$, since $\gamma(K_9) = 3$ (see Theorem 2), therefore, $G' \subseteq K_9 - e$ for an edge e. The two endpoints of e have degree at most 7 in G', hence $\delta_2(G') \leq 7$.

Subcase 2c: $n(G') = 10$. If $\delta_2(G') \geq 8$, then there are at least 9 vertices of degree at least 8 in G'. Thus, there are at least $\frac{8 \cdot 9}{2} = 36$ edges. However, with inequality (1), there are at most $3 \cdot (n-2+2 \cdot \gamma) = 36$ edges. Hence, $G' = K_1 \cup K_9$, and therefore, $\gamma(G') \geq 3$. This is a contradiction, so $\delta_2(G') \leq 7$.

Subcase 2d: $n(G') = 11$. We have $\delta_2(G') \leq 6 + \frac{18}{n(G')-1} < 8$. Hence, $\delta_2(G') \leq 7$.

Case 3: $\gamma(G') = 3$. Again, we use subcases to show $\delta_2(G') \leq 8 \leq 5 + \gamma(G)$.

Subcase 3a: $n(G') \leq 9$. It is clear that $\delta_2(G') \leq 8$.

Subcase 3b: $n(G') = 10$. $G' \neq K_{10}$, since $\gamma(K_{10}) = 4$ (see Theorem 2), therefore, $G' \subseteq K_{10} - \{e\}$ for an edge e. The two endpoints of e have degree at most 8 in G', hence $\delta_2(G') \leq 8$.

Subcase 3c: $n(G') = 11$. If $\delta_2(G') \geq 9$, then there are at least 10 vertices of degree at least 9 in G'. Thus, there are at least $\frac{9 \cdot 10}{2} = 45$ edges. However, with inequality (1), there are at most $3 \cdot (n-2+2 \cdot \gamma) = 45$ edges. Hence, $G' = K_1 \cup K_{10}$, and therefore, $\gamma(G') \geq 4$. This is a contradiction, so $\delta_2(G') \leq 8$.

Subcase 3d: $n(G') = 12$. We have $\delta_2(G') \leq 6 + \frac{30}{n(G')-1} < 9$. Hence, $\delta_2(G') \leq 8$.

Case 4: $\gamma(G') = 4$. Once more, considering subcases helps to show $\delta_2(G') \leq 9 \leq 5 + \gamma(G)$.

Subcase 4a: $n(G') \leq 10$. It is clear that $\delta_2(G') \leq 9$.

Subcase 4b: $n(G') = 11$. $G' \neq K_{11}$, since $\gamma(K_{11}) = 5$ (see Theorem 2), therefore, $G' \subseteq K_{11} - \{e\}$ for an edge e. The two endpoints of e have degree at most 9 in G', hence $\delta_2(G') \leq 9$.

Subcase 4c: $n(G') = 12$. We have $\delta_2(G') \leq 6 + \frac{42}{n(G')-1} < 10$. Hence, $\delta_2(G') \leq 9$.

Case 5: $\gamma(G') = 5$. In this case, we have that $\delta_2(G') \leq 6 + \frac{54}{n(G')-1}$. If $n(G') \leq 11$, then $\delta_2(G') \leq 10$; and if $n(G') = 12$, then $\delta_2(G') \leq 10$, since in that case $6 + \frac{54}{n(G')-1} < 11$. Hence, $\delta_2(G') \leq 10 \leq 5 + \gamma(G)$.

Case 6: $\gamma(G') = 6$. For $n(G') \leq 12$, we automatically have that $\delta_2(G') \leq 11 \leq 5 + \gamma(G)$.

This case distinction is exhaustive, since a graph with at most 12 vertices can have genus at most 6, as can be derived from Theorem 2. The lemma now follows. $\square$

It is easy to see that for any graph G with at least two vertices, it holds that $\delta C(G) \leq \delta_2 C(G)$ (see e.g. [17]). Therefore, we have:

$$\delta C(G) \leq \delta_2 C(G) \leq \gamma(G) + 5$$

Note that $\forall G : \delta C(G) \leq \gamma(G) + 5$ can be proven directly by a similar and easier version of the proof of Lemma 8.

Ramachandramurthi defined in [20] the parameter $\gamma_R(G)$, and he proved that this is a lower bound on the treewidth of G (see [20]).

$$\gamma_R(G) := \min(n-1, \min_{v,w \in V, v \neq w, \{v,w\} \notin E} \max(d(v), d(w)))$$

The γ_R-contraction degeneracy $\gamma_R C(G)$ of a graph was first defined in [17].

$$\gamma_R C(G) := \max_{G'} \{\gamma_r(G') \mid G' \text{ is a minor of } G \wedge |V(G')| \geq 2\}$$

To prove a similar statement as in Lemma 8 for $\gamma_R C$ might be more complicated, because it seems to be difficult to combinatorially capture the property that the considered vertices must be nonadjacent. However, we can prove the following.

Lemma 9. *Let G be a graph of genus γ. Then there exists a function $c(\gamma)$, such that $\gamma_R(G) \leq c(\gamma)$.*

Proof. Let be given an ordering $v_1, ..., v_n$ of the vertices of G, such that $d(v_i) \leq d(v_{i+1})$, for all $i \in \{1, ..., n-1\}$. We define $\delta_i(G) := d_G(v_i)$, for all $i \in \{1, ..., n\}$.

Let $x(\gamma)$ be the minimum number, such that every graph of genus γ does not contain $K_{x(\gamma)}$ as a subgraph. (For example, for planar graphs, i.e. $\gamma = 0$, we have $x(0) = 5$.) Now, note that any $x(\gamma)$ vertices in G do not form a clique. This is also true for the $x(\gamma)$ vertices that have smallest degree in G and hence, we have that $\gamma_R(G) \leq \delta_{x(\gamma)}(G)$.

We observe that G has at least $n - (x(\gamma) - 1)$ vertices of degree at least $\delta_{x(\gamma)}(G)$. Therefore, G has at least $\frac{\delta_{x(\gamma)}(G) \cdot (n-(x(\gamma)-1))}{2}$ edges. Furthermore, with inequality (1), G has at most $3(n - 2 + 2 \cdot \gamma)$ edges. Altogether, this yields for a graph G with at least $x(\gamma)$ vertices:

$$\gamma_R(G) \leq \delta_{x(\gamma)}(G) \leq 6 \cdot \frac{n - 2 + 2 \cdot \gamma}{n - x(\gamma) + 1}$$

If the graph G has at most $x(\gamma) - 1$ vertices, then $\delta_{x(\gamma)}(G)$ is not defined and $\gamma_R(G) \leq x(\gamma) - 2$. Since γ and $x(\gamma)$ are constant, it is easy to see that:

$$\lim_{n \to \infty} 6 \cdot \frac{n - 2 + 2 \cdot \gamma}{n - x(\gamma) + 1} = 6$$

Therefore, we define:

$$c(\gamma) := \max_{n \geq x(\gamma)} \left\lfloor 6 \cdot \frac{n - 2 + 2 \cdot \gamma}{n - x(\gamma) + 1}, x(\gamma) - 2 \right\rfloor < \infty \tag{2}$$

From the above-mentioned, it easily follows that $\gamma_R(G) \leq c(\gamma)$. □

It is interesting to note that we can conclude from the previous proof that for any fixed γ and $x(\gamma)$, $\delta_{x(\gamma)}(G)$ will be at most 6, when n is increased towards infinity. Since the genus does not increase when taking minors and $x(\gamma)$ is monotone in γ, we have the following corollary.

Corollary 2. *Let G be a graph of genus γ. Then $\gamma_R C(G) \leq c(\gamma)$, where $c(\gamma)$ is as in equation (2).*

That the genus $\gamma(G)$ of a graph G determines an upper bound on $\gamma_R(G)$ can also be derived from the fact that $\gamma_R C(G) \leq 2 \cdot \delta_2 C(G)$ (see [17]) and Lemma 8. Taking both together, we have $\gamma_R C(G) \leq 10 + 2 \cdot \gamma(G)$.

These upper bounds on $\delta C(G)$, $\delta_2 C(G)$ and $\gamma_R(G)$ are not always sharp in the following sense. For a graph G on n vertices, we have that $\delta C(G) \leq n-1 = O(n)$, while there are graphs G on n vertices (e.g. K_n, see Theorem 2), such that $\gamma(G) = \Theta(n^2)$. The same is true for $\delta_2 C(G)$ and $\gamma_R(G)$, since $\delta_2 C(G), \gamma_R(G) = O(\delta C(G))$ (see [17]).

The genus of a graph determines not only an upper bound on $\delta C(G)$ and $\delta_2 C(G)$; it also determines a lower bound. Graph G is a minimal forbidden minor for S, if G cannot be embedded in S, but every proper minor of G can be embedded in S. The next theorem is the excluded minor theorem, due to Robertson and Seymour.

Theorem 3 (see e.g. [19]). *For each surface S, the set of all minimal forbidden minors for S is finite.*

Lemma 10. *Let G be a graph of genus γ. Then there exist functions $c_1(\gamma)$, $c_2(\gamma)$ and $c_3(\gamma)$, such that:*

$$c_1(\gamma) \leq \delta C(G); \qquad c_2(\gamma) \leq \delta_2 C(G); \qquad c_3(\gamma) \leq \gamma_R C(G);$$

Proof. Let $F(S)$ denote the set of all minimal forbidden minors for S. We know that G can be embedded in S_γ, but not in $S_{\gamma-1}$. Therefore, G contains a minor G', such that $G' \in F(S_{\gamma-1})$. Since $F(S_i)$ is finite (Theorem 3), we can easily define $c_1(\gamma)$ to be the minimum over all graphs in $F(S_{\gamma-1})$ of the minimum degree of the graph. It is easy to see that we have:

$$c_1(\gamma) := \min_{G_1 \in F(S_{\gamma-1})} \delta(G_1) \leq \delta(G') \leq \delta C(G)$$

Hence, $c_1(\gamma)$ only depends on γ and it is a lower bound on $\delta C(G)$ for any graph G of genus γ. In the same way, we can use $c_2(\gamma) := \min_{G_1 \in F(S_{\gamma-1})} \delta_2(G_1) \leq \delta_2 C(G)$ and $c_3(\gamma) := \min_{G_1 \in F(S_{\gamma-1})} \gamma_R(G_1) \leq \gamma_R C(G)$. □

Note that the proof of Lemma 10 is nonconstructive, due to its use of the excluded minor theorem.

Even though $F(S)$ is finite for each surface S, its size can grow rapidly when the genus of the surface increases. For example, $F(S_0) = \{K_5, K_{3,3}\}$. Therefore, $c_1(1) = c_2(1) = c_3(1) = 3$. However, Cattell et al. and Mohar and Thomassen mention in [7, 19] that $F(S_1)$ probably contains more than 2000 graphs.

6 An Upper Bound on the Contraction Degeneracy

The contraction degeneracy is a very successful treewidth lower bound. However, as we have seen, it has its limits, e.g. on planar graphs (see also Section 5). For every planar graph G, we have that $\delta C(G) \leq 5$, whilst the treewidth of G can be arbitrarily large. Therefore, it is very interesting to have upper bounds on the contraction degeneracy. Such upper bounds on lower bounds can inform us about the perspective to improve these lower bounds. As seen in Section 5, the genus of the graph determines an upper bound for the contraction degeneracy. However, it is NP-hard to compute the genus of a graph [22]. Another idea for a contraction degeneracy upper bound is to take the maximum over all minors of $G = (V, E)$ of the average degree $\bar{d}$ of the minor.

$$\bar{d}(G) := \frac{2 \cdot m(G)}{n(G)} \qquad\qquad \bar{d}C(G) := \max_{G'} \{\bar{d}(G') \mid G' \text{ is a minor of } G\}$$

It is clear that $\delta C(G) \leq \bar{d}C(G)$ for all graphs G. Unfortunately, computing this parameter is also NP-hard.

Lemma 11. *Let be given a graph G and an integer $k \geq 0$. The problem to decide whether $\bar{d}(G) \geq k$ is NP-complete.*

Proof. The proof is very similar to the proof that the problem Contraction Degeneracy is NP-complete, see [4]. Let (G, k) be a Vertex Cover instance, with $G = (V, E)$, $n := |V|$, $n > 2$, $1 \leq k < n$, $m := |E|$, $m \geq 1$. We construct the graph G' as in Figure 2.

$$\begin{aligned}
G' &:= (V', E') \text{ where} \\
V' &= \{u_1, ..., u_k\} \cup V \cup \{w_1, ..., w_{4n}\} \\
E' &= \{\ \{u_i, v\} \mid i \in \{1, ..., k\}, v \in V\} \cup \\
&\quad\ \{\ \{v_i, v_j\} \mid v_i, v_j \in V, v_i \neq v_j, \{v_i, v_j\} \notin E\} \cup \\
&\quad\ \{\ \{v, w_i\} \mid v \in V, i \in \{1, ..., 4n\}\} \cup \{\ \{w_i, w_j\} \mid i \neq j, i, j \in \{1, ..., 4n\}\}
\end{aligned}$$

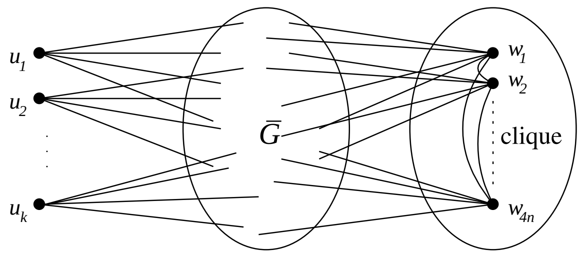

Fig. 2. Graph G'

The constructed instance of the problem stated in the lemma is $(G', 5n-1)$. We will now show that there is a vertex cover for G of size at most k, if and only if $\bar{d}C(G') \geq 5n-1$.

Claim. If there is a vertex cover for G of size at most k, then $\bar{d}C(G') \geq 5n-1$.

Proof. Let be given a vertex cover of size at most k; add some vertices to it to obtain a vertex cover $V_1 = \{v_1, ..., v_k\}$ of size exactly k. Now we define the contraction-set $E_1 := \{\ \{u_i, v_i\} \mid i = 1, ..., k\}$. We will show that G'/E_1 is a clique. Assume there is a non-edge e in G'/E_1. By the construction this can only be between two vertices x and y in V. Hence, let be $e = \{x, y\}$. Therefore, e is an edge in G, and that is why $e \cap V_1 \neq \emptyset$. W.l.o.g. let be $x \in V_1$. That means there is an $i \in \{1, ..., k\}$, with $x = v_i \in V_1$. Note that u_i was contracted to $x = v_i$, and thus, v_i is adjacent to any vertex in V. This is a contradiction. Therefore, G'/E_1 is a clique with exactly $5n$ vertices, and hence, $\bar{d}C(G') \geq \bar{d}(G'/E_1) = 5n-1$. $\diamond$

Claim. If $\bar{d}C(G') \geq 5n-1$, then there is a vertex cover for G of size at most k.

Proof. Note that $n' := |V'| = k + 5n$ and $m' := |E'| = kn + \frac{n^2-n}{2} - m + \frac{4n(4n-1)}{2}$. Since $\bar{d}C(G') \geq 5n-1$, there is a minor G_1 with $\bar{d}(G_1) \geq 5n-1$, such that $G_1 = G'/E_1$. Furthermore, note that we can assume w.l.o.g. that G_1 is a contraction of G', i.e. there is a contraction-set E_1, because deleting edges does not increase the average degree, and instead of deleting vertices we can contract them to a neighbour.

In G', all vertices in $V \cup \{w_1, ..., w_{4n}\}$ have degree at least $4n$, while the vertices in $\{u_1, ..., u_k\}$ have degree exactly n. In a series of derivations, we will now show that even deleting vertices in $\{u_1, ..., u_k\}$ increases the average degree. Let G_2 be a minor of G' with at least $5n$ vertices, such that $\exists i \in \{1, ..., k\}$ with $u_i \in V(G_2)$, $n_2 = n(G_2)$ and $m_2 = m(G_2)$.

$$\begin{array}{rl} & 2 < n \\ \Rightarrow & n^2 < 2n^2 - 2n \\ \Rightarrow & n^2 < 8n^2 - 2n - n(6n) \\ \Rightarrow & n^2 < 8n^2 - 2n - k(k+5n-1) \\ \Rightarrow & nk + 4n^2 + n^2 < nk + 4n^2 + \frac{16n^2-4n}{2} - k(k+n+4n-1) \\ \Rightarrow & n \cdot n_2 \leq n(k+4n+n) < nk + 4n^2 + \frac{4n(4n-1)}{2} - k(k+n+4n-1) \leq m_2 \end{array}$$

Using just a few transformations, we see that $n_2 \cdot n < m_2$ is equivalent to $\frac{2m_2}{n_2} < \frac{2(m_2-n)}{n_2-1}$, and hence, we have:

$$\bar{d}(G_2) = \frac{2m_2}{n_2} < \frac{2(m_2-n)}{n_2-1} \leq \frac{2(m_2 - d_{G_2}(u_i))}{n_2-1} = \bar{d}(G_2 - u_i)$$

This means that the average degree in G_2 is smaller than the average degree in $G_2 - u_i$. We now examine the structure of G_1. G_1 has at least $5n$ vertices, otherwise it could not have average degree at least $5n-1$. If G_1 contains a vertex from $\{u_1, ..., u_k\}$, we can delete it, which increases the average degree of G_1. If the new G_1 still has at least $5n$ vertices, we can iterate this process, stepwise delete a vertex from $\{u_1, ..., u_k\}$ and increasing the average degree. At the end, we obtain a graph with at most $5n$ vertices and hence, with average degree at most $5n-1$. Therefore, we can conclude that G_1 must have at most $5n$ vertices, otherwise it contains a vertex in $\{u_1, ..., u_k\}$. To have average degree $5n-1$, G_1 must be a complete graph on $5n$ vertices. Hence, the contraction-set E_1 (with $G'/E_1 = G_1$) contains exactly k edges to contract all vertices in $\{u_1, ..., u_k\}$ to a neighbour in V to turn V into a clique. We define $V_1 := \bigcup_{e \in E_1} e \cap V$, and show that V_1 is a vertex cover for G. Clearly, $|V_1| \leq k$. Assume that there is an edge $e := \{x, y\} \in E$ with $e \cap V_1 = \emptyset$. Thus, $e \notin E'$. Since e is an edge in G_1, there was an edge $\{u_i, x\}$ or $\{u_i, y\}$ contracted to obtain G_1. Hence, this edge is in E_1, and therefore $x \in V_1$ or $y \in V_1$, which is a contradiction. $\diamond$

The transformation above is a polynomial time transformation. Furthermore, membership in NP of the considered problem is obvious. Hence, the lemma follows. $\square$

It is interesting to note that the above proof is also true when replacing the average degree $\bar{d}$ by the minimum degree δ, because it is easy to see that $\bar{d}(G') = \delta C(G')$, for G' as in the proof.

The origin of the upper bound, examined in this section, was the consideration of the following strategy. Successively contract an edge $\{x, y\}$ in a graph G, such that x and y have as least as possible common neighbours, and record the maximum $mad(G)$ of the average degree encountered during this process. It is clear that this strategy gives a lower bound on $\bar{d}C(G)$. Unfortunately, it is probably not an upper bound on the contraction degeneracy. If $mad(G)$ would also be an upper bound on the contraction degeneracy $\delta C(G)$, then we would have $\delta C(G) \leq mad(G) \leq \bar{d}C(G)$. In the previous proof, however, we constructed an instance G' with $\delta C(G') = mad(G') = \bar{d}C(G')$, and therefore, if $mad(G)$ would be an upper bound on the contraction degeneracy, then we could compute $\bar{d}C(G')$ or $\delta C(G')$ in polynomial time, solving Vertex Cover in polynomial time. Summarising, we can say that if $\delta C(G) \leq mad(G)$ for all graphs G, then $P = NP$.

7 A Bucket Adjacency List Data Structure

In this section, we describe a data structure that can be used in algorithms computing the degeneracy δD or similar parameters (e.g. in [4, 17]). It extends the graph adjacency-list data structure by a bucket structure (similar to bucket sort, sorted on vertex degree), enabling fast operations on this structure. It avoids a logarithmic factor that is typical for tree-structures (as used e.g. in [16]).

Description. Starting from the standard-adjacency-list of $G = (V, E)$, an advanced-adjacency-list is an extension that stores in a doubly linked list the adjacent vertices for every vertex of the graph. Furthermore, we use cross pointers for each edge $\{v_i, v_j\}$. Such a cross pointer connects vertex v_i in the list for vertex v_j directly with vertex v_j in the 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 (or pointers to them) with degree d in the current graph. We associate to every vertex v a pointer $p(v)$ that points to the exact position in the list B_d that contains v for the appropriate d. We also maintain a value $d(v)$ for every vertex v that simply is the degree of v.

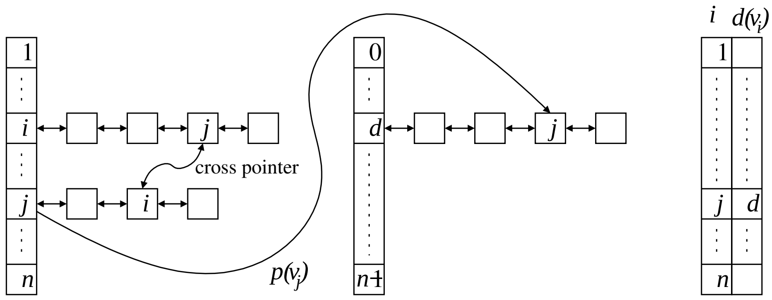

Fig. 3. Data structure with doubly-linked adjacency-lists and cross pointers; buckets as doubly-linked lists (in the middle of the figure); j in bucket d means $d(v_j) = d$ which can also be seen in the array on the right.

Cost of Operations. Here, we briefly describe how the graph operations that are interesting for us can be carried out on our data structure and how much running time they need.

Construction. Given a graph $G = (V, E)$ with $n = |V|$ and $m = |E|$ as a standard adjacency list, the advanced-adjacency-list structure described above can be constructed in time $O(n + m)$ by traversing the standard-adjacency-list and constructing doubly-linked lists and cross pointers while the traversal. Applying a bucket sort and building the pointers $p(v_i)$ for all $i = 1, ..., n$ can also be done in time $O(n + m)$. It is an easy task to compute the values $d(v_i)$ for all $i = 1, ..., n$ in the required time.

Degree Update. Let be given a vertex v_i. Changing the degree of v_i can be done in constant time in the following way. With $d(v_i)$, we know the bucket v_i is contained in. Using this and the change in degree, we can compute the new bucket v_i has to be inserted in. This insertion (at the start or end of the corresponding doubly linked list), the deletion in the old bucket-list (using pointer $p(v_i)$) and also updating $d(v_i)$ can all be done in constant time.

Vertex Deletion. Deleting a vertex v_i means deactivating it in the advanced-adjacency-list structure. For every vertex v_j in the adjacency list of v_i, we have to delete v_i in the adjacency list of v_j (using cross pointers), and we also have to update the degree of v_j. All can be done in constant time per neighbour v_j of v_i. Also, deleting v_i from its corresponding bucket list can be done in constant time using $p(v_i)$. Therefore, deleting vertex v_i takes time $O(d(v_i))$. Note that also $d(v_i)$ edges are deleted in the graph.

Edge Contraction. Contraction an edge $\{v_i, v_j\}$ can be done as follows. First, we traverse the list of neighbours of v_j and for each such neighbour v, we exchange v_j by v_i in the adjacency list of v (using cross pointers). We concatenate the adjacency list of v_j to the one of v_i, and we deactivate (i.e. delete) v_j. In order $O(d(v_i) + d(v_j))$, we can check this list for multiple occurrences and delete them and occurrences of v_i and v_j (this can be done using a temporary array counting the occurrences of a vertex in this list). On vertices occurring more than once a degree update has to be done. Furthermore, the degree of the new vertex (also called v_i) has to be updated. All can be done in $O(d(v_i) + d(v_j))$ time.

Find a vertex of smallest degree. Maintaining a pointer δB to the nonempty bucket B_i with smallest index i helps to find a vertex with minimum degree in constant time. Also this pointer may have to be updated when a vertex is deleted or an edge is contracted. Note that in either case the degree of any vertex can only decrease by one, and the number of vertices is also decreased by one. Hence, pointer δB can make at most n single steps to the left, i.e. from B_d to B_{d-1} (if the buckets are sorted $B_0, ..., B_{n-1}$). Thus this pointer can make at most $2 \cdot n$ single steps to the right, and therefore altogether it needs $O(n)$ time for a sequence of n operations (vertex deletions or edge contractions). Hence, we have amortised time $O(1)$ per operation.

Find a vertex of second smallest degree. To find a vertex with second smallest degree in constant time, we use a pointer $\delta_2 B$. Also this pointer makes amortised $O(1)$ single steps per operation, for a sequence of n operations.

Lemma 12. *Let be given a graph $G = (V, E)$ with $n = |V|$ and $m = |E|$. A sequence of $O(n)$ vertex deletions, searches for of a vertex with smallest or second smallest degree costs $O(n + m)$ time.*

Proof. Using the data structure described in this section, the result follows directly from the paragraphs considering the time of the corresponding operations. □

If we have a sequence of operations that involve edge-contractions, then the running time might be asymptotically higher, since repetitively concatenate adjacency lists and check for doubly occurring vertices takes $O(n)$ time.

Experimental Comparison. We implemented the data structure and compared the resulting running times to those obtained with a tree-based priority queue as used in e.g. [3, 4, 16]. The experimental setup is the same as in [3, 4, 17]. In Table 1, we see the results of the δD, $\delta_2 D$ and δC lower bounds algorithms and heuristics for a selection of input graphs. For more details on the algorithms and heuristics that are used here, see [3, 4, 17]. We also see the running times obtained with the tree-based priority queue (see columns 'tree' in Table 1) and the bucket-based priority queue (see columns 'bucket' in Table 1).

instance	size		δD				$\delta_2 D$				δC (least-c)			
			tree		bucket		tree		bucket		tree		bucket	
	$\|V\|$	$\|E\|$	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU
link	724	1738	4	0.01	4	0.01	4	5.22	4	3.15	11	0.04	11	0.04
munin1	189	366	4	0.00	4	0.00	4	0.29	4	0.19	10	0.01	10	0.01
munin3	1044	1745	3	0.01	3	0.01	3	11.02	3	5.65	7	0.03	7	0.02
pignet2	3032	7264	4	0.05	4	0.03	4	108.91	4	64.28	38	0.21	38	0.22
celar06	100	350	10	0.00	10	0.00	11	0.09	11	0.09	11	0.01	11	0.00
celar07pp	162	764	11	0.00	11	0.01	12	0.30	12	0.24	15	0.01	15	0.01
graph04	200	734	6	0.01	6	0.01	6	0.42	6	0.31	19	0.02	20	0.02
rl5934-pp	904	1800	3	0.01	3	0.01	3	9.18	3	5.13	5	0.04	5	0.04
school1	385	19095	73	0.03	73	0.04	74	7.61	74	10.15	122	0.84	122	0.94
school1-nsh	352	14612	61	0.02	61	0.03	62	5.55	62	7.14	106	0.62	106	0.70
zeroin.i.1	126	4100	48	0.01	48	0.01	48	0.58	48	0.76	50	0.05	50	0.05

Table 1. Comparison of running times of basic algorithms with two different data-structures: tree-based and bucket-based priority queues

The running times of the algorithm for computing δD are nearly equal, no matter which data structure is used. This might be explained by the fact that these running times are very small anyway. Hence, large absolute improvements are not possible and relative improvements are difficult to observe.

Similar behaviour can be observed in the columns representing the running times of the heuristic computing δC. Also these values are very small and a clear trend is difficult to be identified. It is interesting that a different lower bound value for 'graph04' is obtained with the two different data-structures. This is possible, because the considered algorithm is a heuristic and does not always compute the exact value. Therefore, different data-structures may lead to a different order in which the vertices are processed. This, on its turn, can lead to different results.

The running times of the algorithm computing $\delta_2 D$ are the largest, and here, we can see the following trend (that can also be observed in the other columns in a much smaller degree): If the graph is rather sparse (like the first 8 graphs in Table 1), then the bucket-based priority queue gives the faster algorithms. On the other hand, if the graph is rather dense (as the last 3 graphs in Table 1), then the tree-based priority queue is to be preferred.

Acknowledgements

We would like to thank Andreas Eisenblätter, Carsten Thomassen and Peter Lennartz for useful discussions.

References

1. M. Behzad, G. Chartrand, and L. Lesniak-Foster. *Graphs and Digraphs*. Pindle, Weber & Schmidt, Boston, 1979.
2. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comp. Sc.*, 209:1–45, 1998.
3. 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.
4. 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.
5. H. L. Bodlaender and T. Wolle. Contraction degeneracy on cographs. Technical Report UU-CS-2004-031, Institute for Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands, 2004.
6. K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *J. Comp. Syst. Sc.*, 13:335–379, 1976.
7. K. Cattell, M. J. Dinneen, R. G. Downey, M. R. Fellows, and M. A. Langston. On computing graph minor obstruction sets. *Theor. Comp. Sc.*, 233:107–127, 2000.
8. G. Demoucron, Y. Malgrange, and R. Pertuiset. Graphes planaires: reconnaissance et construction de representations planaires topologiques. *Rev. Francaise Recherche Operationelle*, 8:33–47, 1964.
9. R. Diestel. *Graph Theory*. Springer-Verlag, New York, 2000.
10. I. S. Filotti, G. L. Miller, and J. Reif. On determining the genus of a graph in $O(V^{O(g)})$ steps. In *Proceedings of the 11th Annual Symposium on Theory of Computing*, pages 27–37, New York, 1979. ACM Press.
11. V. Gogate and R. Dechter. A complete anytime algorithm for treewidth. In proceedings UAI'04, Uncertainty in Artificial Intelligence, 2004.
12. M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.
13. J. Gross and J. Yellen. *Graph Theory and Its Applications*. New York, CRC Press, 1999.
14. F. Harary. *Graph Theory*. Addison-Wesley, Reading, MA, 1969.
15. J. E. Hopcroft and R. E. Tarjan. Efficient planarity testing. *J. ACM*, 21:549–568, 1974.
16. 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.
17. 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.

18. S. K. Lando and A. K. Zvonkin. *Graphs on Surfaces and Their Applications*. Springer-Verlag, Heidelberg, 2004.
19. B. Mohar and C. Thomassen. *Graphs on Surfaces*. The Johns Hopkins University Press, Baltimore, 2001.
20. S. Ramachandramurthi. *Algorithms for VLSI Layout Based on Graph Width Metrics*. PhD thesis, Computer Science Department, University of Tennessee, Knoxville, Tennessee, USA, 1994.
21. P. Scheffler. *Die Baumweite von Graphen als ein Maß für die Kompliziertheit algorithmischer Probleme*. PhD thesis, Akademie der Wissenschaften der DDR, Berlin, 1989.
22. C. Thomassen. The graph genus problem is NP-complete. *J. Algorithms*, 10:568–576, 1989.
23. T. Wolle and H. L. Bodlaender. A note on edge contraction. Technical Report UU-CS-2004-028, Institute of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands, 2004.