Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

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

Degree-Based Treewidth Lower Bounds

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

ZIB-Report 04–44 (November 2004)

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@cs.uu.nl 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 [19] as well as the frequency assignment problem [18] in cellular wireless networks, it has been shown that such an algorithm to compute the optimal solution can be used in practice as well.

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. 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 [6, 7, 13], upper bounds [11, 10, 17], lower bounds [5, 8, 11, 20, 23], and exact algorithms (e.g. [15]).

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. [15]). The better the lower bounds, the bigger the branches that can be

* 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*).

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 [8, 9] 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 [20].

In this paper, we continue 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 [23]). 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). 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; i.e. taking the maximum of a treewidth lower bound over all minors of the graph is a very valid approach for improving this lower bound.

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. $N(S) = \bigcup_{s \in S} N(s) \setminus S$.

Edge Contraction. A more formal approach to edge contractions as well as basic lemmas can be found in [27]. 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 u and v, and deletes vertices u and v and all edges incident to u or v:

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

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). We explicitely exclude the null graph (the empty graph on 0 vertices), as a subgraph or minor of a graph.

Treewidth. The notions treewidth and tree decomposition were introduced by Robertson and Seymour in [24]. 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 $v, w \in X_i$,
- 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.

Lemma 1 (see e.g. [4]). *If G' is a minor of G, then $tw(G') \leq tw(G)$.*

This is a well known result and an important fact for proving the parameters, considered in this paper, to be treewidth lower bounds.

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 [3] 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 interesting that this definition reflects an algorithm to compute the degeneracy: Successively deleting a vertex of minimum degree and returning the maximum of the encountered minimum degrees. Furthermore, it is easy to see that this definition of the degeneracy is equivalent (see [28]) 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 [17]). The *δ-contraction degeneracy* or simply the *contraction degeneracy* δC of a graph G was first defined in [8]. 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 [8, 9]. 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 [22, 23], 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 $|V| \geq 2$, it holds:*

$$\delta(G) \leq \delta_2(G) \leq \gamma_R(G) \leq tw(G)$$

Proof. The first two inequalities follow directly from the definitions of the according parameters. The last one was proven by Ramachandramurthi in [22]. □

Lemma 3. *For a graph G and $x \in \{\delta, \delta_2, \gamma_R\}$, it holds:*

$$x(G) \leq xD(G) \leq xC(G) \leq tw(G)$$

Proof. Note that G is a subgraph of G, and any subgraph of G is also a minor of G. Therefore, the first two inequalities are easy to see. Furthermore, taking minors does not increase the treewidth (see Lemma 1). We also have that $x(G') \leq tw(G')$ (which follows from Lemma 2) for G' a minor of G. Hence, it follows that $x(G') \leq tw(G') \leq tw(G)$ for any minor G' of G, and therfore $xC(G) \leq tw(G)$. □

Lemma 4. *For a graph $G = (V, E)$ with $|V| \geq 2$ and $X \in \{D, C\}$, it holds:*

$$\delta X(G) \leq \delta_2 X(G) \leq \gamma_R X(G) \leq tw(G)$$

Proof. Lemma 2 holds for every subgraph or minor G' of G, unless G' has only one vertex. However, in that case the minimum degree is zero, and since G has at least two vertices it is obvious that $\delta_2 X(G) \geq 0$ and $\gamma_R X(G) \geq 0$. Therefore, the first two inequalities follow. Note that $\gamma_R X(G) \leq \gamma_R C(G) \leq tw(G)$ (see Lemma 3), and hence $\gamma_R X(G) \leq tw(G)$. □

Lemma 5. *For a graph $G = (V, E)$ with $|V| \geq 2$ and $X \in \{D, C\}$, it holds:*

$$\delta_2 X(G) \leq \delta X(G) + 1$$

Proof. Let $G' = (V', E')$ be a subgraph (if $X = D$) or minor (if $X = C$) of G with $\delta_2(G') = \delta_2 X(G)$, and let v_1 and v_2 be vertices in G' with smallest and second smallest degree in G', respectively, i.e. $d_{G'}(v_1) = \delta(G')$ and $d_{G'}(v_2) = \delta_2(G')$. We consider the graph $G'' := G'[V' \setminus \{v_1\}]$. Note that G'' is also a subgraph ($X = D$) or minor ($X = C$) of G. It is clear that we have:

$$\forall v \in V(G'') \ : \ d_{G'}(v) - 1 \leq d_{G''}(v) \leq d_{G'}(v)$$

Let $w \in V(G'')$ be a vertex with minimum degree in G'', i.e. $\delta(G'') = d_{G''}(w)$. Due to the definition of v_2, it holds that:

$$\delta_2 X(G) = d_{G'}(v_2) \leq d_{G'}(w)$$

Otherwise v_2 was not a vertex of second smallest degree in G'. Altogether, we have:

$$\delta_2 X(G) - 1 = d_{G'}(v_2) - 1 \leq d_{G'}(w) - 1 \leq d_{G''}(w) = \delta(G'') \leq \delta X(G)$$

□

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

Lemma 6. *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*

Proof. (1.) Since $d(v_l) \leq d(v_j)$ and $\{v_i, v_l\} \notin E(G)$, we clearly have $\max\{d(v_l), d(v_j)\} = d(v_j) \geq \gamma_R(G)$, and because there is no $v_{j'}$ with $d(v_{j'}) \leq d(v_j)$ and $j' < j$, such that there is a $v_{l'} \in \{v_1, ..., v_{j'} - 1\}$, with $\{v_{l'}, v_{j'}\} \notin E(G)$, the equality follows.

(2.) This follows because if v_j is the left most vertex in the given sequence that is not adjacent to all the vertices $v_1, ..., v_{j-1}$ to its left, then the vertices $v_1, ..., v_{j-1}$ must form a clique. □

Lemma 7. *For a graph $G = (V, E)$ with $|V| \geq 2$ and $X \in \{D, C\}$, it holds:*

$$\gamma_R X(G) \leq 2 \cdot \delta_2 X(G)$$

Proof. Let $G' = (V', E')$ be a minimal subgraph (in case $X = D$) or a minimal minor (in case $X = C$) of G with $\gamma_R(G') = \gamma_R X(G)$, i.e. there is no subgraph or minor G^* of G' with $\gamma_R(G^*) \geq \gamma_R(G')$. If G' is a complete graph on $n' := |V'|$ vertices, then the lemma follows easily. For technical reasons, we assume in the following w.l.o.g. that $G' \neq K_{n'}$. Let be given an ordering $v_1, ..., v_{n'}$ of V', such that $d_{G'}(v_i) \leq d_{G'}(v_{i+1})$, for all $i \in \{1, ..., n'-1\}$. We define $j := \min\{i \in \{1, ..., n'\} \mid \exists l \in \{1, ..., i-1\} : \{v_i, v_l\} \notin E'\}$. From Lemma 6, we know that $d_{G'}(v_j) = \gamma_R(G')$ and that $v_1, ..., v_{j-1}$ form a clique in G'. Thus, these vertices have degree at least $j-2$. To show the lemma, we need a slightly higher degree as stated by the following claim.

Claim. $\delta_2(G') = d_{G'}(v_2) \geq j - 1$.

Proof. Assume the opposite, namely that $d_{G'}(v_1) = d_{G'}(v_2) = j - 2$. Hence, v_1 and v_2 are only adjacent to the vertices in the clique formed by $v_1, ..., v_{j-1}$, and therefore, both v_1 and v_2 are not adjacent to v_j. Note that $\max\{d_{G'}(v_1), d_{G'}(v_j)\} = \max\{d_{G'}(v_2), d_{G'}(v_j)\} = \gamma_R(G')$. Now, we consider $G'[V' \setminus \{v_1\}]$. The deletion of v_1 decreases the degree of the vertices $v_2, ..., v_{j-1}$ by one. However, $v_2, ..., v_{j-1}$ still form a clique in $G'[V' \setminus \{v_1\}]$. Furthermore, note that due to the deletion of v_1, we only deleted non-edges $\{v_1, v_i\}$, i.e. pairs of vertices $\{v_1, v_i\} \notin E'$. Deleting elements of a set over which a minimum is taken can never decrease the value of that minimum. Therefore, we can conclude that $\{v_2, v_j\}$ is a non-edge in $G'[V' \setminus \{v_1\}]$, determining $\gamma_R(G'[V' \setminus \{v_1\}])$. Since the degree of v_j did not change when deleting v_1, we have that $\max\{d_{G'[V' \setminus \{v_1\}]}(v_2), d_{G'[V' \setminus \{v_1\}]}(v_j)\} = \gamma_R(G'[V' \setminus \{v_1\}]) = \max\{d_{G'}(v_2), d_{G'}(v_j)\} = \gamma_R(G')$. Hence, $\gamma_R(G'[V' \setminus \{v_1\}]) \geq \gamma_R(G')$, which contradicts the choice of G'. ◇

Hence, we have that $j - 1 \leq d_{G'}(v_j) = \gamma_R X(G)$. Note that the following holds:

$$\delta_2 X(G) \geq \delta_2 X(G') \geq \delta_2(G') = d_{G'}(v_2) \geq j - 1$$

We consider now the graph $G'' := G'[V' \setminus \{v_1, ..., v_{j-1}\}]$, which is also a subgraph ($X = D$) or minor ($X = C$) of G. It is clear that deleting $j-1$ vertices in a graph can decrease the degree of any vertex at most by $j - 1$. Therefore, we have:

$$\delta_2 X(G) \geq \delta X(G) \geq \delta(G'') \geq d_{G'}(v_j) - (j-1)$$

Hence, altogether, we have:

$$\begin{aligned} \delta_2 X(G) &\geq \max(j - 1, d_{G'}(v_j) - (j-1)) \\ &\geq \frac{d_{G'}(v_j)}{2} = \frac{\gamma_R(G')}{2} = \frac{\gamma_R X(G)}{2} \end{aligned}$$

□

It follows directly from Lemma 2, Lemma 3 and Lemma 4 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 5). In Section 3.2, we will see that $\delta_2 D$ can be computed in polynomial time. Therefore, Lemma 7 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.

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 [28]. 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 8 (see [28]). *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 [22] or Section 4.1. The definition of the degeneracy as in [3] 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 [8].

$\delta_2 D$ Is Computable in Polynomial Time.

First, we note that $\delta_2 D$ cannot be computed using the strategy of the algorithm for computing the degeneracy δD of a graph G. This becomes evident when considering the example graph in Figure 1. There, we see that successively deleting a vertex of smallest degree (in the example in

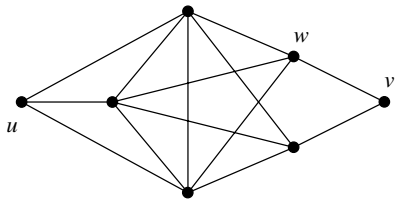

Fig. 1. Graph G as an example that deleting vertices of smallest degree does not lead to a correct algorithm for computing $\delta_2 D$

Figure 1 the vertex of smallest degree is v) does not lead to a subgraph where δ_2 is maximal, i.e. $\delta_2 D(G \setminus \{v\}) = 3$. Instead, we must delete vertex u (a vertex with second smallest degree), and we

obtain $\delta_2 D(G \setminus \{u\}) = \delta_2 D(G) = 4$. This gives the clue that deleting a vertex with second smallest degree might be a correct approach, but it is easy to find a counter-example for this strategy as well.

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 can be implemented to take $O(n \cdot m)$ time. 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 be 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 9. *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* $|V| \geq 2$.

Proof. First, we will show that $delta2D = \delta_2 D(G)$. Note that every H considered in the algorithm is a subgraph of G. Therefore, it is easy to see that $delta2D \leq \delta_2 D(G)$, since:

$$delta2D = \max_H \{\delta_2(H) \mid H \text{ occurs during the run of the algorithm}\}$$

Now, we show that there is a subgraph $H \subseteq G$ considered during the algorithm with $\delta_2(H) = \delta_2 D(G)$. Let be given $G' = (V', E') \subseteq G$ with $\delta_2(G') = \delta_2 D(G)$. Furthermore, let $v \in V'$ be a vertex of minimum degree in G'. We consider the run of the for-loop of algorithm `Delta2D` where v was chosen (in Line 2) to always remain in the graph. Note that the algorithm selects and deletes successively a vertex $u \neq v$ whose degree is as small as possible. Now, consider the first time when in the repeat-loop, i.e. in the current graph H, a vertex $u \in V'$ is selected to be deleted. Because u is the first such vertex, we have $G' \subseteq H$. Therefore, for all $w \in V' \setminus \{v\}$, we have $\delta_2(G') \leq d_{G'}(w) \leq d_H(w)$. Hence, since $u \in V' \setminus \{v\}$, it holds that $\delta_2(G') \leq d_H(u)$. Because u is a vertex in $V^* = V(H) \setminus \{v\}$ with degree in H as small as possible, all vertices in V^* have degree at least $d_H(u) \geq \delta_2(G')$. Therefore, we have $\delta_2(H) \geq \delta_2(G') = \delta_2 D(G)$. Hence, the algorithm considers a graph $H \subseteq G$ with $\delta_2(H) = \delta_2 D(G)$. This proves our initial claim $delta2D = \delta_2 D(G)$.

If we use the data structure as described at the beginning of this section, it is not difficult to see the claimed running time of the algorithm. □

$\gamma_R D$ Is Hard to Compute.

In this section, we formulate the decision problem corresponding to the parameter $\gamma_R D$, and we show its NP-completeness. We will give a proof, that is similar to the proof of the NP-completeness of Contraction Degeneracy in [8]. However, before that, we show that considering only induced subgraphs when computing $\gamma_R D$ is sufficient.

Lemma 10. *For all graphs G, there exists an induced subgraph G' of G such that $\gamma_R(G') = \gamma_R D(G)$.*

Proof. Let be given a subgraph $G' = (V', E')$ of G with $\gamma_R(G') = \gamma_R D(G)$. Let be $e \in E(G[V']) \setminus E'$. If no such edge exists, then G' is an induced subgraph. Adding edge e to G' has two effects. First, note that the degree of two vertices is increased by one. We have that $\forall v \in V' : d_{G'}(v) \leq d_{(V', E' \cup \{e\})}(v) \leq d_{G'}(v) + 1$. Furthermore, note that adding an edge deletes one of the non-edges $\{v_i, v_j\}$ with $v_i \neq v_j; v_i, v_j \in V'$, over which the minimum of $\max(d_{G'}(v_i), d_{G'}(v_j))$ is taken. However, deleting an element of a set over which a minimum is taken can never decrease the value of that minimum. Therefore, we have $\gamma_R(G') \leq \gamma_R((V', E' \cup \{e\}))$. The lemma is shown by applying this argumentation successively until an induced subgraph is obtained. □

Now, we define the decision problem for the γ_R-degeneracy and prove its NP-completeness.

Problem: γ_R-Degeneracy
Instance: Graph $G = (V, E)$ with $|V| \geq 2$ and integer $k \geq 0$.
Question: Is $\gamma_R D(G) \geq k$?

Theorem 1. *The problem γ_R-Degeneracy is NP-complete.*

Proof. Membership in NP is easy to see, since we only have to guess a subgraph and then compute γ_R of that subgraph in polynomial time. The hardness proof is a transformation of the known NP-complete problem Vertex Cover, see [14]. In the Vertex Cover problem, we are given a graph $G = (V, E)$ and an integer l, and look for a vertex cover of size at most l, i.e. a set $W \subseteq V$ with $|W| \leq l$, such that each edge in E has at least one endpoint in W.

Let be given a Vertex Cover instance (G, l), with $G = (V, E)$ and $V = \{v_1, ..., v_n\}$. We assume that $1 \leq l \leq n - 1$, which is not a restriction, since $l = 0$ and $l = n$ are trivial instances for Vertex Cover. We will construct a γ_R-Degeneracy instance.

Construction: We take a clique with vertex set $U = \{u_1, ..., u_{n+l}\}$, an independent set $W = \{w_1, ..., w_{n+l-1}\}$, and we take the complement $\bar{G}$ of G. We add all edges between vertices in U and W, and all edges between vertices in W and V. The resulting graph G' (see Figure 2) is formally

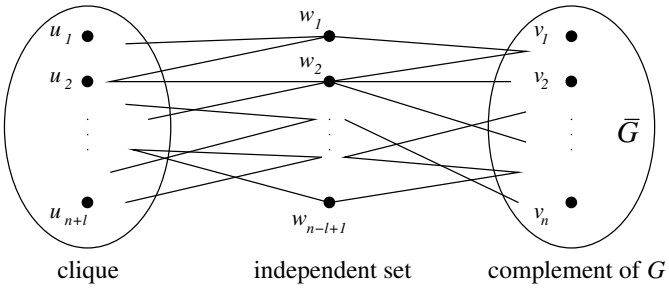

Fig. 2. Graph G' constructed in the proof of Theorem 1

defined as follows:

$$
\begin{aligned}
G' &:= (V', E') \text{ where} \\
V' &= U \cup W \cup V \\
E' &= \{\ \{u_i, u_j\} \mid u_i \neq u_j \wedge u_i, u_j \in U\ \} \\
&\quad \cup \{\ \{u, w\} \mid u \in U \wedge w \in W\ \} \\
&\quad \cup \{\ \{w, v\} \mid w \in W \wedge v \in V\ \} \\
&\quad \cup \{\ \{v_i, v_j\} \notin E \mid v_i, v_j \in V, v_i \neq v_j\ \}
\end{aligned}
$$

Our constructed γ_R-Degeneracy instance is $(G', 2 \cdot n)$. Now, we show that there is a vertex cover for G of size at most k if, and only if $\gamma_R D(G') \geq 2 \cdot n$.

Claim. If there is a vertex cover V_1 of G of size at most l, then $\gamma_R D(G') = 2 \cdot n$, i.e. then there is a subgraph G'' of G' with $\gamma_R(G'') = 2 \cdot n$.

Proof. Assume V_1 is a vertex cover of G with $|V_1| = l$. Note that $\bar{G}[V \setminus V_1]$ is a clique of size $n - l$, since a non-edge $\{v_i, v_j\}$ in $\bar{G}[V \setminus V_1]$ $(v_i \neq v_j)$, would be an edge in G and therefore, $v_i \in V_1$ or $v_j \in V_1$. We consider $G'' := G'[V' \setminus V_1]$. Since the remaining vertices of G form a clique in G'', the only non-edges are $\{w_i, w_j\}$ with $i \neq j; w_i, w_j \in W$, and $\{u_i, v_j\}$ with $u_i \in U; v_j \in V$. Furthermore, note that $\max(d_{G''}(w_i), d_{G''}(w_j)) = 2 \cdot n$, and $\max(d_{G''}(u_i), d_{G''}(v_j)) = 2 \cdot n$. Therefore, $\gamma_R(G'') = 2 \cdot n$. ◇

Claim. If $\gamma_R D(G') \geq 2 \cdot n$, i.e. if there is a subgraph G'' of G' with $\gamma_R(G'') \geq 2 \cdot n$, then there is a vertex cover V_1 of G of size at most l.

Proof. Let be given $G'' = (V'', E'')$ as an induced subgraph of G' (see Lemma 10), such that $\gamma_R D(G') = \gamma_R(G'') \geq 2 \cdot n$. Note that the only non-edges are $\{w_i, w_j\}$ with $i \neq j; w_i, w_j \in W$, $\{u_i, v_j\}$ with $u_i \in U; v_j \in V$, and $\{v_i, v_j\} \in E(G)$. The degree in G' (and also in G'') of a vertex in V is at most $(n-1) + (n-l+1) = 2 \cdot n - l < 2 \cdot n$. Hence, all pairs of vertices in V remaining in G'' are joined by an edge. Therefore, $V'' \cap V$ is a (perhaps empty) clique in G''.

Fact: $V'' \cap V$ is a clique of size at least $n - l$. This can be seen by assuming $|V'' \cap V| < n - l$. Note that every vertex in W has degree in G'' at most $(n + l) + |V'' \cap V| < 2 \cdot n$. If there are at least two vertices in W, then we have $\gamma_R(G'') < 2 \cdot n$. On the other hand, if there is at most one vertex of W left in G'', then every vertex in $V'' \cap U$ has degree at most $n + l < 2 \cdot n$. If $|V'' \cap V| = 0$, then there are at most $n + l + 1 \leq 2 \cdot n$ vertices left in G'', rendering $\gamma_R(G'') = 2 \cdot n$ impossible. If $|V'' \cap V| > 0$, then the only non-edges are $\{u_i, v_j\}$ with $u_i \in U, v_j \in V$. But then we have $\max(d_{G''}(u_i), d_{G''}(v_j)) < 2 \cdot n$. Hence, $V'' \cap V$ is a clique of size at least $n - l$.

Now we define $V_1 := V \setminus (V'' \cap V)$, i.e. V_1 contains exactly those vertices of V that are not present in G''. We will show that V_1 is a vertex cover of G of size at most l. We clearly have that $|V_1| \leq l$. Assume there is an edge $\{v_i, v_j\} \in E$ with $\{v_i, v_j\} \cap V_1 = \emptyset$. Then $\{v_i, v_j\}$ is a non-edge in G', and by definition of V_1, it is also a non-edge in G''. This is a contradiction to the fact that all pairs of vertices in V remaining in G'' are joined by an edge. Hence, V_1 is a vertex cover of G. ◇

Since the construction described above is a polynomial time construction, the NP-completeness of γ_R-Degeneracy follows. □

$\gamma_R C$ Is Hard to Compute

We formulate the decision problem corresponding to $\gamma_R C$ and prove its NP-completeness. The proof is very similar to the proof of the NP-completeness of Contraction Degeneracy in [8].

Problem: γ_R-Contraction Degeneracy
Instance: Graph $G = (V, E)$ with $|V| \geq 2$ and integer $k \geq 0$.
Question: Is $\gamma_R C(G) \geq k$?

Theorem 2. *The problem γ_R-Contraction Degeneracy is NP-complete.*

Proof. It is easy to see that the problem belongs to NP, since we only have to guess a minor and then compute γ_R of that minor in polynomial time. We prove the hardness by transforming the known NP-complete problem Vertex Cover, see [14]. Let be given a Vertex Cover instance (G, l), with $G = (V, E)$, $V = \{v_1, ..., v_n\}$ and $n \geq 2$ (note that $n < 2$ implies a trivial Vertex Cover instance). From this instance, we will construct a γ_R-Contraction Degeneracy instance.

Construction: We take l vertices $u_1, ..., u_l$, the complement $\bar{G}$ of G, a C_4 which is a cycle with vertices $w_1, ..., w_4$, and we take a vertex x. We add all edges between vertices u_i and v_j, between v_j and w_p and we connect vertex x to all vertices w_p, for $i \in \{1, ..., l\}$, $j \in \{1, ..., n\}$, $p \in \{1, ..., 4\}$. We call the resulting graph G', formally defined as follows (see Figure 3):

$$
\begin{aligned}
G' &:= (V', E') \text{ where} \\
V' &= \{u_1, \ldots, u_l\} \cup V \cup \{w_1, \ldots, w_4\} \cup \{x\} \\
E' &= \{\ \{u, v\} \mid u \in \{u_1, \ldots u_l\} \wedge v \in V\ \} \\
&\quad \cup \{\ \{v_i, v_j\} \notin E \mid v_i, v_j \in V, v_i \neq v_j\ \} \\
&\quad \cup \{\ \{v, w\} \mid v \in V \wedge w \in \{w_1, \ldots, w_4\}\ \} \\
&\quad \cup \{\ \{w_1, w_2\}, \{w_2, w_3\}, \{w_3, w_4\}, \{w_4, w_1\}\ \} \\
&\quad \cup \{\ \{w, x\} \mid w \in \{w_1, \ldots, w_4\}\ \}
\end{aligned}
$$

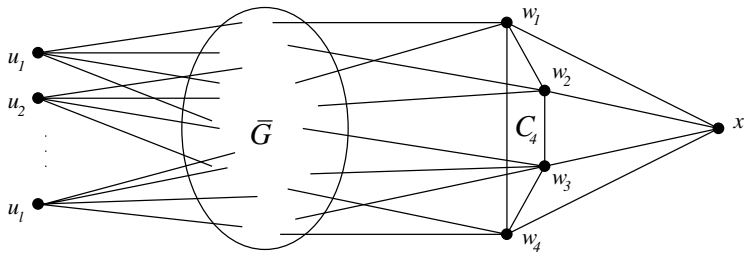

Fig. 3. Graph G' constructed in the proof of Theorem 2

The constructed instance of the γ_R-Contraction Degeneracy problem is $(G', n+3)$. So, we have to show that there is a vertex cover for G of size at most l if, and only if $\gamma_R C(G') \geq n+3$.

Claim. If there is a vertex cover of G of size at most l, then there is a $E^* \subseteq E'$, such that $\gamma_R(G'/E^*) \geq n+3$.

Proof. Suppose there is a vertex cover of size at most l. Now, take a vertex cover $V_1 = \{y_1, \ldots, y_l\}$ of G of size exactly l. (If we add vertices to a vertex cover, we obtain again a vertex cover.) In G', we contract edge set E^* defined as follows: $E^* := \{\ \{u_i, y_i\} \mid i = 1, \ldots, l\}$; i.e. each vertex y_i in the vertex cover V_1 has the vertex u_i contracted to it. We claim that for the resulting graph G'/E^* holds: $\gamma_R(G'/E^*) \geq n+3$. Therefore, we claim that all vertices in G'/E^*, apart from x, have degree $n+3$. Note that each vertex w in $\{w_1, ..., w_4\}$ is adjacent to vertex x, to all vertices in V and to exactly two other vertices from $\{w_1, ..., w_4\}$. Hence, all vertices in $\{w_1, ..., w_4\}$ are adjacent to $n+3$ vertices. Now, we claim that the vertices of V form a clique in G'/E^*. Assume there are two vertices z_1 and z_2 in V with $z_1 \neq z_2$ and $\{z_1, z_2\} \notin E(G'/E^*)$. Therefore, $\{z_1, z_2\} \in E$, and hence z_1 or z_2 has to be in V_1. We assume w.l.o.g. $z_1 \in V_1$, i.e. $\exists i \in \{1, ..., l\}$, such that $y_i = z_1$. Since we contracted $\{u_i, y_i\}$ and $\{u_i, z_2\} \in E(G')$, we have $y_i = z_1$ is adjacent to z_2 in G'/E^*, which is a contradiction. Hence, V forms an n-clique and therefore, the degree in G'/E^* of a vertex in V is $n+3$ (since every vertex in V is also adjacent to $w_1, ..., w_4$). Thus, $\gamma_R(G'/E^*) = n+3$, since x is not adjacent to a vertex in V. ◇

Claim. If there is an $E^* \subseteq E'$, such that $\gamma_R(G'/E^*) \geq n+3$, then there is a vertex cover $V_1 \subseteq V$ of G of size at most l.

Proof. Assume, there is a $E^* \subseteq E$, such that $\gamma_R(G'/E^*) \geq n+3$. First, we state some observations about E^* and the structure of G'/E^*. After that, we construct a set $V_1 \subseteq V$ and show that this is a vertex cover of G.

Fact: $l \leq |E^*| \leq l+1$. This is because, if $|E^*| < l$, then there are at least two vertices in $\{u_1, ...u_l, x\}$ left in G'/E^*, and hence, $\gamma_R(G'/E^*) \leq n$. On the other hand, if $l+1 < |E^*|$, then G'/E^* has at most $n+3$ vertices, and thus, $\gamma_R(G'/E^*) \leq n+2$.

Fact: $|E^*| \neq l+1$. To see this, we assume $|E^*| = l+1$. From the previous fact, we already know that at least l vertices from $\{u_1, ...u_l, x\}$ have to be contracted to a neighbour. Depending on which of these vertices were contracted, we distinguish two cases to show $|E^*| \neq l+1$.
Case '$u_1, ..., u_l$ were contracted, and one more edge e was contracted in G'/E^*.' If $e \in V \times V$ then the remaining vertices in V have degree at most $n+2$, and hence, $\gamma_R(G'/E^*) \leq n+2$. If $e \in V \times \{w_1, ..., w_4\}$, then one vertex in V is adjacent to x, and all other vertices in V have degree at most $n+2$; hence, $\gamma_R(G'/E^*) \leq n+2$. If $e \in \{w_1, ..., w_4\} \times \{w_1, ..., w_4\}$ then the vertices in V have degree at most $n+2$, and therefore, $\gamma_R(G'/E^*) \leq n+2$. If $e \in \{x\} \times \{w_1, ..., w_4\}$ then two vertices in $\{w_1, ..., w_4\}$ will have degree at most $n+2$ and are not adjacent; thus $\gamma_R(G'/E^*) \leq n+2$.
Case '$u_1, ..., u_{i-1}, u_{i+1}, ..., u_l$ were contracted, x was contracted w.l.o.g. to w_1 and one more edge e was contracted.' If $e \in \{u_i\} \times V$ then this case was already considered above. If $e \in V \times V$ then w_2 and w_4 have degree at most $n+1$, and hence, $\gamma_R(G'/E^*) \leq n+1$. If $e \in V \times \{w_1, ..., w_4\}$ then the vertices in $\{w_1, ..., w_4\} \setminus e$ have degree at most $n+2$; thus $\gamma_R(G'/E^*) \leq n+2$. If $e \in \{w_1, ..., w_4\} \times \{w_1, ..., w_4\}$ then the remaining vertices in $\{w_1, ..., w_4\}$ have degree at most $n+2$; therefore, $\gamma_R(G'/E^*) \leq n+2$.
In all cases and subcases, we concluded $\gamma_R(G'/E^*) \leq n+2$, which contradicts our initial assumption that $\gamma_R(G'/E^*) \geq n+3$. Hence, $|E^*| = l+1$ is not possible.

Fact: All vertices in $\{u_1, ..., u_l\}$ were contracted, and x was not contracted. We know that exactly l vertices in $\{u_1, ..., u_l, x\}$ were contracted. If a $u_i \in \{u_1, ..., u_l\}$ was not contracted, then x was contracted w.l.o.g. to w_1. Thus, w_2 and w_4 have degree at most $n+2$ and are non-adjacent; we conclude $\gamma_R(G'/E^*) \leq n+2$, which is a contradiction. Hence, after all the considerations above, the only possibility is that $u_1, ..., u_l$ were contracted to a neighbour, and x was not contracted.

Fact: The vertices of V form a clique in G'/E^*. Since x is not adjacent to any vertex in V, all vertices in V must have degree $n+3$. This is only possible if V forms a clique.

Now, we know that G'/E^* has the following structure. We have a clique on n vertices, all of which are adjacent to the vertices of a C_4. Furthermore, we have a vertex x adjacent to all vertices of the C_4. Hence, E^* contains exactly l edges, one for each $u_i, i \in \{1, ..., l\}$, with the other endpoint in V. We will now define $V_1 \subseteq V$ and show that this is a vertex cover of G. Let be $V_1 := \bigcup_{e \in E^*} e \setminus \bigcup_{i=1,...l} u_i$. Clearly, $|V_1| = l$, and we claim that V_1 is a vertex cover of G. Assume, there is an edge $f = \{z_1, z_2\}$ in G with $V_1 \cap f = \emptyset$. Hence, f is not an edge in $\bar{G}$ and also not in G'. Since V forms an n-clique in G'/E^*, edge f exists in G'/E^*, which means: f was created by contracting another edge $\{u_i, v_j\} \in E^*$, since u_i is adjacent to all vertices in V. This can only be the case if $v_j = z_1$ or $v_j = z_2$. According to the definition of V_1, we have: $v_j \in V_1$, which contradicts $V_1 \cap f = \emptyset$. Hence, V_1 is a vertex cover of size l. ◇

As G' can be constructed in polynomial time, the NP-completeness of the γ_R-Contraction Degeneracy problem now follows. □

Theorem 3. *It is NP-complete to decide whether $\gamma_R C(G) \geq k$ for a given bipartite graph G and an integer k.*

Proof. Already in [9], we observed that $\delta C(G) \geq 3 \Leftrightarrow tw(G) \geq 3$. Since $\delta C(G) \leq \gamma_R C(G) \leq tw(G)$, we also have $\gamma_R C(G) \geq 3 \Leftrightarrow tw(G) \geq 3$. It is known that graphs of treewidth at most two can be recognised in polynomial time, e.g. with preprocessing rules (see [1, 2, 4, 7]) or with Robertson and Seymour graph minor theory (see e.g. [12, 21]) as used in the proofs of the fixed parameter results in [9]. Therfore, it is no restriction that we assume for technical reasons that $\gamma_R C(G) \geq 3$ (and hence, also $k \geq 3$).

The membership in NP is obvious. We use a polynomial transformation from γ_R-Contraction Degeneracy on general graphs, which is known to be NP-complete (see Theorem 2). Let be given an instance (G, k) of the γ_R-Contraction Degeneracy problem. We subdivide any edge in G, i.e. we place a new vertex on each edge, and obtain G', formally defined in the following way:

$$G' := (V', E') \text{ where}$$

$$V' = V \cup \{\ s_e \mid e \in E\ \}$$
$$E' = \{\ \{a, s_e\}, \{s_e, b\} \mid e = \{a, b\} \in E\ \}$$

The new constructed instance is (G', k). G' is a bipartite graph, as all edges in E' are between a vertex in V and a vertex in $\{\ s_e \mid e \in E\ \}$. Now, we have to show that $\gamma_R C(G) \geq k$ if, and only if $\gamma_R C(G') \geq k$.

Claim. If $\gamma_R C(G) \geq k$ then $\gamma_R C(G') \geq k$.

Proof. Note that G is a minor of G', since $G = G'/E^*$, where E^* is set of edges to undo the subdivisions. Therefore, any minor of G is also a minor of G'. Hence, the claim follows. ◇

Claim. If $\gamma_R C(G') \geq k$ then $\gamma_R C(G) \geq k$.

Proof. Let $G_1 \preceq G'$ be a minor of G such that $\gamma_R(G_1) \geq k$. If all vertices in $\{\ s_e \mid e \in E\ \}$ were contracted in G_1 to a neighbour, then G_1 is also a minor of G, and hence $\gamma_R C(G) \geq k$. Furthermore, if G_1 contains two vertices of $\{\ s_e \mid e \in E\ \}$, then $\gamma_R(G_1) = 2$, since all vertices in $\{\ s_e \mid e \in E\ \}$ are pairwise nonadjacent and have degree two. Therefore, we have to consider the case that exactly one vertex $s \in \{\ s_e \mid e \in E\ \}$ is present in G_1. This vertex s subdivides an edge $\{u, v\} \in E$. We distinguish the following cases.
Case 1: $\{u, v\} \notin E(G_1)$. Contracting edge $\{s, u\}$ does not change the degree of any vertex in $V(G_1)$ (apart from s). However, s cannot be the vertex determining $\gamma_R(G_1)$, since $d_{G_1}(s) = 2$. This contraction decreases the number of pairs over which the minimum is taken when computing $\gamma_R(G_1)$. Such a decrease can only increase the value of that minimum. Therefore, $k \leq \gamma_R(G_1) \leq \gamma_R(G_1/\{s, u\}) \leq \gamma_R C(G)$, since $G_1/\{s, u\} \preceq G$.
Case 2: $\{u, v\} \in E(G_1)$. In this case u, v and s form a triangle. We define $G_2 := G_1 - s = G_1/\{s, u\} = G_1/\{s, v\}$. Let $\{x, y\}$ be a nonedge in G_2 that determines $\gamma_R(G_2)$, with $d_{G_2}(x) \leq d_{G_2}(y)$. Since $\{u, v\} \in E(G_2)$, $\{x, y\} \neq \{u, v\}$. If $\{x, y\} \cap \{u, v\} = \emptyset$, then $d_{G_2}(x) = d_{G_1}(x)$ and $d_{G_2}(y) = d_{G_1}(y)$. Hence, we have that $\gamma_R(G_1) \leq \max(d_{G_1}(x), d_{G_1}(y)) = \max(d_{G_2}(x), d_{G_2}(y)) = \gamma_R(G_2)$. We consider the case where $\{x, y\} \cap \{u, v\} \neq \emptyset$ in two subcases.
Subcase 1: $x \in \{u, v\}$. We have $d_{G_1}(x) \geq d_{G_2}(x)$ and $d_{G_1}(y) = d_{G_2}(y)$, and $\{s, y\}$ is a nonedge in G_1. Therefore, we have that $\gamma_R(G_1) \leq \max(d_{G_1}(s), d_{G_1}(y)) \leq \max(d_{G_2}(x), d_{G_2}(y)) = \gamma_R(G_2)$.
Subcase 2: $y \in \{u, v\}$. In this case, we have that $d_{G_1}(x) = d_{G_2}(x)$ and $d_{G_1}(y) \geq d_{G_2}(y)$, and $\{s, x\}$ is a nonedge in G_1. Hence, $\gamma_R(G_1) \leq \max(d_{G_1}(s), d_{G_1}(x)) \leq \max(d_{G_2}(x), d_{G_2}(y)) = \gamma_R(G_2)$.
Hence, we can conclude that $\gamma_R(G_1) \leq \gamma_R(G_2)$. Furthermore, note that G_2 is also a minor of G, and thus, we have that $k \leq \gamma_R(G_1) \leq \gamma_R(G_2) \leq \gamma_R C(G)$. ◇

Because the transformation described above is a polynomial one, the theorem follows. □

$\delta_2 C$ Is Hard to Compute

We formulate the decision problem corresponding to $\delta_2 C$ and state its NP-completeness.

Problem: δ_2-Contraction Degeneracy
Instance: Graph $G = (V, E)$ with $|V| \geq 2$ and integer $k \geq 0$.
Question: Is $\delta_2 C(G) \geq k$?

Theorem 4. *The problem* δ_2-Contraction Degeneracy *is* NP*-complete.*

To prove this, we can use an easier variant of the proof for Theorem 2. When computing γ_R, we consider non-adjacent vertices. However, we can observe that the proof of Theorem 2 also holds when computing δ_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 both parameters can be exactly computed in linear time. The parameters δD and $\delta_2 D$ were computed as described in Section 3.2. Ramachandramurthi shows in [22] 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. Our algorithm appears to be simpler and is easy to implement. Let be given a sequence $v_1, ..., v_n$ of the vertices of the graph, such that $d(v_i) \leq d(v_{i+1}), \forall i \in \{1, ..., n-1\}$. Our algorithm as well as the algorithm in [22] are based on the fact that γ_R is determined by the leftmost vertex in this sequence that is not adjacent to all vertices to the left of it (see Lemma 6). As an invariant in our algorithm we have that counter $c(v_i)$ counts the number of neighbours of v_i to the left of v_i. Therefore, it is easy to find the first vertex with $i \neq c(v_i) + 1$.

```
Algorithm GammaR

 1   obtain sequence v_1, ..., v_n by bucket sorting
     the vertices according to nondecreasing degree
 2   initialise counter c(v_j) := 0, ∀j ∈ {1, ..., n}
 3   i := 1
 4   while i = c(v_i) + 1 and i < n do
 5       for all u ∈ N(v_i) do
 6           c(u) := c(u) + 1
 7       endfor
 8       i := i + 1
 9   endwhile
10   return d(v_i)
```

It is easy to see that our algorithm runs in time $O(n+m)$, since for at most every vertex (line 4 to 9), we walk along its list of neighbours (line 5 to 7), and sorting and initialising (line 1 to 3) can be done in that time as well.

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 6 and the description of the algorithm to compute γ_R). 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 three 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. The heuristic $\gamma_R D$-min-e (minimum number of edges) chooses to remove either V' or v_j depending on which of the two deletes fewer edges to obtain an induced subgraph with as many edges as possible.

δ-contraction degeneracy: For computing lower bounds for δC, we have examined various strategies for contraction in [8]. 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. [8, 9] 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, [8, 9] and this article) can be obtained from [26].

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 [17]) 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. 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 three $\gamma_R D$-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.

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

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

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 [8]). For $\delta_2 C$ and $\gamma_R C$, the results of the heuristics as described in Section 4.2 are shown.

instance	δC		$\delta_2 C$						$\gamma_R C$							
	least-c		all-v		MSD+1		MSD+2		left		right		min-e		all-v	
	LB	CPU	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	11	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	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	0.01	7	312.92
pignet2	38	0.11	40	369.00	39	0.12	39	0.14	38	0.12	39	0.12	39	0.11	40	11525.15
celar06	11	0.00	11	0.16	11	0.01	11	0.00	11	0.00	11	0.00	11	0.01	11	0.30
celar07pp	15	0.00	15	0.77	15	0.01	15	0.01	15	0.00	15	0.01	15	0.00	15	2.08
graph04	20	0.01	20	2.72	20	0.01	19	0.01	20	0.02	19	0.01	20	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	5	0.03	6	221.72
school1	122	0.48	124	180.30	123	0.48	122	0.51	122	0.45	122	0.49	122	0.45	125	215.35
school1-nsh	106	0.37	108	173.51	106	0.35	107	0.38	104	0.34	106	0.36	106	0.32	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	0.03	50	5.43

Table 2. Treewidth lower bounds with contraction

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 5). In some cases other small improvements (compared to δD and $\delta_2 D$) could be obtained by the heuristics for $\gamma_R D$. The three $\gamma_R D$-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 5, 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 bigger than 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 7 is far from the ratios observed in our experiments.

It was already remarked in [8] 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 [28]). 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 [8] 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 [22].

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. However, an exception is the computation of the γ_R-degeneracy, which has been shown to be NP-hard. Figure 4 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 Lemmas 2, 3 and 4. The thin line marks the border between polynomial computability and NP-hardness of the corresponding parameters.

In our experiments, we could observe (as in [8]) potent improvements when comparing the simple parameters with their degeneracy counterparts. An even bigger improvement was achieved when edge-contractions (i.e. minors) 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 [28]). 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 [16, 25]).

References

1. S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability – A survey. *BIT*, 25:2–23, 1985.
2. S. Arnborg and A. Proskurowski. Characterization and recognition of partial 3-trees. *SIAM J. Alg. Disc. Meth.*, 7:305–314, 1986.
3. M. Behzad, G. Chartrand, and L. Lesniak-Foster. *Graphs and Digraphs.* Pindle, Weber & Schmidt, Boston, 1979.
4. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comp. Sc.*, 209:1–45, 1998.
5. 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.
6. 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.

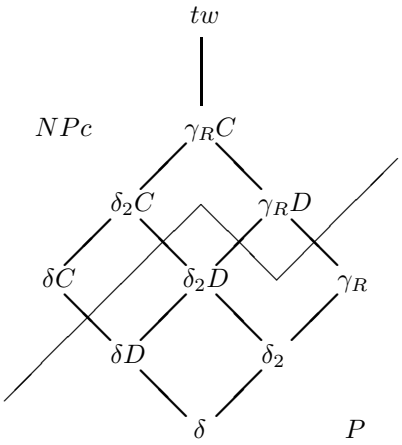

Fig. 4. An overview over some theoretical results

7. H. L. Bodlaender, A. M. C. A. Koster, F. v. d. Eijkhof, and L. C. van der Gaag. Pre-processing 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1998.
13. 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.
14. M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.
15. V. Gogate and R. Dechter. A complete anytime algorithm for treewidth. In proceedings UAI'04, Uncertainty in Artificial Intelligence, 2004.
16. I. V. Hicks. Planar branch decompositions I: The ratcatcher. INFORMS Journal on Computing (to appear, 2005).
17. 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.
18. 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.
19. 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.
20. B. Lucena. A new lower bound for tree-width using maximum cardinality search. *SIAM J. Disc. Math.*, 16:345–353, 2003.
21. B. Mohar and C. Thomassen. *Graphs on Surfaces*. The Johns Hopkins University Press, Baltimore, 2001.

22. S. Ramachandramurthi. *Algorithms for VLSI Layout Based on Graph Width Metrics*. PhD thesis, Computer Science Department, University of Tennessee, Knoxville, Tennessee, USA, 1994.
23. S. Ramachandramurthi. The structure and number of obstructions to treewidth. *SIAM J. Disc. Math.*, 10:146–157, 1997.
24. N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7:309–322, 1986.
25. P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.
26. Treewidthlib. http://www.cs.uu.nl/people/hansb/treewidthlib, 2004-03-31.
27. 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.
28. 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.