Contraction and Treewidth Lower Bounds

Hans L. Bodlaender

Arie M. C. A. Koster

Thomas Wolle

institute of information and computing sciences, utrecht university

technical report UU-CS-2004-034

www.cs.uu.nl

Contraction and Treewidth Lower Bounds*

Hans L. Bodlaender† Arie M. C. A. Koster‡ Thomas Wolle§

Abstract

Edge contraction is shown to be a useful mechanism to improve lower bound heuristics for treewidth. A successful lower bound for treewidth is the degeneracy: the maximum over all subgraphs of the minimum degree. The degeneracy is polynomial time computable. We introduce the notion of contraction degeneracy: the maximum over all minors of the minimum degree. We show that the contraction degeneracy problem is NP-complete, even for bipartite graphs, but for fixed k, it is polynomial time decidable if a given graph G has contraction degeneracy at least k. Heuristics for computing the contraction degeneracy are proposed and evaluated. It is shown that these can lead in practice to considerable improvements of the lower bound for treewidth, but can perform arbitrarily bad on some examples. A study is also made for the combination of contraction with Lucena's lower bound based on Maximum Cardinality Search [23]. Finally, heuristics for the treewidth are proposed and evaluated that combine contraction with a treewidth lower bound technique by Clautiaux et al. [12].

1 Introduction

It is about two decades ago that the notion of treewidth and the equivalent notion of partial k-tree were introduced. Nowadays, these play an important role in many theoretic studies in graph theory and algorithms, but also their use for practical applications is growing, see e.g. [21, 22]. A first step when solving problems on graphs of bounded treewidth is to compute a tree decomposition of (close to) optimal width, on which often a dynamic programming approach is applied. Such a dynamic programming algorithm typically has a running time that is exponential in the treewidth of the graph. Since the treewidth problem is NP-complete [1], it is rather unlikely to find efficient algorithms for computing the treewidth. Therefore, we are interested in lower and upper bounds for the treewidth of a graph.

This paper focuses on lower bounds on the treewidth of a graph. Good lower bounds can serve to speed up branch and bound methods, inform us about the quality of upper bound heuristics, and in some cases, tell us that we should not use tree decompositions to solve a problem on a certain instance. A large lower bound on the treewidth of a graph implies that we should not hope for computationally efficient dynamic programming algorithms that use tree decompositions for this particular instance.

More work has been done recently on practical algorithms for determining the treewidth of graphs, for instance on preprocessing methods (see [7, 8, 32]), upper bound heuristics (e.g. [12, 11, 18, 20]), lower bound heuristics (e.g. [6, 12, 23, 25]), and some exact methods (e.g. [18, 29]). In many cases, exact methods are still too slow, and for many instances, there are large gaps between the bounds given by upper bound and lower bound heuristics. Thus, the study of algorithms and heuristics for treewidth remains interesting also from a practical point of view.

*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*).

†Institute of Information and Computing Sciences (IICS), Utrecht University, P. O. Box 80.089, 3508 TB Utrecht, the Netherlands, email: hansb@cs.uu.nl

‡Zuse Institute Berlin (ZIB), Takustraße 7, D-14194 Berlin, Germany, email: koster@zib.de

§IICS, Utrecht University, email: thomasw@cs.uu.nl

In this paper, we propose and study algorithms that find lower bounds for treewidth, that use contraction of edges. In each of the algorithms, we have a combination of contraction with existing lower bound methods. In particular, we study how contraction can be used to improve the degeneracy (or MMD) lower bound [20], the lower bound based on maximum cardinality search MCSLB, introduced by Lucena [23], and the technique introduced by Clautiaux et al. [12]. Descriptions of these existing lower bound methods can be found in Section 2.

Contraction of an edge is the operation that replaces its two endpoints by a single vertex, which is adjacent to all vertices at least one of the two endpoints was adjacent to. Combining the notion of contraction with degeneracy gives the new notion of *contraction degeneracy* of a graph G: the maximum over all graphs that can be obtained by contracting edges and taking subgraphs of G of the minimum degree. It provides us with a new lower bound on the treewidth of graphs. While unfortunately, computing the contraction degeneracy of a graph is NP-complete (as is shown in Section 3.2), the fixed parameter cases are polynomial time solvable (see Section 3.3), and there are simple heuristics that provide us with good bounds on several instances taken from real life applications (see Section 5).

In a very recent paper, Gogate and Dechter [18] propose a branch and bound algorithm for treewidth with a good anytime performance. Independently of our work, Gogate and Dechter also propose the lower bound heuristic which we call the MMD+(min-d) heuristic in this paper. We compare this heuristic with other heuristics in Section 6, and see that the strategy where we contract to a neighbour with minimum number of common neighbours almost always outperforms the MMD+(min-d) heuristic where we contract to a neighbour of minimum degree. For more details, see [18], Section 3 and Section 6.

The lower bound provided by MCSLB is never smaller than the degeneracy, but can be larger [6]. Motivated by this, we also studied contraction in combination with the MCSLB algorithm. Unfortunately, the problem to determine if some bound can be obtained with MCSLB for a graph obtained from G by contracting edges is also NP-complete (Section 4.2). Its fixed parameter case is linear time solvable (Section 4.3). We also studied some heuristics for this bound.

In our experiments, we have seen that, typically, the bound by MCSLB is equal to the degeneracy or slightly larger. In both cases, often a large increase in the lower bound is obtained when we combine the method with contraction. See Section 6 for results of our computational experiments. They show that contraction is a very viable idea for obtaining or improving treewidth lower bounds.

A further improvement to the lower bounds can be obtained by using a method found by Clautiaux et al. [12]. This method uses another treewidth lower bound algorithm as a subroutine. In [12], the authors use the degeneracy (or MMD) as a subroutine, but one can also use other algorithms. Our experiments showed that the contraction degeneracy heuristics generally outperform the method of [12] with degeneracy, but when we combine the method of [12] with the heuristics of this paper, we get in several cases an additional small improvement to the lower bound. We finally propose a heuristic that combines the method of [12] and contraction in another way, by doing a contraction between every round of 'graph improvement'. See Section 5 for more details. This latter heuristic often costs considerably more time, but can give also significant increases to the lower bound.

2 Preliminaries

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

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

Subgraphs and Minors. After deleting vertices of a graph and their incident edges, we get an *induced subgraph*. A *subgraph* is obtained, if we additionally allow deletion of edges. If we furthermore allow edge-contractions, we get a *minor*. It is known that the treewidth of a minor of G is at most the treewidth of G (see e.g. [4]). We explicitly exclude the null graph (the graph without vertices and without edges) as a minor or subgraph.

Edge-Contraction. Contracting edge $e = \{u, v\}$ in the graph $G = (V, E)$, denoted as G/e, is the operation that introduces a new vertex a_e and new edges, such that a_e is adjacent to all the neighbours of v and u, and 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: $H = G/E'$.

Treewidth. A *tree decomposition* of $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, with $\{X_i \mid i \in I\}$ a family of subsets of V, and T a tree, such that

- $\bigcup_{i \in I} X_i = V$.
- For all $\{v, w\} \in E$, there is an $i \in I$ with $v, w \in X_i$.
- 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.

One can alternatively define the treewidth in terms of chordal graphs. A graph is chordal, if and only if it does not contain an induced cycle with length at least four. The treewidth of a graph is exactly the minimum over all chordal graphs H that contain G as a subgraph of the size of the maximum clique in H minus 1, see [4].

Degeneracy/MMD. We also use the term MMD (Maximum Minimum Degree) for the degeneracy. The degeneracy δD of a graph G is defined to be:

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

The minimum degree of a graph is a lower bound on its treewidth, and the treewidth of G cannot increase by taking subgraphs. Hence, the treewidth of G is at least its degeneracy. (See also [20].)

Maximum Cardinality Search. MCS is a method to number the vertices of a graph. It was first introduced by Tarjan and Yannakakis for the recognition of chordal graphs [30]. We start by giving some vertex number 1. In step $i = 2, \ldots, n$, we choose an unnumbered vertex v that has the largest number of already numbered neighbours, breaking ties as we wish. Then we associate number i to vertex v. An *MCS ordering* ψ can be defined by mapping each vertex to its number: $\psi(v) :=$ number of v. For a fixed MCS ordering, let $v_i := \psi^{-1}(i)$.

Definition 1 *Let be given a graph G and an MCS ordering ψ of G, and let $v_i := \psi^{-1}(i)$. The visited degree $vd_\psi(v_i)$ of v_i is defined as follows:*

$$vd_\psi(v_i) := d_{G[v_1,\ldots,v_i]}(v_i)$$

The visited degree $MCSLB_\psi$ of an MCS ordering ψ is defined as follows:

$$MCSLB_\psi := \max_{i=1,\ldots,n} vd_\psi(v_i)$$

In [23], Lucena shows that for every graph G and MCS ordering ψ of G,

$$MCSLB_\psi \leq tw(G)$$

Thus, an MCS numbering gives a lower bound on the treewidth of a graph.

Improved Graphs. In [5], two notions of *improved graphs* were introduced. Let k be an integer. The $(k+1)$*-neighbours improved graph* $G' = (V, E')$ of $G = (V, E)$ is obtained as follows: we take G, and then, as long as there are non-adjacent vertices u, v, that have at least $k+1$ common neighbours in the graph, we add the edge $\{u, v\}$. This improvement step is motivated by the following lemma.

Lemma 1 (See [3, 5, 12].) *Any tree decomposition of G with width at most k is also a tree decomposition of the $(k+1)$-neighbours improved graph G' of G with width at most k, and vice versa.*

Clautiaux et al. use improved graphs to provide iterative methods to improve existing lower bounds for treewidth [12]. They use the MMD for computing lower bounds, but their approach works with every lower bound heuristic. Their algorithm LB_N works as follows:

- Suppose we have a lower bound $LB \leq tw(G)$ on the treewidth of G (e.g. LB was computed with the MMD heuristic).
- Use as hypothesis that $LB = tw(G)$. Build the $(LB+1)$-neighbours improved graph G' of G. (Note that if the hypothesis holds, then $tw(G) = tw(G')$)
- Compute a lower bound LB' of G' (e.g. with the MMD heuristic).
- If $LB' > LB$, we have a contradiction, showing the hypothesis $LB = tw(G)$ to be wrong.
- Therefore, $LB < tw(G)$ and $LB + 1$ is also a lower bound.
- Set LB to $LB + 1$, and repeat the process until there is no contradiction.

We see that the LB_N algorithm uses another treewidth lower bound algorithm as a subroutine, and thus, for every choice of such an algorithm, we obtain a different version of the LB_N algorithm. If algorithm Y is used as subroutine, then we call the resulting algorithm LBN(Y), e.g. the algorithm discussed by Clautiaux et al. in [12] is the LBN(MMD) algorithm.

In [12], Clautiaux et al. also propose a related method, that sometimes gives better lower bounds, but also uses more time. Here, we have a different notion of improved graph. Let k be an integer. The $(k+1)$*-paths improved graph* $G'' = (V, E'')$ of $G = (V, E)$ is obtained by adding an edge $\{u, v\}$ to E for all vertex pairs u and v such that there are at least $k+1$ vertex disjoint paths between u and v in G. Similar to Lemma 1, we have here the following.

Lemma 2 (See [3, 5, 12].) *Any tree decomposition of G with width at most k is also a tree decomposition of the $(k+1)$-paths improved graph G'' of G with width at most k, and vice versa.*

We can build the $(k+1)$-paths improved graph in polynomial time, as we can decide in polynomial time whether there are at least $k+1$ vertex disjoint paths between a pair of vertices with help of network flow techniques. However, the running time to compute the paths improved graph is much larger than for the neighbour version. If we use $(k+1)$-paths improved graphs instead of $(k+1)$-neighbours improved graphs, then we obtain a new lower bound heuristic for treewidth, called LB_P in [12]. If we use as subroutine a lower bound algorithm Y in this algorithm, we call the resulting algorithm LBP(Y).

3 Contraction Degeneracy

We first look at the treewidth lower bound heuristic, obtained by combining the degeneracy with contraction. The algorithm to compute the degeneracy of a graph repeatedly removes the vertex of minimum degree, and outputs the largest of these minimum degrees seen in the process. However, we can also get a lower bound for treewidth if we contract the vertex of minimum degree with a neighbour instead of deleting it. In this case, we can get different values if we make different choices which minimum degree vertex to select, and with which neighbour to contract it. The best way of doing these contractions is captured by the notion of *contraction degeneracy*.

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

3.1 Definition of the Problem

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

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

When G is connected, $\delta C(G)$ can also be defined as the maximum over all contractions of G of the minimum degree of the contraction. This does not necessarily hold for disconnected graphs: when G has connected components whose contraction degeneracy is smaller than the contraction degeneracy of G, we must delete this component entirely to obtain the minor with maximum minimum degree. The corresponding decision problem is formulated as usual:

Problem: Contraction Degeneracy
Instance: Graph $G = (V, E)$ and integer $k \geq 0$.
Question: Is the contraction degeneracy of G at least k?

Lemma 3 *For any graph G, we have that $\delta C(G) \leq tw(G)$.*

Proof: Note that for any minor G' of G, we have that $tw(G') \leq tw(G)$ (see e.g. [4]). Furthermore, for any graph G': $\delta(G') \leq tw(G')$. The lemma follows now directly. □

3.2 NP-completeness

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

Proof: Clearly, the problem is in NP as we only have to guess an edge set E', and then compute in polynomial time $\delta(G/E')$.

The hardness proof is a transformation from the Vertex Cover problem, which is known to be NP-complete, see [17]. In the Vertex Cover problem, we are given a graph $G = (V, E)$ and an integer k, and look for a vertex cover of size at most k, i.e. a set $W \subseteq V$ with $|W| \leq k$, such that each edge in E has at least one endpoint in W. Let be given a Vertex Cover instance (G, k), with $G = (V, E)$.

Construction: We build a graph in two steps. In the first step, we construct a graph G' by taking the complement $\bar{G}$ of G, two adjacent vertices and k pairwise non-adjacent vertices, and making the new vertices adjacent to each vertex in G. G' is formally defined as follows, see Figure 1:

$$\begin{aligned}
G' &:= (V', E') \text{ where} \\
V' &= V \cup \{w_1, w_2\} \cup \{u_1, \ldots, u_k\} \\
E' &= (\{\{v, w\} \notin E \mid v, w \in V, v \neq w\}) \cup \{\ \{w_1, w_2\}\ \} \\
&\quad \cup\{\ \{w_i, v\} \mid i \in \{1, 2\} \wedge v \in V\ \} \\
&\quad \cup\{\ \{u_i, v\} \mid i \in \{1, \ldots, k\} \wedge v \in V\ \}
\end{aligned}$$

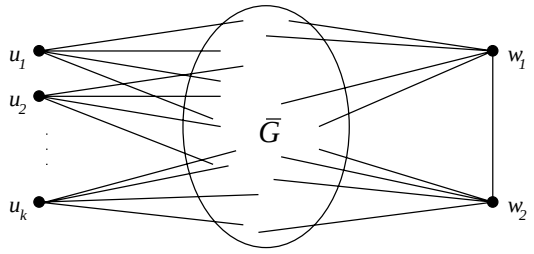

Figure 1: Intermediate graph G' constructed for the transformation.

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

$$\begin{aligned} G^* &:= (V^*, E^*) \text{ where} \\ V^* &= V' \cup \{\ v_e \mid e \in E'\ \} \\ E^* &= \{\ \{u, v_e\}, \{v_e, w\} \mid e = \{u, w\} \in E'\ \} \end{aligned}$$

Let be $n := |V|$. The constructed instance of the Contraction Degeneracy problem is $(G^*, n+1)$. G^* is a bipartite graph, as all edges in G^* are between a vertex in V' and a vertex in $\{\ v_e \mid e \in E'\ \}$. Now, we have to show that there is a vertex cover for G of size at most k if, and only if $\delta C(G^*) \geq n+1$.

Claim 1 *If there is a vertex cover of G of size at most k, then there is a $E_3 \subseteq E^*$, such that $\delta(G^*/E_3) \geq n+1$.*

Proof: Suppose there is a vertex cover of size at most k. Now take a vertex cover $V_1 = \{v_1, \ldots, v_k\}$ of G of size exactly k. (If we add vertices to a vertex cover, we obtain again a vertex cover.) We build the set of edges to be contracted in two steps. Let E_1 be the edge set, such that $G^*/E_1 = G'$, i.e. E_1 consists of $|E'|$ edges to undo the subdivisions. First, we contract the edges in E_1 and obtain G^*/E_1. Then, in G^*/E_1, we contract edge set E_2 defined as follows: $E_2 := \{\ \{u_i, v_i\} \mid i = 1, \ldots, k\}$. I.e. each vertex in the vertex cover has a vertex of the type u_i contracted to it. We write $E_3 := E_1 \cup E_2$. We claim that the resulting graph $G_3 := G^*/E_3$ is an $(n+2)$-clique. Assume there are two vertices x and y with $\{x, y\} \notin E(G_3)$. Since the vertices w_1 and w_2 are universal in G_3, we have $\{x, y\} \subseteq V$. Therefore, $\{x, y\} \in E$, and hence x or y is in V_1. We assume w.l.o.g. $x \in V_1$, i.e. $\exists i \in \{1, \ldots, k\}$, such that $v_i = x$. Since we contracted $\{u_i, v_i\}$ and $\{u_i, y\} \in E(G')$, we have $v_i = x$ is adjacent to y in G_3, which is a contradiction. Hence, $G_3 = G^*/E_3$ is an $(n+2)$-clique and $\delta(G^*/E_3) = n+1$. ◇

Claim 2 *If there is a $E_3 \subseteq E^*$, such that $\delta(G^*/E_3) \geq n+1$, then there is a vertex cover V_1 for G of size at most k.*

Proof: We have $|E'|$ vertices $V_{E'} := \{\ v_e \mid e \in E'\ \}$ of degree two in G^*, namely the subdivisions. Assuming G has more than just one vertex, i.e. assuming $n \geq 2$, we see that the degree of all vertices in $V_{E'}$ is too small. We have to contract edges, such that all vertices in $V_{E'}$ will get a larger degree. Hence, there is a $E_2 \subseteq E_3$, such that $G^*/E_2 = G'$. Let be $E_1 := E_3 \setminus E_2$. Because of the commutativity of contraction-operations, we assume that we first contract all edges in E_2. A vertex $u_i, \forall i \in \{1, \ldots, k\}$ has degree exactly n in G'. Thus, for each $u_i, i \in \{1, \ldots, k\}$, we have to contract an edge incident to u_i. After contracting these edges, there are $n+2$ vertices left in the graph. Therefore we cannot contract another edge, since then we could not obtain the minimum degree of $n+1$. Furthermore, we see that G^*/E_3 is an $(n+2)$-clique. Hence, E_1 contains exactly k edges, one for every $u_i, i \in \{1, \ldots, k\}$, with the other endpoint in V. Let be $V_1 := \bigcup_{e \in E_1} e \ \setminus\ \bigcup_{i=1,\ldots k} u_i$. Clearly, $|V_1| = k$, and we claim that V_1 is a vertex cover of G. Assume, there is an edge $f = \{x, y\}$ in G with $V_1 \cap f = \emptyset$. Hence, f is not an edge in G'. Since

G^*/E_3 is an $(n+2)$-clique, edge f exists in G^*/E_3, which means: f was created by contracting another edge $\{u_i, v_j\} \in E_1$. This can only be the case if $v_j = x$ or $v_j = y$. 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 k. $\diamond$

As G^* can be constructed in polynomial time, NP-completeness of the Contraction Degeneracy problem now follows. $\square$

3.3 Fixed Parameter Cases of Contraction Degeneracy

Now, we consider the fixed parameter case of the Contraction Degeneracy problem. I.e. for a fixed integer k, we consider the problem to decide for a given graph G if G has a minor with minimum degree k. Graph minor theory gives a fast answer to this problem. For a good introduction to the algorithmic consequences of this theory, see [16].

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

Proof: Let k be a fixed integer. Consider the class of graphs $\mathcal{G}_k = \{G \mid G$ has contraction degeneracy at most $k-1\}$. $\mathcal{G}_k$ is closed under taking minors: if H is a minor of G and H has contraction degeneracy at least k, then G has also contraction degeneracy at least k. As every class of graphs that is closed under minors has an $O(n^3)$ algorithm to test membership by Robertson-Seymour graph minor theory (see [16]), the theorem for the case that $k \geq 6$ follows.

Suppose now that $k \leq 5$. There exists a planar graph G_k with minimum degree k (for example for $k = 5$ the icosahedron, see [10]). Hence, $G_k \notin \mathcal{G}_k$. A class of graphs that is closed under taking minors and does not contain all planar graphs has a linear time membership test (see [16]), which shows the result for the case that $k \leq 5$. $\square$

It can be noted that the cases that $k = 1, 2$ and 3 are very simple: a graph has contraction degeneracy at least 1, if and only if it has at least one edge, and it has contraction degeneracy at least 2, if and only if it is not a forest. For a graph to have contraction degeneracy at least 3, all vertices of degree 2 or less have to be contracted recursively. If the result is a non-empty graph, the contraction degeneracy is at least 3. Vertices of degree 2 can be contracted to either of the neighbours without loss of generality. In the same way graphs that have treewidth at least 3 are identified [2, 8], and hence graphs with $\delta C(G) \geq 3$ are exactly those with $tw(G) \geq 3$.

The result is non-constructive when $k \geq 6$; when $k \leq 5$, the result can be made constructive by observing that the property that G has contraction degeneracy k can be formulated in monadic second order logic for fixed k. Thus, we can solve the problem as follows: the result of [28] applied to G_k, a planar graph with minimum degree k, gives an explicit upper bound c_k on the treewidth of graphs in $\mathcal{G}_k = \{G \mid G$ has contraction degeneracy at most $k-1\}$. Test if G has treewidth at most c_k, and if so, find a tree decomposition with width at most c_k with the algorithm of [3]. If G has treewidth at most c_k, use the tree decomposition to test if the MSOL formula holds for G [14]; if not, we directly know that G has contraction degeneracy at least k. It should also be noted that the constant factors hidden in the O-notation of these algorithms are very large; it would be nice to have practical algorithms that do not rely on graph minor theory. We summarise the different cases in the following table.

4 Maximum Cardinality Search with Contraction

As discussed in Section 2, we obtain a lower bound on the treewidth of a graph from a maximum cardinality search ordering. We now study the combination of this MCSLB heuristic with contraction, and we analyse the complexity of finding an optimal way of contracting and building an MCS ordering to obtain the best lower bound possible with this method. We define four computation problems, and show that each of these is either NP-complete or NP-hard, respectively. For some of these, we also can show that the fixed parameter cases are tractable.

k	Time	Reference
1	$O(n)$	Trivial
2	$O(n)$	G is not a forest
3	$O(n)$	$tw(G) \geq 3$
4, 5	$O(n)$	[3, 14, 28], MSOL
fixed $k \geq 6$	$O(n^3)$	[27, 26]
variable k	NP-complete	Theorem 1

Table 1: Complexity of contraction degeneracy

4.1 Definition of the Problems

We consider the following problem and variants.

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

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

Problem: MinMCSLB With Contraction
Instance: Graph $G = (V, E)$, integer k.
Question: Does G have a contraction H, such that every MCS ordering ψ has visited degree at least k?

Problem: MinMCSLB With Minors
Instance: Graph $G = (V, E)$, integer k.
Question: Does G have a minor H, such that every MCS ordering ψ has visited degree at least k?

4.2 NP-completeness

Theorem 3 MCSLB With Contraction *is NP-complete.*

Proof: Clearly MCSLB With Contraction belongs to NP. We just have to guess a contraction H and an MCS ordering ψ and check in polynomial time, whether the visited degree of ψ in H is at least k.

To prove NP-hardness, we use a transformation from Vertex Cover. Let be given a Vertex Cover instance (G, k), where $G = (V, E)$ with $n = |V|$, and k is an integer. We construct a graph G' in the following way:

Construction. First, we take $n + 2$ copies of the complement of G. We call the vertices in these copies *graph vertices*. We add $k \cdot (n + 2)$ *extra vertices*. Each extra vertex has degree n: it is adjacent to all graph vertices in one copy of $\bar{G}$ and no other vertex; each copy has exactly k such extra vertices. Hence, in total, we have $k(n + 2)$ extra vertices. Finally, we add an edge between each pair of graph vertices that belong to different copies. Let G' be the resulting graph, see Figure 2. The MCSLB With Contraction instance is $(G', n(n + 2) - 1)$.

Now, we will show that G' has a contraction H that has an MCS ordering ψ with the visited degree of ψ at least $n(n + 2) - 1$, if and only if G has a vertex cover of size at most k.

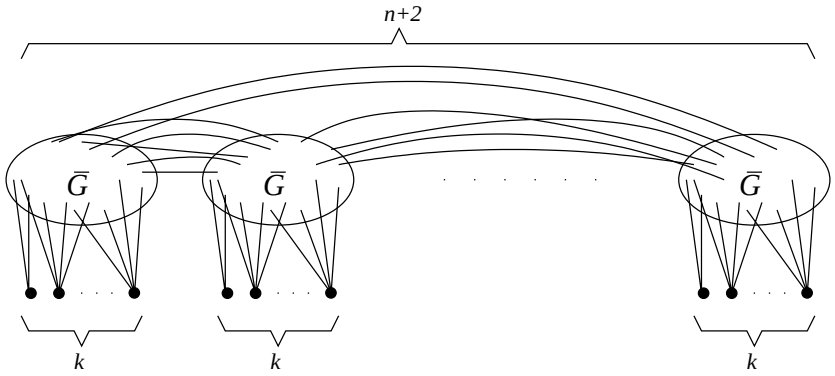

Figure 2: The graph G' constructed for the transformation.

Claim 3 *If G has a vertex cover of size at most k, then G' has a contraction H that has an MCS ordering ψ with the visited degree of ψ at least $n(n+2)-1$.*

Proof: Let V' be a vertex cover of G of size at most k. Now, we perform the following in each copy of $\bar{G}$. Contract all the extra vertices to vertices in the vertex cover V', such that each vertex in V' has at least one extra vertex contracted to it. This turns the set of graph vertices of G' into a clique of size $n(n+2)$, because for each pair of nonadjacent graph vertices v, w in G', $\{v, w\}$ is an edge in G, so an extra vertex, adjacent to v and w is contracted to v or w, after which the edge $\{v, w\}$ is formed in H. (Compare with the proof of Claim 1.) As H is a clique of $n(n+2)$ vertices, any MCS ordering of H has visited degree exactly $n(n+2)-1$. ◇

Now, we will show the other direction. For this, we need a series of claims. Suppose that G' has a contraction H that has an MCS ordering ψ with the visited degree of ψ at least $n(n+2)-1$. Let y be the first vertex in ψ that is visited with visited degree $n(n+2)-1$, and let Y be the vertices that are visited up to y (including y). Note that y must be a graph vertex. By Lucena's theorem [23], $H[Y]$ has treewidth at least $n(n+2)-1$. Let X be the set of the vertices in H that are extra vertices that are not contracted.

Claim 4 *There are at most $n+1$ copies of $\bar{G}$ that have at least one extra vertex that belongs to $X \cap Y$.*

Proof: Consider the MCS ordering ψ up to the point that there are $n+1$ copies of $\bar{G}$ with at least one extra vertex in $X \cap Y$. As the set of visited vertices is connected, each copy must have a (possibly contracted) graph vertex that is visited. Before we can visit a vertex in X of the last copy, we must first visit a (possibly contracted) graph vertex of that copy. After that visit, each graph vertex has visited degree at least $n+1$, while vertices in X have degree at most n, so yet unvisited vertices in X will not be visited before all graph vertices are visited, in particular, only after y is visited. ◇

So, there is at least one copy of $\bar{G}$ that has no uncontracted extra vertices in Y. Let V_i be the set of vertices of that copy in Y.

Claim 5 *There are at least $n(n+2)$ graph vertices in Y.*

Proof: If the opposite holds, then the treewidth of $H[Y]$ would be less than $n(n+2)-1$. Consider e.g. the following triangulation of $H[Y]$: turn the set of (possibly contracted) graph vertices into a clique. The maximum clique size will be less than $n(n+2)$ and the treewidth less than $n(n+2)-1$. This contradicts the fact that the treewidth of $H[Y]$ is at least $n(n+2)-1$. ◇

Claim 6 *V_i is a clique.*

Proof: Assume the opposite. Let v and w be non-adjacent vertices in V_i. We can triangulate $H[Y]$ as follows: Add an edge between each pair of non-adjacent (possibly contracted) graph vertices, except that we do not add the edge $\{v, w\}$. Since V_i does not have extra vertices that are not contracted, this gives a chordal graph. The vertices in X are simplicial with degree at most n. After we remove these, we get a graph that is obtained by removing an edge from a clique with at most $n(n+2)$ vertices, yielding a graph with clique-size at most $n(n+2)-1$. Hence the treewidth is at most $n(n+2)-2$, which is a contradiction. ◇

Because there are $n(n+2)$ graph vertices in Y, we know that $|V_i| = n$, and we cannot have contracted other graph vertices to vertices in V_i, since then we would have less then $n(n+2)$ graph vertices in Y. So, V_i was formed into a clique by the contraction of the k extra vertices of the copy to the graph vertices in V_i. Let Z be the set of vertices in V_i that have an extra vertex contracted to it. We have $|Z| \leq k$.

Claim 7 *Z is a vertex cover.*

Proof: For each edge $\{v, w\} \in E$, v and w are non-adjacent in H. Thus, we must have an extra vertex contracted to v or an extra vertex contracted to w. Therefore, we have $v \in Z$ or $w \in Z$ for each edge $\{v, w\} \in E$. ◇

Hence, we can conclude that if G' has a contraction H that has an MCS ordering ψ with the visited degree of ψ at least $n(n+2)-1$, then G has a vertex cover of size at most k, which proves the other direction. The proof of the NP-completeness of MCSLB WITH CONTRACTION is now complete. □

The same proof can be used for the related problems given in Section 4.1, except that membership in NP is trivial only for MCSLB WITH MINORS, and we have no proof for membership in NP for the other two problems. Therefore, we conclude the following statement.

Corollary 1 MCSLB WITH MINORS *is NP-complete, and* MINMCSLB WITH CONTRACTION *and* MINMCSLB WITH MINORS *are NP-hard.*

4.3 Fixed Parameter Cases

The fixed parameter case of MCSLB WITH MINORS can be solved in linear time with help of graph minor theory. Observing that the set of graphs $\{G \mid G$ does not have a minor H, such that H has an MCS ordering ψ with the visited degree of ψ at least $k\}$ is closed under taking of minors, and does not include all planar graphs (see [6]), gives us by the Graph Minor theorem of Robertson and Seymour and the results in [3, 28] the following result. See again [16] for more background information.

Theorem 4 MCSLB WITH MINORS *and* MINMCSLB WITH MINORS *are linear time solvable for fixed k.*

The fixed parameter cases of MINMCSLB WITH MINORS give an $O(n^3)$ algorithm (linear when $k \leq 5$), similar as for Theorem 2. Note that these results are non-constructive, and that the constant factors in the O-notation of these algorithms can be expected to be too large for practical purposes.

5 Heuristics

In this section, we discuss a number of heuristics, each giving a lower bound for treewidth. In Section 6, we discuss experimental evaluations of the heuristics. Here, we describe the heuristics, and in some cases, give some analysis of them. In Section 5.1, we propose and analyse some heuristics for the contraction degeneracy. In Section 5.2, we discuss heuristics for MCSLB with contraction. In Section 5.3, we look at the LBN and LBP heuristics, introduced in [12]. These can be combined with any of the other heuristics, but we also propose a new heuristic where contractions alternate with constructions of neighbours or paths improved graphs.

5.1 Heuristics for the Contraction Degeneracy

An almost trivial heuristic for the contraction degeneracy is the degeneracy, $\delta D(G)$. We denote it in our overviews with the shorter abbreviation MMD ('Minimum Maximum Degree'). It can be computed in linear time, by iteratively selecting a vertex of minimum degree, and deleting it and its incident edges. The largest seen minimum degree in these steps is the degeneracy.

From this algorithm, we derive the MMD+ algorithm (with three variants.) In this algorithm, we select a vertex v of minimum degree, and contract it with one of its neighbours u. In each case, the algorithm outputs the maximum over all vertices of its degree when it was selected as minimum degree vertex. Clearly, this is a lower bound on the contraction degeneracy of a graph. We consider three strategies how to select a neighbour:

- *min-d* selects a neighbour with minimum degree. This heuristic is motivated by the idea that the smallest degree is increased as fast as possible in this way.
- *max-d* selects a neighbour with maximum degree. This heuristic is motivated by the idea that we end up with some vertices of very high degree.
- *least-c* selects a neighbour u of v, such that u and v have the least number of common neighbours. Note that for each common neighbour w of u and v, the two edges' $\{u, w\}$ and $\{v, w\}$ become the same edge in the graph, meaning that for each common neighbour, effectively one edge is removed from the graph. Thus, the least-c heuristic is motivated by the idea to delete as few as possible edges in each iteration to get a high minimum degree.

We call the resulting heuristics MMD+(min-d), MMD+(max-d) and MMD+(least-c). In Section 6, we experimentally evaluate these heuristics. While the heuristics (and especially the least-c heuristic) do often well in practice, unfortunately, each of the three heuristics can do arbitrarily bad. In Sections 5.1.1 – 5.1.4, we give examples of graphs where there is a large difference between the contraction degeneracy and a possible lower bound for it obtained by the considered heuristic.

We observe that each of the MMD+ heuristics gives a value that is at least the degeneracy: consider a subgraph H of G with minimum degree the degeneracy of G. Consider the graph G' that we currently have just before the first vertex v from H is to be contracted by the heuristic. All neighbours of v in H are also neighbours of v in G', hence the algorithm gives as bound at least the degree of v in H, hence at least the degeneracy of G.

The minimum degree of a vertex is also a trivial lower bound for both the treewidth and the contraction degeneracy. We call this heuristic MD; it plays a role in combination with a technique based on work by Clautiaux et al. [12], see Section 5.3.

5.1.1 Degeneracy versus contraction degeneracy

The MMD algorithm can perform arbitrarily bad. Consider a clique with n vertices, and then subdivide every edge. Let G be the resulting graph. Clearly, $\delta(G) = 2$. We also have $\delta D(G) = 2$ since all subdivisions have degree 2 and must be deleted, which also deletes all edges in G. However, $\delta C(G) = n-1$, because undoing the subdivisions results in an n-clique with minimum degree $n-1$.

5.1.2 A bad example for the MMD+(max-d) heuristic

An example where the MMD+(max-d) heuristic can perform arbitrarily bad is not hard to find. One simple example is the following. Take a clique with n vertices, subdivide every edge, and then add one universal vertex x. (I.e. x is adjacent to each other vertex in the graph.) Let G be the resulting graph. The MMD+(max-d) heuristic will contract each vertex to x, and hence will give 3 as a result. However, $\delta C(G) = n$, since if we contract the subdivision vertices to the clique vertices, we obtain a clique with $n + 1$ vertices.

5.1.3 A bad example for the MMD+(min-d) heuristic

The example where the MMD(min-d) performs bad is somewhat more involved. For each r, we build a graph where the min-d heuristic can possibly give a lower bound of three, while the contraction degeneracy of the graph is r. We assume, as 'adversary', that we can decide in which way tie-breaking is done (i.e. the adversary can select a vertex among those who have minimum degree.)

Let $r \geq 3$ be some integer. Build a graph G_r as follows. We take for each i, j, $1 \leq i < j \leq r$ a vertex v_{ij}. We take for each i, $1 \leq i \leq r$ a vertex w_i. We take a vertex x. Now, we add edges $\{v_{ij}, w_i\}$, $\{v_{ij}, w_j\}$ and $\{v_{ij}, x\}$, for each i, j, $1 \leq i < j \leq r$.

To the graph thus obtained, we add a number of cliques. Each clique consists of three new vertices and one vertex of the type w_i, $1 \leq i \leq r$ or x. We have one such clique that contains x. For each i, we take r^2 such cliques that contain w_i, $1 \leq i \leq r$. (It is possible to make a more compact construction, using about $r^2/6$ cliques.) We call the new vertices in these cliques the *additional clique vertices*. In this way, each w_i has a degree that is larger than $3r^2$. Let G_r be the resulting graph. See Figure 3.

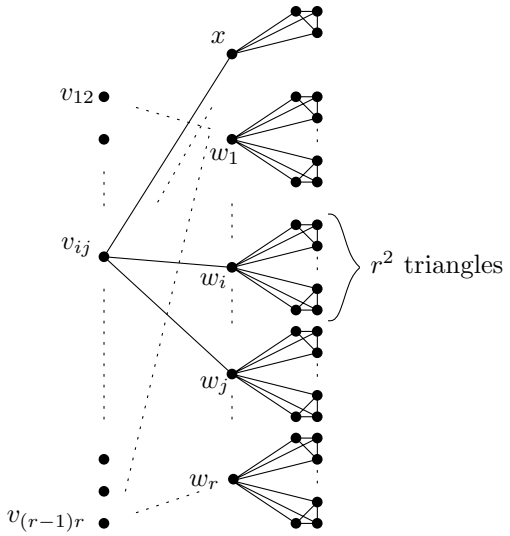

Figure 3: The structure of G_r

Proposition 1 *Let $r \geq 3$. The contraction degeneracy and treewidth of G_r equal r.*

Proof: If we contract v_{1r} to w_r and each other vertex of the form v_{ij} to w_i, and each additional clique vertex to its neighbour of type w_i or x, then the resulting graph is a clique on $\{w_i \mid 1 \leq i \leq r\} \cup \{x\}$. Each vertex in this clique has degree r, so the contraction degeneracy of G_r is at least r, and hence the treewidth of G_r is at least r.

If we add to G_r an edge between each distinct pair of vertices in $\{w_i \mid 1 \leq i \leq r\} \cup \{x\}$, then we obtain a chordal graph with maximum clique size $r + 1$. So, the treewidth of G_r is at most r, and hence also its contraction degeneracy is at most r. □

Proposition 2 *The MMD+(min-d) heuristic can give a lower bound of three when applied to G_r.*

Proof: Consider the following start of a sequence of contractions: first, contract the vertices of the form v_{ij} one by one to x, for all i, j, $1 \leq i < j \leq r$. Note that the min-d heuristic can start with this sequence: at each point during this phase, the vertices of the form v_{ij} have degree three, which is the minimum degree in the graph, and the degree of x is at most $r(r-1)/2+3+r$, which is less than the degree of vertices of the form w_i, which have degree at least $3r^2$. So, during this

first part of the running of the algorithm, the bound for the contraction degeneracy is not larger than three.

After all vertices v_{ij} have been contracted to x, the graph has the following form: x is adjacent to all w_i; there are no edges between vertices w_i, $w_{i'}$, $i \neq i'$; and then there are a number of four-cliques that have one vertex in common with the rest of the graph. This is a chordal graph with maximum clique size four. So, this graph has treewidth three, and hence contraction degeneracy at most three. Hence, the min-d heuristic cannot give a bound larger than three in the remainder of the run of the algorithm. Thus, the maximum bound it obtains can be three. □

Corollary 2 *The MMD+(min-d) heuristic can give a solution that is a factor of $\Omega(\sqrt{n})$ away from optimal.*

We can use cliques with four instead of three additional clique vertices. In that case, it holds that *every possible* run of the MMD+(min-d) heuristic gives a lower bound of four on the graph.

5.1.4 A bad example for the MMD+(least-c) heuristic

The example for the MMD+(least-c) heuristic is a modification of the one for the min-d heuristic.

Let $r \geq 3$ be again an integer. We take G_r and modify it as follows. Each edge of the form $\{v_{ij}, w_i\}$ or $\{v_{ij}, w_j\}$ is replaced by the structure given in Figure 4. In words: the edge is subdivided, and we add a clique with three new vertices and the subdivision vertex to the graph. Let G'_r be the resulting graph.

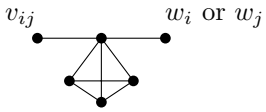

Figure 4: The structure that replaces edges of the form $\{v_{ij}, w_i\}$ or $\{v_{ij}, w_j\}$

Proposition 3 *Let $r \geq 3$. The contraction degeneracy and treewidth of G'_r equal r.*

Proof: We can contract G'_r to G_r: contract each structure as in Figure 4 to the vertex of the form v_{ij}. So, the contraction degeneracy of G'_r is at least the contraction degeneracy of G_r, hence at least r. So the treewidth of G'_r is at least r.

The treewidth of G'_r is at most r: Add to G'_r edges between each pair of distinct vertices in $\{w_i \mid 1 \leq i \leq r\} \cup \{x\}$. Then, for each i, j, $1 \leq i < j \leq r$, add edges $\{v_{ij}, w_i\}$ and $\{v_{ij}, w_j\}$. This gives a chordal graph with maximum clique size $r + 1$. So, the treewidth of G'_r is at most r. Hence, its contraction degeneracy is also at most r. □

Proposition 4 *The MMD+(least-c) heuristic can give a bound of three when applied to G'_r.*

Proof: Like for the min-d heuristic, the algorithm can start with contracting each vertex of the form v_{ij} to x. During this phase, vertices v_{ij} have the minimum degree in the graph, namely three, and have no common neighbours with x. So, during this phase, the lower bound is set to three.

After all vertices of the form v_{ij} are contracted to x, the graph G'' has treewidth three. This can be seen as follows. The treewidth of a graph is the maximum treewidth of its biconnected components. The biconnected components of G'' are either cliques with four vertices, single edges, or consist of x, a vertex w_i, and a number of paths of length two between x and w_i (for some i, $1 \leq i \leq r$). In the first case, the treewidth of the component is three; in the last case, the component has treewidth three. So, after the contractions of the v_{ij}-vertices to x, the bound of three cannot be increased. □

Corollary 3 *The MMD+(least-c) heuristic can give a solution that is a factor of $\Omega(\sqrt{n})$ away from optimal.*

It is possible to modify the construction such that any run of the MMD+(least-c) heuristic gives a result far from optimal. Instead of cliques with three new vertices and one 'old' vertex, we use cliques with five new vertices and one old vertex. The structure of Figure 4 is replaced by the structure of Figure 5. In this way, we obtain a graph that has contraction degeneracy r, but for which any run of the MMD+(least-c) heuristic gives a lower bound that is at most five.

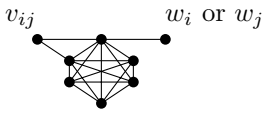

Figure 5: An alternative structure that replaces edges of the form $\{v_{ij}, w_i\}$ or $\{v_{ij}, w_j\}$

5.2 Heuristics for MCSLB With Contractions

Based on the result by Lucena [23] that the visited degree of an MCS ordering of G is a lower bound for the treewidth, we will look at heuristics based upon maximum cardinality search and contraction.

For comparison, we have the MCSLB heuristic. This heuristic computes $|V|$ MCS orderings ψ – one for each vertex as start vertex. It returns the maximum over these orderings of $MCSLB_\psi$, cf. [6].

The MCSLB+ heuristic starts by using MCSLB to find a start vertex w with largest $MCSLB_\psi$. Then, we iteratively select a vertex and a neighbour to contract, compute an MCS ordering, and repeat until the graph has no edges. To reduce the CPU time consumption, an MCS is carried out only with start vertex w (or vertices resulting from contractions that involve w) instead of with all possible start vertices. Three strategies for selecting a vertex v to be contracted are examined:

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

Once a vertex v is selected, we select a neighbour u of v using the two strategies *min-d* and *least-c* that are already explained for MMD+. We thus have six versions of the MCSLB+ heuristic. These are experimentally evaluated in Section 6. We did not evaluate the MCSLB+(max-d) heuristic, because of the negative experimental results for the MMD+(max-d) strategy.

5.3 Contraction and the LBN and LBP Heuristics

For each treewidth lower bound algorithm Y, we have two lower bound heuristics LBN(Y) and LBP(Y), based on the technique by Clautiaux et al. [12], see Section 2. So, we can have, e.g. the LBN(MMD+) algorithm.

A different method to combine the LBN or LBP methods with contraction is to alternate improvement steps with contractions. We describe the LBN+(Y) algorithm for some treewidth lower bound heuristic Y below. If we instead of making a neighbours improved graph, take a paths improved graph, we obtain the LBP+(Y) algorithm; the latter one is slower but often gives better bounds.

Algorithm LBN+(Y)
1 Initialise L to some lower bound on the treewidth of G, e.g. $L = 0$.
2 $H := G$.
3 repeat
4 H = the $(L+1)$-neighbours improved graph of H.
5 b :=Y(H)

6 if $b > L$
7 then $L := L + 1$
8 goto step 2
9 else select a vertex v of minimum degree in H.
10 select neighbour u of v according to the least-c strategy.
11 contract the edge $\{u, v\}$ in H.
12 endif
13 until H is empty
14 output L.

In the description above, we used the least-c strategy, as this one performed best for the other heuristics; of course, variants with other contraction strategies can also be considered.

Proposition 5 *If algorithm Y is an algorithm that outputs a lower bound on the treewidth of its input graph, then LBN+(Y) and LBP+(Y) output lower bounds on the treewidth of its input graph.*

Proof: Let G be the input graph of algorithm LBN+(Y) or LBP+(Y). An invariant of the algorithm is that the treewidth of G is at least L. A second invariant of the algorithm is that when the treewidth of G equals L, then the treewidth of H is at most L. Clearly, these invariants hold initially. Lemmas 1 and 2 show that the second invariant holds also after making an improved graph in step 4. The fact that contraction cannot increase treewidth shows that the second invariant holds after a contraction in step 11. Similar as in [12], when Y outputs a value larger than L on H, then the treewidth of H and hence the treewidth of G (by the second invariant) is larger than L, so increasing L by one in step 7 maintains the first invariant. □

In our experiments, we started the algorithm by setting the lower bound L to the value computed by the MMD+ heuristic.

5.3.1 Faster Implementation of LBP+

A straightforward implementation of an LBP+(Y) heuristic can be very slow. However, we can observe that some steps are not necessary. Contracting an edge can increase the number of vertex disjoint paths between two vertices, but not for all pairs. Lemma 4 tells us that contracting an edge $\{x, y\}$ cannot increase the number of vertex disjoint paths between u and v, if $\{x, y\} \cap \{u, v\} = \emptyset$.

Lemma 4 *Let be given vertices u and v and edge $e = \{x, y\}$ in $G = (V, E)$. Furthermore, let N be the maximum number of vertex disjoint paths between u and v in G, and let N' be the maximum number of vertex disjoint paths between u and v in $G/\{x, y\}$. Then we have:*

$$\{x, y\} \cap \{u, v\} = \emptyset \Longrightarrow N' \leq N$$

Proof: Let a_e be the new vertex created by contracting edge e. We consider a set P' of vertex disjoint paths $p_1, ..., p_{N'}$ between u and v in G/e. Since these paths are vertex disjoint and $\{x, y\} \cap \{u, v\} = \emptyset$, there can be at most one path p' in P' going through the new vertex a_e, i.e. a_e is contained in at most one path p' of P'.

One easily sees that there is a path p in G between u and v that uses all vertices of p' except a_e and x and/or y. Therefore, we have a set P of N' vertex disjoint paths between u and v in G. Hence, $N' \leq N$. □

In other words, the number of vertex disjoint paths between u and v can be increased by an edge contraction, only if an edge incident to u or v is contracted. A consequence of this is that after contracting edge e which results in a new vertex a_e, we only have to look for the number of vertices disjoint paths of pairs of vertices that contain a_e. This results in a drastic speed up compared to the case when checking all pairs of vertices for $L + 1$ vertex disjoint paths, as we check $O(n)$ pairs instead of $\Theta(n^2)$ pairs. However, once we have found an improvement edge in the graph, we then must check all other pairs, as possibly, after an improvement edge is added, pairs of vertices that do not contain a_e can have $L + 1$ vertex disjoint paths.

5.3.2 LBN+(MD) versus LBN+(MMD)

We now compare the LBN+(MD) heuristic with the LBN+(MMD) heuristic, and similarly, the LBP+(MD) heuristic with the LBP+(MMD) heuristic. We show that these output the same lower bound result. I.e. in the LBN+(MD) heuristic, we just use the minimum degree of a vertex in H as lower bound, while in the LBN+(MMD) heuristic, we compute the degeneracy.

The intuitive idea is that LBN+(MD) and LBN+(MMD) compute the same result, because due to the additional contraction step, the subgraphs considered by the MMD lower bound, will also be considered in the algorithm LBN+(MD); similarly for the version with paths improvement. Below, we show that this intuition is correct. However, before that, we give one lemma.

Lemma 5 (See [33].) *Let be given a graph $G = (V, E)$, vertex $v \in V$ and edge $e \in E$. Furthermore, let a_e be the resulting new vertex after contracting edge e. If $v \notin e$, then $d_{G/e}(v) \geq d_G(v)-1$. If $v \in e$, then $d_{G/e}(a_e) \geq d_G(v) - 1$.*

Lemma 6 *Let $G = (V, E)$ be a graph. Let the result of running the LBN+(MD), LBN+(MMD), LBP+(MD) and LBP+(MMD) algorithms on G be respectively α_n, β_n, α_p and β_p. Then $\alpha_n = \beta_n$ and $\alpha_p = \beta_p$.*

Proof: The proof is the same for the versions with neighbours and paths improvement. Thus, in the proof below, we write LBX+(MD) and LBX+(MMD), where the X can stand for N or P, and we drop the subscripts n and p from α and β.

First note that when the LBX+(MD) and LBX+(MMD) enter the loop at step three with the same value of L, then they will work with the same graph H. Thus, we have that $\alpha \leq \beta$: when LBX+(MD) increases L by one, we have that L is smaller than the minimum degree of H, hence also smaller than the degeneracy of H, and hence the LBX+(MMD) algorithm will also increase L by one at the corresponding point during the execution of the algorithms. To show equality, we assume the following, and we will derive a contradiction:

$$\alpha < \beta \tag{1}$$

Consider the moments step 2 is done by algorithm LBX+(MMD) and by algorithm LBX+(MD) when $L = \alpha$. As LBX+(MD) outputs α, this is the last time step 2 is done by the LBX+(MD) algorithm, while the LBX+(MMD) algorithm will increase L further (as $\alpha < \beta$), and hence will execute later the 'goto step 2' command at least once.

Let H^* be the the graph H at the moment the LBX+(MMD) algorithm is at step 7 and 8 when the algorithm increases L from α to $\alpha + 1$. This graph H^* is formed from G by a sequence of contractions and $(\alpha + 1)$-neighbours or $(\alpha + 1)$-paths improvement steps. As the test in step 6 was true, the degeneracy of H^* is at least $\alpha + 1$.

The LBX+(MD) algorithm has started a run of the main iteration with $L = \alpha$. As the algorithm outputs α, this is its last iteration. During this iteration, it does the same improvement steps as the LBX+(MMD) algorithm, and hence, at some point, creates the graph H^*. However, it cannot execute steps 7 and 8 now, so the test in step 6 was false for the LBX+(MD) algorithms. Thus, we have:

$$\delta(H^*) \leq \alpha < \delta D(H^*)$$

Write $d = \delta D(H^*)$. Therefore, there exists an induced subgraph $H' \subset H^*$ with

$$\delta(H^*) < \delta(H') = d$$

Note that all vertices in $V(H')$ have degree at least $d := \delta D(H^*)$ in H^*. We now consider the execution of LBX+(MD), starting when H is the graph H^*, up to just before the point that the first vertex from H' is selected as minimum degree vertex v in step 9. During this part of the execution, we have that H' is a subgraph of the graph H used by the algorithm: improvement steps only add edges, and no edges between vertices in H' are contracted.

Now, consider the first vertex v^* from H' that is selected as minimum degree vertex v in step 9 by LBX+(MD). As H' is a subgraph of the graph H, we have that the degree of v^* at the moment

it is selected is at least its degree in H', which is at least d. But, as v^* is the minimum degree of a vertex in H, all vertices in H have at this point degree at least d. This gives a contradiction: consider the test at step 6 just before v^* was selected: the minimum degree of H is at least d, which is larger than the current value of L, i.e. α. So, this test is true, and the algorithm will increase L, contradiction.

So, we can conclude that the assumption $\alpha < \beta$ is false, hence $\alpha = \beta$. □

Whether in practice LBX+(MD) would be more time-efficient than LBX+(MMD) is unclear from the above. The lower bound MMD is more time consuming than MD, but can result in a $b > L$ earlier during the contraction process, by this avoiding a number of graph improvement steps. By Lemma 6, the number of graph improvement steps in the last iteration will be equal, slowing down the algorithm on this point.

Similarly LBX+(MMD+) can be more time-efficient than LBX+(MD): MMD+ is more time consuming but can reduce the number of improvement steps. Moreover, LBX+(MMD+) can return a better bound than LBX+(MD), although this rarely happens. Experimental results with LBX+(MD), LBX+(MMD), and LBX+(MMD+) have shown that the computation times of LBX+(MD) are significantly smaller than those for LBX+(MMD) which on their turn are significantly smaller than those.for LBX+(MMD+).

6 Experimental Results

In this section, we report on the results of computational experiments we have carried out. We tested our algorithms on a number of graphs. The first set of instances are probabilistic networks from existing decision support systems from fields like medicine and agriculture. Central to the use of these networks is to solve the probabilistic inference problem. One of the most used methods for probabilistic inference is the following: one constructs the so-called moralised graph from the probabilistic network. After this (simple) step, one builds a tree decomposition of the moralized graph, and then uses this tree decomposition to solve the probabilistic inference problem. The time for the last step is exponential in the width of the tree decomposition, but linear in the number of nodes. Thus, computing the treewidth of these moralized graphs is of great practical use. The second set of instances are from frequency assignment problems from the EUCLID CALMA project. In [19, 21], tree decompositions were used to solve the frequency assignment problem on many of the networks from this collection of instances. In addition, we use versions of the network, obtained by preprocessing [8]. We have also used these sets of instances in earlier experiments. A third set of instances are taken from the work of Cook and Seymour [13]. Here, they present a heuristic for the travelling salesman problem where they use branch decompositions (a notion strongly related to tree decompositions) of graphs formed by merging a number of TSP-tours. Finally, we computed the lower bounds for many of the DIMACS colouring instances [15]. Among all these, we excluded those networks for which the MMD heuristic already gives the exact treewidth. Some of the graphs can be obtained from [31]. 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.

Tables 2, 3 and 4 give the results for some selected instances, whose behaviour is typical for the entire set of instances. In the appendix, we give longer tables for the entire collection of graphs we tested our algorithms on. In Table 2, we included the best known upper bound (UB) for comparison [20, 11]. For the six variants of the MCSLB+ algorithm we give for space reasons only the average time. There were no large differences in the running times between the different MCSLB+ heuristics.

We can see from these results that contraction is a very useful method for obtaining lower bounds for treewidth. The improvements obtained by using MMD+ instead of MMD, or MCSLB+ instead of MCSLB are in many cases quite significant.

Concerning the different strategies for MMD+, we can observe that the least-c strategy is best. In many cases, it performs much better than the other two strategies, and in all our experiments,

there is only one case where its bound is one smaller than that obtained with the min-d strategy. The max-d strategy appears to do bad, giving in general much smaller lower bounds than the other two. Thus, we did not use this strategy for the other heuristics.

For the MCSLB+, again the least-c strategy seems to be better than the min-d strategy. We observe that min-deg and last-mcs for selecting the contraction vertex, combined with least-c outperform the other strategies. The differences between MMD+(least-c) and MCSLB+(least-c) are usually small, but in a few cases, the MCSLB+(least-c) gives a significant larger bound. The time of these heuristic is often much larger than that of the MMD+ heuristics.

The LBN and LBP strategies appear to be usually preferable to the MCSLB-based ones. The time of these on improvement based strategies is often much smaller (except for LBP+(MD)), while the bounds are at least as good. For the instances from probabilistic networks and frequency assignment, the LBN+(MD) and LBP+(MD) algorithm give often rather significant increases to the lower bounds, but often at the cost of more time use. The situation for the TSP-instances is interesting. Here, the LBN(MMD+) and LBP+(MD) algorithms seem to give the best tradeoff between lower bound and running time. The LBP+(MD) algorithm appears to use very much time on these instances. A few cases could not be run to completion due to the large time used; others give a result only after several many hours of computation time.

We can also observe that in a few cases (e.g. pignet2), the LBN+(MD) algorithm performs faster than the LBN(MMD). This can be explained by the fact that the LBN+(MD) and LBP+(MD) algorithms start with the lower bound value given by the MMD+ algorithm, while the LBN(MMD) and the LBP(MMD) algorithms start with the value provided by the MMD algorithm. Thus, these latter algorithms have more rounds, and as often many improvements are possible for small values of k, the earlier rounds are often more time consuming. Hence, for speeding up LBN, LBP, LBN+ or LBP+ based heuristics, one should start with a good start value of the lower bound. For instance, one might want to start an LBP+(MD) heuristic with a lower bound obtained by an LBN+(MD) heuristic.

For the class of instances derived from the work of Cook and Seymour [13], our heuristics seem not well suited. This can be explained as follows. For planar graphs, the contraction degeneracy is at most five (as planar graphs and hence minors of planar graphs have always a vertex of degree at most five). The TSP-instances can be expected to be close to planar; thus one can expect the MMD+ based heuristics not to do well on such instances in general.

Overall, for 31 out of 155 graphs, the best lower bound computed equals the best known upper bound, for many other instances the remaining gap is very small. From these results, we can conclude that combining existing methods with contraction can give considerable improvements of treewidth lower bounds. The MMD+(least-c) appears to be a good algorithm with often (almost) negligible running times and good bounds; better bounds can be obtained by slower algorithms, like the LBN(MMD+) or the LBP+(MD).

instance	size		UB	MMD		MMD+					
						min-d		max-d		least-c	
	$\|V\|$	$\|E\|$		LB	CPU	LB	CPU	LB	CPU	LB	CPU
link	724	1738	13	4	0.00	8	0.02	5	0.01	11	0.03
munin1	189	366	11	4	0.00	8	0.01	5	0.00	10	0.00
munin3	1044	1745	7	3	0.01	7	0.01	4	0.02	7	0.02
pignet2	3032	7264	135	4	0.01	29	0.11	10	0.07	38	0.20
celar06	100	350	11	10	0.00	11	0.00	10	0.00	11	0.01
celar07pp	162	764	18	11	0.00	13	0.00	12	0.00	15	0.01
graph04	200	734	55	6	0.00	12	0.01	7	0.00	19	0.02
rl5934-pp	904	1800	23	3	0.01	5	0.02	4	0.02	5	0.03
queen15-15	225	5180	171	42	0.00	52	0.07	42	0.02	58	0.19

Table 2: Upper bounds and results of MMD and MMD+ for selected instances

instance	MCSLB		MCSLB+ LBs						
			min-deg		last-mcs		max-mcs		average
	LB	CPU	min-d	least-c	min-d	least-c	min-d	least-c	CPU
link	5	3.09	8	10	8	11	8	6	43.08
munin1	4	0.17	8	10	9	10	9	7	0.95
munin3	4	5.87	6	7	7	7	6	7	33.80
pignet2	5	59.60	28	39	30	39	16	18	509.60
celar06	11	0.06	11	11	11	11	11	11	0.33
celar07pp	12	0.16	14	15	13	15	13	15	1.22
graph04	8	0.25	12	20	13	20	14	16	1.95
rl5934-pp	4	8.27	5	6	5	6	5	6	35.98
queen15-15	42	1.80	52	59	52	62	52	58	27.79

Table 3: Results of MCSLB+ for selected instances

instance	LBN (MMD)		LBN (MMD+)		LBN+ (MD)		LBP (MMD)		LBP (MMD+)		LBP+ (MD)	
	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU
link	4	0.06	11	0.08	11	0.40	4	0.12	11	0.16	12	40.70
munin1	4	0.01	10	0.00	10	0.03	4	0.01	10	0.02	10	0.16
munin3	3	0.21	7	0.06	7	0.53	3	0.43	7	0.12	7	31.31
pignet2	6	79.97	38	0.60	41	21.58	6	87.94	38	0.63	48	1280.96
celar06	10	0.00	11	0.02	11	0.02	10	0.00	11	0.01	11	0.12
celar07pp	13	0.25	15	0.04	15	0.06	13	0.22	15	0.04	16	2.11
graph04	6	0.02	19	0.07	21	0.34	6	0.03	19	0.07	24	4.34
rl5934-pp	3	0.52	5	0.12	5	0.25	3	0.55	5	0.11	9	38137.20
queen15-15	42	0.19	58	0.78	60	7.36	42	0.10	58	0.41	73	7579.01

Table 4: Results of LBN+/LBP+ for selected instances

7 Discussion and Concluding Remarks

In this article, we examined the notion of contraction degeneracy, and several heuristics for treewidth lower bounds which are based on the combination of contraction with existing treewidth lower bound methods. We showed some corresponding decision problems to be NP-complete, but also introduced several heuristics.

The practical experiments show that contracting edges is a very good approach for obtaining lower bounds for treewidth as it considerably improves known lower bounds. The MMD+ heuristics appear to be attractive, due to the fact that the running time of these heuristics is almost always negligible, and the bound is reasonably good. The MCSLB+ heuristics have much larger running time, and often give only a small improvement on the MMD+ based lower bound. The LBN, LBP, LBN+ and LBP+ heuristics often use more time than the MMD+, but less than the MCSLB+ (except for LBP+(MD)), and can give further lower bound improvements. The LBP+(MD) heuristic usually is slowest but gives often the best results. Furthermore, we see that the strategy for selecting a neighbour u of v with the least number of common neighbours of u and v often performs best and appears to be the clear choice for such a strategy.

Notice that although the gap between lower and upper bound could be significantly closed by contracting edges within the algorithms, the absolute gap is still large for many graphs (pignet2, graph*). While it is known that the treewidth has polynomial time approximation algorithm with logarithmic performance ratios, the existence of polynomial time approximation algorithms for treewidth with constant bounded ratios remains a long standing open problem. Thus, obtaining good lower bounds for treewidth is both from a theoretical as from a practical viewpoint a highly interesting topic for further research.

A different lower bound for treewidth was provided by Ramachandramurthi [24, 25]. While this lower bound appears to generally give small lower bound values, it can also be combined with contraction. Work in this direction is in progress.

Apart from its function as a treewidth lower bound, the contraction degeneracy appears to be an attractive and elementary graph measure, worth further study. For instance, interesting topics are its computational complexity on special graph classes or the complexity of approximation algorithms with a guaranteed performance ratio. We recently found a polynomial time algorithm for cographs [9], and we observed that the problem is easy for chordal graphs [33], but many other cases remain open.

Acknowledgements

We would like to thank Dieter Kratsch for useful discussions, and also the anonymous referees of an extended abstract of this article.

References

[1] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Alg. Disc. Meth.*, 8:277–284, 1987.

[2] S. Arnborg and A. Proskurowski. Characterization and recognition of partial 3-trees. *SIAM J. Alg. Disc. Meth.*, 7:305–314, 1986.

[3] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25:1305–1317, 1996.

[4] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comp. Sc.*, 209:1–45, 1998.

[5] H. L. Bodlaender. Necessary edges in k-chordalizations of graphs. *Journal of Combinatorial Optimization*, 7:283–290, 2003.

[6] 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.

[7] 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.

[8] H. L. Bodlaender, A. M. C. A. Koster, F. van den Eijkhof, and L. C. van der Gaag. Preprocessing for triangulation of probabilistic networks. In J. Breese and D. Koller, editors, *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pages 32–39, San Francisco, 2001. Morgan Kaufmann.

[9] 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.

[10] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. American Elsevier, MacMillan, New York, London, 1976.

[11] 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.

[12] 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.

[13] W. Cook and P. D. Seymour. Tour merging via branch-decomposition. *Informs J. on Computing*, 15(3):233–248, 2003.

[14] B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.

[15] The second DIMACS implementation challenge: NP-Hard Problems: Maximum Clique, Graph Coloring, and Satisfiability. See http://dimacs.rutgers.edu/Challenges/, 1992–1993.

[16] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1998.

[17] 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.

[18] V. Gogate and R. Dechter. A complete anytime algorithm for treewidth. To appear in proceedings UAI'04, Uncertainty in Artificial Intelligence, 2004.

[19] A. M. C. A. Koster. *Frequency Assignment - Models and Algorithms*. PhD thesis, Univ. Maastricht, Maastricht, the Netherlands, 1999.

[20] 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.

[21] 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.

[22] 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.

[23] B. Lucena. A new lower bound for tree-width using maximum cardinality search. *SIAM J. Disc. Math.*, 16:345–353, 2003.

[24] S. Ramachandramurthi. A lower bound for treewidth and its consequences. In E. W. Mayr, G. Schmidt, and G. Tinhofer, editors, *Proceedings 20th International Workshop on Graph Theoretic Concepts in Computer Science WG'94*, pages 14–25. Springer Verlag, Lecture Notes in Computer Science, vol. 903, 1995.

[25] S. Ramachandramurthi. The structure and number of obstructions to treewidth. *SIAM J. Disc. Math.*, 10:146–157, 1997.

[26] N. Robertson and P. D. Seymour. Graph minors. XX. Wagner's conjecture. To appear.

[27] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory Series B*, 63:65–110, 1995.

[28] N. Robertson, P. D. Seymour, and R. Thomas. Quickly excluding a planar graph. *J. Comb. Theory Series B*, 62:323–348, 1994.

[29] K. Shoikhet and D. Geiger. A practical algorithm for finding optimal triangulations. In *Proc. National Conference on Artificial Intelligence (AAAI '97)*, pages 185–190. Morgan Kaufmann, 1997.

[30] R. E. Tarjan and M. Yannakakis. Simple linear time algorithms to test chordiality of graphs, test acyclicity of graphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13:566–579, 1984.

[31] Treewidthlib. http://www.cs.uu.nl/people/hansb/treewidthlib, 2004-03-31.

[32] F. van den 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.

[33] 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 Netherland, 2004.

A All Computational Results

Below, we present the results of our experiments. Earlier, we presented a number of selected results. See Section 6 for details on the backgrounds of the graphs and the experimental setting.

instance	size		UB	MMD		MMD+					
						min-d		max-d		least-c	
	$\|V\|$	$\|E\|$		LB	CPU	LB	CPU	LB	CPU	LB	CPU
barley	48	126	7	5	0.00	6	0.00	5	0.00	6	0.00
diabetes	413	819	4	3	0.00	4	0.01	4	0.00	4	0.00
link	724	1738	13	4	0.00	8	0.02	5	0.01	11	0.03
mildew	35	80	4	3	0.00	4	0.00	3	0.00	4	0.00
munin1	189	366	11	4	0.00	8	0.01	5	0.00	10	0.00
munin2	1003	1662	7	3	0.01	6	0.01	4	0.01	6	0.02
munin3	1044	1745	7	3	0.01	7	0.01	4	0.02	7	0.02
munin4	1041	1843	8	4	0.01	7	0.01	5	0.01	7	0.02
oesoca+	67	208	11	9	0.00	9	0.00	9	0.00	9	0.00
oow-trad	33	72	6	3	0.00	4	0.00	4	0.00	5	0.00
oow-bas	27	54	4	3	0.00	4	0.00	3	0.00	4	0.00
oow-solo	40	87	6	3	0.00	4	0.00	4	0.00	5	0.00
pathfinder	109	211	6	5	0.00	6	0.00	5	0.00	6	0.01
pignet2	3032	7264	135	4	0.01	29	0.11	10	0.07	38	0.20
pigs	441	806	10	3	0.00	6	0.01	4	0.00	7	0.01
ship-ship	50	114	8	4	0.00	6	0.00	4	0.00	6	0.00
water	32	123	10	6	0.00	7	0.00	7	0.00	8	0.00
wilson	21	27	3	2	0.00	3	0.00	3	0.00	3	0.00
barley-pp	26	78	7	5	0.00	6	0.00	5	0.00	6	0.00
link-pp	308	1158	13	6	0.00	8	0.01	6	0.00	11	0.02
munin1-pp	66	188	11	4	0.00	8	0.00	5	0.00	10	0.01
munin2-pp	167	455	7	4	0.00	6	0.01	5	0.00	6	0.01
munin3-pp	96	313	7	4	0.00	7	0.00	5	0.00	7	0.01
munin4-pp	217	646	8	5	0.00	7	0.00	5	0.01	8	0.01
munin-kgo-pp	16	41	5	4	0.00	4	0.00	5	0.00	4	0.00
oesoca+-pp	14	75	11	9	0.00	10	0.00	9	0.00	9	0.00
oow-trad-pp	23	54	6	4	0.00	5	0.00	4	0.00	5	0.00
oow-solo-pp	27	63	6	4	0.00	5	0.00	4	0.00	5	0.00
pathfinder-pp	12	43	6	5	0.00	6	0.00	5	0.00	6	0.00
pignet2-pp	1024	3774	142	5	0.01	29	0.09	10	0.03	38	0.13
pigs-pp	48	137	10	4	0.00	7	0.00	4	0.00	7	0.00
ship-ship-pp	30	77	8	4	0.00	6	0.00	4	0.00	6	0.00
water-pp	22	96	10	6	0.00	8	0.00	7	0.00	8	0.00

Table 5: Upper bounds and results of MMD and MMD+ for (preprocessed versions of) probabilistic networks

instance	size		UB	MMD		MMD+					
						min-d		max-d		least-c	
	$\|V\|$	$\|E\|$		LB	CPU	LB	CPU	LB	CPU	LB	CPU
celar01	458	1449	17	8	0.00	12	0.01	9	0.00	14	0.02
celar02	100	311	10	9	0.00	9	0.00	9	0.00	10	0.00
celar03	200	721	15	8	0.00	11	0.00	9	0.00	13	0.01
celar04	340	1009	16	9	0.00	12	0.01	9	0.00	13	0.01
celar05	200	681	15	9	0.01	11	0.00	9	0.00	13	0.01
celar06	100	350	11	10	0.00	11	0.00	10	0.00	11	0.01
celar06pp	82	327	11	10	0.00	11	0.00	10	0.00	11	0.01
celar07	200	817	18	11	0.00	13	0.00	12	0.01	15	0.00
celar07pp	162	764	18	11	0.00	13	0.00	12	0.00	15	0.01
celar08	458	1655	18	11	0.00	13	0.01	12	0.01	15	0.01
celar08pp	365	1539	18	11	0.00	13	0.00	12	0.01	15	0.01
celar09	340	1130	18	11	0.01	13	0.01	12	0.00	15	0.01
celar10	340	1130	18	11	0.00	13	0.01	12	0.01	15	0.01
celar11	340	975	15	8	0.00	11	0.00	9	0.00	13	0.01
graph01	100	358	25	8	0.00	9	0.01	9	0.00	14	0.00
graph02	200	709	51	6	0.00	11	0.00	7	0.01	20	0.02
graph03	100	340	22	5	0.00	8	0.00	6	0.01	14	0.00
graph04	200	734	55	6	0.00	12	0.01	7	0.00	19	0.02
graph05	100	416	26	8	0.00	9	0.00	9	0.00	15	0.00
graph06	200	843	53	8	0.00	12	0.01	9	0.00	21	0.02
graph06pp	119	348	18	5	0.00	7	0.00	6	0.00	11	0.01
graph07	200	843	53	8	0.00	12	0.01	9	0.00	21	0.02
graph08	340	1234	91	7	0.00	16	0.02	8	0.01	26	0.04
graph09	458	1667	118	8	0.01	17	0.03	9	0.01	29	0.06
graph10	340	1275	96	6	0.00	15	0.01	7	0.01	27	0.04
graph11	340	1425	98	7	0.00	17	0.01	8	0.01	27	0.05
graph11pp	340	1424	98	7	0.00	16	0.02	8	0.01	28	0.05
graph12	340	1256	90	5	0.00	16	0.01	6	0.01	25	0.04
graph13	458	1877	126	6	0.00	18	0.02	7	0.01	31	0.07
graph13pp	456	1874	134	6	0.01	18	0.03	7	0.01	31	0.07
graph14	458	1398	121	4	0.00	20	0.02	8	0.02	27	0.05

Table 6: Upper bounds and results of MMD and MMD+ for frequency assignment instances

instance	size		UB	MMD		MMD+					
						min-d		max-d		least-c	
	$\|V\|$	$\|E\|$		LB	CPU	LB	CPU	LB	CPU	LB	CPU
celar01-pp	157	804	16	8	0.00	12	0.00	9	0.00	14	0.01
celar02-pp	19	115	10	9	0.00	9	0.00	9	0.00	10	0.00
celar03-pp	81	413	15	9	0.00	11	0.01	10	0.00	13	0.00
celar04-pp	114	524	16	9	0.00	12	0.00	10	0.00	13	0.01
celar05-pp	80	426	15	9	0.00	12	0.00	9	0.01	13	0.00
celar07-pp	92	521	18	11	0.00	13	0.01	11	0.00	15	0.00
celar08-pp	189	1016	18	11	0.00	13	0.01	12	0.00	15	0.01
celar09-pp	133	646	18	11	0.00	13	0.00	11	0.00	15	0.01
celar10-pp	133	646	18	11	0.01	13	0.01	11	0.00	15	0.01
celar11-pp	96	470	15	9	0.00	11	0.00	9	0.01	13	0.00
graph01-pp	89	332	26	8	0.00	10	0.00	9	0.01	15	0.00
graph02-pp	179	659	48	6	0.00	12	0.01	7	0.01	20	0.01
graph03-pp	79	293	24	6	0.00	8	0.00	6	0.01	13	0.00
graph04-pp	179	678	53	6	0.01	12	0.00	7	0.01	19	0.01
graph05-pp	91	394	26	8	0.00	9	0.01	9	0.00	15	0.01
graph06-pp	180	790	54	8	0.00	13	0.01	9	0.00	21	0.02
graph07-pp	180	790	54	8	0.00	13	0.01	9	0.00	21	0.02
graph08-pp	314	1173	90	7	0.01	16	0.02	8	0.01	26	0.04
graph09-pp	405	1525	121	8	0.00	18	0.04	9	0.02	29	0.08
graph10-pp	328	1253	97	6	0.00	16	0.03	7	0.01	26	0.05
graph11-pp	307	1338	93	7	0.00	16	0.02	8	0.01	28	0.04
graph12-pp	312	1177	87	6	0.00	16	0.02	7	0.01	25	0.03
graph13-pp	420	1772	134	6	0.01	19	0.03	7	0.01	31	0.07
graph14-pp	395	1325	127	5	0.00	20	0.02	8	0.02	27	0.04

Table 7: Upper bounds and results of MMD and MMD+ for preprocessed versions of frequency assignment instances

instance	size		UB	MMD		MMD+					
						min-d		max-d		least-c	
	$\|V\|$	$\|E\|$		LB	CPU	LB	CPU	LB	CPU	LB	CPU
fl3795	2103	3973	13	3	0.01	5	0.09	4	0.06	5	0.10
fnl4461	3326	5147	35	3	0.02	5	0.10	4	0.08	5	0.13
pcb3038	1985	3109	25	2	0.01	5	0.05	4	0.04	5	0.07
rl5915	1939	2935	25	3	0.01	5	0.06	4	0.04	5	0.06
rl5934	2048	3087	26	3	0.01	5	0.05	4	0.04	5	0.07
fl3795-pp	1433	3098	13	3	0.01	5	0.04	4	0.05	6	0.05
fnl4461-pp	1528	3114	33	3	0.01	5	0.04	4	0.03	5	0.06
pcb3038-pp	948	1920	25	3	0.01	5	0.03	4	0.02	5	0.03
rl5915-pp	863	1730	23	3	0.00	5	0.02	4	0.02	5	0.03
rl5934-pp	904	1800	23	3	0.01	5	0.02	4	0.02	5	0.03

Table 8: Upper bounds and results of MMD and MMD+ for TSP graphs and preprocessed versions of these

instance	size		UB	MMD		MMD+					
						min-d		max-d		least-c	
	$\|V\|$	$\|E\|$		LB	CPU	LB	CPU	LB	CPU	LB	CPU
anna	138	493	12	10	0.00	11	0.01	10	0.00	11	0.01
david	87	406	13	10	0.00	10	0.00	10	0.00	12	0.00
fpsol2.i.1	269	11654	66	64	0.01	66	0.05	64	0.04	66	0.34
games120	120	638	33	8	0.00	12	0.01	9	0.01	19	0.01
homer	556	1628	31	12	0.01	19	0.01	12	0.01	22	0.02
inithx.i.1	519	18707	56	55	0.01	55	0.11	55	0.09	56	0.72
inithx.i.2	558	13979	35	31	0.01	31	0.10	31	0.08	31	0.54
inithx.i.3	559	13969	35	31	0.01	31	0.14	31	0.08	31	0.53
le450-15a	450	8168	272	24	0.01	59	0.15	27	0.03	73	0.45
le450-15b	450	8169	270	24	0.01	59	0.15	27	0.04	75	0.45
le450-15c	450	16680	359	49	0.01	98	0.30	51	0.08	109	1.22
le450-15d	450	16750	360	51	0.00	97	0.30	52	0.07	109	1.14
le450-25a	450	8260	234	26	0.00	56	0.12	29	0.04	75	0.42
le450-25b	450	8263	233	25	0.00	53	0.12	28	0.04	74	0.42
le450-25c	450	17343	327	52	0.01	95	0.28	54	0.08	111	1.14
le450-25d	450	17425	336	51	0.01	97	0.29	53	0.08	111	1.17
le450-5a	450	5714	256	17	0.01	53	0.13	19	0.03	62	0.31
le450-5b	450	5734	254	17	0.00	53	0.13	19	0.03	62	0.32
le450-5c	450	9803	272	33	0.00	74	0.22	35	0.04	86	0.63
le450-5d	450	9757	278	32	0.01	73	0.22	34	0.04	85	0.62
miles1000	128	3216	49	41	0.00	45	0.01	41	0.01	48	0.06
miles1500	128	5198	77	72	0.00	76	0.03	72	0.01	77	0.10
miles250	125	387	9	7	0.00	8	0.00	7	0.00	9	0.00
miles500	128	1170	22	19	0.00	21	0.00	20	0.01	22	0.02
miles750	128	2113	36	31	0.00	33	0.01	31	0.01	34	0.04
mulsol.i.1	138	3925	50	48	0.00	50	0.02	49	0.01	50	0.06
mulsol.i.2	173	3885	32	31	0.00	31	0.01	31	0.02	32	0.06
mulsol.i.3	174	3916	32	31	0.00	32	0.03	31	0.02	32	0.06
mulsol.i.4	175	3946	32	31	0.00	31	0.01	31	0.02	32	0.07
mulsol.i.5	176	3973	31	31	0.00	31	0.01	31	0.02	31	0.06
myciel3	11	20	5	3	0.00	5	0.00	3	0.00	5	0.00
myciel4	23	71	10	5	0.00	8	0.00	5	0.00	8	0.00
myciel5	47	236	19	8	0.00	14	0.01	8	0.00	14	0.00
myciel6	95	755	35	12	0.00	24	0.00	13	0.01	26	0.01
myciel7	191	2360	66	18	0.00	40	0.03	19	0.01	42	0.07
queen10-10	100	1470	72	27	0.00	31	0.02	27	0.01	35	0.03
queen11-11	121	1980	88	30	0.00	34	0.03	30	0.01	38	0.04
queen12-12	144	2596	104	33	0.00	39	0.03	33	0.01	44	0.07
queen13-13	169	3328	122	36	0.01	42	0.05	36	0.01	48	0.10
queen14-14	196	4186	141	39	0.00	48	0.06	39	0.01	53	0.14
queen15-15	225	5180	163	42	0.00	52	0.07	42	0.02	58	0.19
queen16-16	256	6320	186	45	0.01	56	0.11	45	0.02	63	0.26
queen5-5	25	160	18	12	0.00	12	0.00	12	0.00	12	0.00
queen6-6	36	290	25	15	0.00	15	0.00	15	0.00	15	0.01
queen7-7	49	476	35	18	0.01	18	0.00	18	0.00	20	0.01
queen8-12	96	1368	67	25	0.00	29	0.02	25	0.00	33	0.03
queen8-8	64	728	46	21	0.01	22	0.00	21	0.01	25	0.01
queen9-9	81	1056	58	24	0.00	26	0.02	24	0.00	29	0.02
school1	385	19095	188	73	0.01	97	0.19	74	0.08	122	1.04
school1-nsh	352	14612	162	61	0.01	82	0.13	61	0.05	106	0.74
zeroin.i.1	126	4100	50	48	0.00	50	0.02	48	0.01	50	0.08
zeroin.i.2	157	3541	32	29	0.00	31	0.02	30	0.01	32	0.07
zeroin.i.3	157	3540	32	29	0.00	31	0.02	30	0.01	32	0.07

Table 9: Upper bounds and results of MMD and MMD+ for DIMACS colouring instances

instance	MCSLB		MCSLB+ LBs						
			min-deg		last-mcs		max-mcs		average
	LB	CPU	min-d	least-c	min-d	least-c	min-d	least-c	CPU
barley	5	0.01	6	6	6	6	6	5	0.06
diabetes	4	0.92	4	4	4	4	4	4	5.23
link	5	3.09	8	10	8	11	8	6	43.08
mildew	3	0.00	4	4	4	4	4	4	0.03
munin1	4	0.17	8	10	9	10	9	7	0.95
munin2	4	5.46	6	6	6	6	5	6	31.3
munin3	4	5.87	6	7	7	7	6	7	33.8
munin4	5	6.06	7	7	7	8	6	7	48.3
oesoca+	9	0.02	9	9	9	9	9	9	0.15
oow-trad	4	0.00	5	5	5	5	5	4	0.03
oow-bas	3	0.00	4	4	4	4	4	4	0.02
oow-solo	4	0.01	4	5	4	5	5	5	0.05
pathfinder	6	0.05	6	6	6	6	6	6	0.34
pignet2	5	59.60	28	39	30	39	16	18	509.6
pigs	3	1.01	7	7	7	7	6	6	5.12
ship-ship	5	0.01	6	6	6	6	6	6	0.06
water	8	0.00	8	8	8	8	8	8	0.04
wilson	3	0.00	3	3	3	3	3	3	0.01
barley-pp	6	0.00	6	6	6	6	6	6	0.02
link-pp	6	0.60	8	11	8	11	8	9	4.65
munin1-pp	5	0.02	9	10	9	9	9	8	0.14
munin2-pp	5	0.15	6	6	6	6	6	6	0.91
munin3-pp	5	0.05	7	7	7	7	6	6	0.34
munin4-pp	5	0.26	7	7	7	7	7	7	1.7
munin-kgo-pp	5	0.00	5	5	5	5	5	5	0.01
oesoca+-pp	10	0.00	10	10	10	10	10	10	0.01
oow-trad-pp	4	0.00	5	5	5	5	5	5	0.01
oow-solo-pp	5	0.00	5	5	5	5	5	5	0.02
pathfinder-pp	6	0.00	6	6	6	6	6	6	0
pignet2-pp	6	7.59	29	38	29	39	20	21	74.76
pigs-pp	5	0.01	7	7	7	7	7	6	0.07
ship-ship-pp	4	0.00	6	6	6	6	6	6	0.02
water-pp	8	0.00	8	8	8	8	8	8	0.02

Table 10: Results of MCSLB+ for (preprocessed versions of) probabilistic networks

instance	MCSLB		MCSLB+ LBs						
			min-deg		last-mcs		max-mcs		average
	LB	CPU	min-d	least-c	min-d	least-c	min-d	least-c	CPU
celar01	10	1.20	12	13	12	14	12	13	17.08
celar02	9	0.06	9	10	9	10	9	9	0.3
celar03	9	0.23	11	12	11	13	12	12	1.54
celar04	11	0.66	11	13	12	13	12	13	4.85
celar05	9	0.23	12	13	12	13	12	12	1.8
celar06	11	0.06	11	11	11	11	11	11	0.33
celar06pp	11	0.04	11	11	11	11	11	11	0.24
celar07	12	0.24	13	15	13	15	13	15	1.66
celar07pp	12	0.16	14	15	13	15	13	15	1.22
celar08	12	1.45	13	15	14	15	13	15	17.04
celar08pp	12	0.86	14	15	14	15	14	14	8.39
celar09	12	0.82	13	15	14	15	14	15	4.62
celar10	12	0.71	13	15	14	15	14	15	4.77
celar11	10	0.66	11	13	12	13	12	12	4.43
graph01	9	0.06	9	15	11	14	12	13	0.45
graph02	8	0.24	12	19	12	19	14	13	2.54
graph03	6	0.06	9	13	10	14	9	13	0.49
graph04	8	0.25	12	20	13	20	14	16	1.95
graph05	9	0.06	10	15	11	16	11	14	0.47
graph06	9	0.26	12	22	14	22	14	15	2.22
graph06pp	6	0.07	7	11	7	11	7	8	0.47
graph07	9	0.26	12	22	14	22	14	15	2.15
graph08	9	0.76	17	26	18	26	18	20	5.89
graph09	9	1.43	17	30	20	28	22	23	11.51
graph10	8	0.77	18	26	17	26	19	22	6.25
graph11	8	0.80	15	27	18	27	17	26	6.53
graph11pp	8	0.81	16	27	17	28	18	23	7.25
graph12	7	0.76	15	25	16	25	17	21	5.92
graph13	8	1.48	18	32	19	31	20	28	12.89
graph13pp	8	1.46	19	31	19	32	21	17	12.98
graph14	5	1.35	19	28	20	28	21	25	10.11

Table 11: Results of MCSLB+ for frequency assignment instances

instance	MCSLB		MCSLB+ LBs						
			min-deg		last-mcs		max-mcs		average
	LB	CPU	min-d	least-c	min-d	least-c	min-d	least-c	CPU
celar01-pp	10	0.16	11	13	12	13	12	13	1.27
celar02-pp	10	0.00	10	10	10	10	10	10	0.01
celar03-pp	10	0.04	11	13	12	13	13	12	0.29
celar04-pp	11	0.08	11	13	13	13	13	13	0.57
celar05-pp	9	0.04	11	12	12	12	11	12	0.31
celar07-pp	12	0.11	13	15	14	15	14	15	0.4
celar08-pp	12	0.25	14	15	14	15	14	15	1.89
celar09-pp	12	0.12	14	15	14	15	14	15	0.82
celar10-pp	12	0.12	14	15	14	15	14	15	0.82
celar11-pp	10	0.06	12	13	11	12	11	11	0.42
graph01-pp	9	0.05	10	15	11	14	12	14	0.33
graph02-pp	8	0.22	12	19	14	19	14	15	1.46
graph03-pp	6	0.03	9	14	9	13	10	13	0.25
graph04-pp	8	0.20	12	20	13	20	13	17	1.56
graph05-pp	9	0.06	10	15	10	15	12	14	0.37
graph06-pp	9	0.23	12	22	15	21	15	17	1.75
graph07-pp	9	0.23	12	22	15	21	15	17	1.73
graph08-pp	8	0.67	16	26	17	25	19	23	5.15
graph09-pp	9	1.31	18	29	20	30	21	24	9.35
graph10-pp	8	0.76	16	26	18	26	16	22	5.71
graph11-pp	9	0.67	16	27	18	28	18	25	5.51
graph12-pp	7	0.65	16	25	17	25	17	25	4.98
graph13-pp	8	1.31	19	32	20	32	21	30	10.74
graph14-pp	6	1.03	20	28	21	28	21	25	7.92

Table 12: Results of MCSLB+ for preprocessed versions of frequency assignment instances

instance	MCSLB		MCSLB+ LBs						
			min-deg		last-mcs		max-mcs		average
	LB	CPU	min-d	least-c	min-d	least-c	min-d	least-c	CPU
fl3795	4	41.12	5	6	5	6	5	5	160.84
fnl4461	4	101.46	5	5	5	5	5	5	406.3
pcb3038	4	33.02	5	5	5	5	5	5	127.66
rl5915	4	33.39	5	5	5	6	5	5	114.83
rl5934	4	37.70	5	5	5	6	5	5	128.27
fl3795-pp	4	18.52	5	6	5	6	5	5	115.28
fnl4461-pp	4	22.26	5	5	5	5	5	5	124.29
pcb3038-pp	4	7.48	5	5	5	5	5	5	45.56
rl5915-pp	4	9.13	5	5	5	6	5	6	34.57
rl5934-pp	4	8.27	5	6	5	6	5	6	35.98

Table 13: Results of MCSLB+ for TSP graphs and preprocessed versions of these

instance	MCSLB		MCSLB+ LBs						
			min-deg		last-mcs		max-mcs		average
	LB	CPU	min-d	least-c	min-d	least-c	min-d	least-c	CPU
anna	10	0.21	10	12	11	12	11	11	1.66
david	10	0.10	11	12	10	12	10	11	0.79
fpsol2.i.1	66	4.45	66	66	66	66	66	66	79.51
games120	10	0.23	12	20	15	19	14	18	1.58
homer	13	3.54	19	23	20	23	18	17	39.36
inithx.i.1	56	13.62	56	56	56	56	56	56	262.49
inithx.i.2	31	12.35	31	31	31	31	31	31	226.48
inithx.i.3	31	12.17	31	31	31	31	31	31	225.48
le450-15a	28	7.08	60	74	59	73	59	71	131.96
le450-15b	27	6.73	59	74	59	75	60	73	127.27
le450-15c	51	10.50	97	110	98	109	97	105	223.34
le450-15d	53	10.66	98	110	97	110	96	107	251.67
le450-25a	30	6.06	56	75	58	75	59	69	138.04
le450-25b	29	6.22	54	75	55	76	56	69	126.24
le450-25c	55	12.23	97	111	96	111	97	110	262.69
le450-25d	54	11.05	97	111	98	111	98	110	264.69
le450-5a	20	5.26	53	63	52	62	53	62	89.01
le450-5b	20	5.16	51	62	52	62	52	61	90.96
le450-5c	35	7.14	73	86	73	87	74	85	152.89
le450-5d	34	7.39	72	85	73	86	73	84	150.14
miles1000	46	0.60	46	48	46	49	46	48	10.21
miles1500	74	1.00	77	77	77	77	76	76	13.16
miles250	8	0.16	9	9	9	9	8	8	1.49
miles500	21	0.30	22	22	22	22	22	22	2.78
miles750	32	0.44	33	34	33	34	33	34	6.23
mulsol.i.1	50	0.77	50	50	50	50	50	50	10.62
mulsol.i.2	32	0.95	32	32	32	32	32	32	15.92
mulsol.i.3	32	0.99	32	32	32	32	32	32	14.51
mulsol.i.4	32	0.99	32	32	32	32	32	32	15.72
mulsol.i.5	31	1.00	31	31	31	31	31	31	14.53
myciel3	3	0.00	4	4	5	4	4	4	0.00
myciel4	5	0.00	8	8	8	8	8	7	0.03
myciel5	8	0.03	14	14	14	14	14	12	0.23
myciel6	13	0.16	23	25	24	24	23	23	1.70
myciel7	20	0.84	40	43	40	43	39	35	13.05
queen10-10	27	0.25	31	35	30	36	30	32	2.72
queen11-11	30	0.41	34	39	35	40	35	37	4.64
queen12-12	33	0.62	37	44	40	45	39	42	7.58
queen13-13	36	0.91	42	51	45	49	43	47	12.54
queen14-14	39	1.29	46	54	49	55	47	52	18.57
queen15-15	42	1.80	52	59	52	62	52	58	27.79
queen16-16	45	2.48	56	64	57	65	55	63	39.83
queen5-5	12	0.01	12	12	12	12	12	12	0.05
queen6-6	15	0.03	15	15	15	16	15	16	0.15
queen7-7	18	0.04	18	19	18	19	19	19	0.37
queen8-12	25	0.22	30	34	29	33	29	31	2.24
queen8-8	21	0.08	22	25	22	25	22	24	0.73
queen9-9	24	0.15	25	30	27	30	26	28	1.21
school1	85	9.73	97	122	110	118	104	118	228.85
school1-nsh	72	6.70	81	108	92	101	88	105	155.16
zeroin.i.1	50	0.67	50	50	50	50	50	50	8.90
zeroin.i.2	31	0.80	31	32	31	32	31	31	10.91
zeroin.i.3	31	0.78	31	32	31	32	31	31	11.45

Table 14: Results of MCSLB+ for DIMACS colouring instances

instance	LBN (MMD)		LBN (MMD+)		LBN+ (MD)		LBP (MMD)		LBP (MMD+)		LBP+ (MD)	
	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU
barley	5	0.00	6	0.01	6	0.01	5	0.00	6	0.00	6	0.10
diabetes	3	0.04	4	0.04	4	0.13	3	0.06	4	0.06	4	17.89
link	4	0.06	11	0.08	11	0.40	4	0.12	11	0.16	12	40.70
mildew	3	0.00	4	0.01	4	0.00	3	0.00	4	0.01	4	0.05
munin1	4	0.01	10	0.00	10	0.03	4	0.01	10	0.02	10	0.16
munin2	3	0.10	6	0.06	6	0.47	3	0.20	6	0.11	6	54.54
munin3	3	0.21	7	0.06	7	0.53	3	0.43	7	0.12	7	31.31
munin4	4	0.06	7	0.06	8	1.05	4	0.11	7	0.13	8	99.52
oesoca+	9	0.00	9	0.01	10	0.03	9	0.01	9	0.01	10	0.79
oow-trad	3	0.00	5	0.00	5	0.00	3	0.00	5	0.00	5	0.02
oow-bas	3	0.00	4	0.00	4	0.00	3	0.00	4	0.00	4	0.01
oow-solo	3	0.00	5	0.01	5	0.01	3	0.00	5	0.00	5	0.04
pathfinder	5	0.00	6	0.00	6	0.04	5	0.01	6	0.01	6	0.16
pignet2	6	79.97	38	0.60	41	21.58	6	87.94	38	0.63	48	1280.96
pigs	3	0.06	7	0.02	7	0.13	3	0.07	7	0.02	8	9.58
ship-ship	4	0.00	6	0.00	6	0.00	4	0.00	6	0.01	6	0.03
water	6	0.00	8	0.00	9	0.01	6	0.00	8	0.00	9	0.16
wilson	2	0.00	3	0.00	3	0.00	2	0.00	3	0.00	3	0.00
barley-pp	5	0.00	6	0.00	6	0.00	5	0.00	6	0.00	6	0.03
link-pp	6	0.05	11	0.09	11	0.17	6	0.05	11	0.10	12	29.94
munin1-pp	4	0.01	10	0.01	10	0.02	4	0.01	10	0.01	10	0.05
munin2-pp	4	0.02	6	0.03	6	0.05	4	0.02	6	0.02	6	3.16
munin3-pp	4	0.01	7	0.02	7	0.03	4	0.01	7	0.02	7	0.23
munin4-pp	5	0.01	8	0.05	8	0.09	5	0.02	8	0.04	8	3.78
munin-kgo-pp	4	0.00	4	0.00	5	0.00	4	0.00	4	0.00	5	0.03
oesoca+-pp	9	0.00	10	0.00	10	0.00	9	0.00	10	0.00	10	0.00
oow-trad-pp	4	0.00	5	0.01	5	0.00	4	0.00	5	0.00	5	0.02
oow-solo-pp	4	0.00	5	0.00	5	0.00	4	0.00	5	0.01	5	0.02
pathfinder-pp	5	0.00	6	0.00	6	0.00	5	0.00	6	0.00	6	0.00
pignet2-pp	9	36.90	38	0.66	41	5.65	9	38.75	38	0.66	48	288.83
pigs-pp	4	0.00	7	0.00	8	0.02	4	0.00	7	0.00	8	0.03
ship-ship-pp	4	0.00	6	0.00	6	0.00	4	0.00	6	0.00	6	0.02
water-pp	6	0.00	8	0.00	9	0.01	6	0.00	8	0.01	9	0.07

Table 15: Results of LBN+/LBP+ for (preprocessed versions of) probabilistic networks

instance	LBN (MMD)		LBN (MMD+)		LBN+ (MD)		LBP (MMD)		LBP (MMD+)		LBP+ (MD)	
	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU
celar01	10	0.58	14	0.05	14	0.17	10	0.79	14	0.06	15	15.64
celar02	9	0.01	10	0.01	10	0.01	9	0.01	10	0.01	10	0.30
celar03	9	0.16	13	0.03	13	0.06	9	0.15	13	0.03	14	2.59
celar04	10	0.19	13	0.06	14	0.19	10	0.19	13	0.06	15	3.35
celar05	10	0.16	13	0.03	13	0.07	10	0.17	13	0.03	14	2.75
celar06	10	0.00	11	0.02	11	0.02	10	0.00	11	0.01	11	0.12
celar06pp	10	0.01	11	0.03	11	0.02	10	0.00	11	0.01	11	0.12
celar07	13	0.28	15	0.05	16	0.12	13	0.26	15	0.04	16	2.41
celar07pp	13	0.25	15	0.04	15	0.06	13	0.22	15	0.04	16	2.11
celar08	13	1.03	15	0.12	16	0.38	13	1.02	15	0.12	16	39.72
celar08pp	13	0.83	15	0.11	16	0.32	13	0.81	15	0.11	16	32.55
celar09	13	0.44	15	0.07	16	0.22	13	0.45	15	0.07	16	3.92
celar10	13	0.46	15	0.07	16	0.22	13	0.47	15	0.07	16	4.03
celar11	9	0.25	13	0.06	14	0.20	9	0.25	13	0.06	14	2.05
graph01	8	0.00	14	0.03	15	0.07	8	0.00	14	0.02	16	0.66
graph02	6	0.02	20	0.08	21	0.21	6	0.03	20	0.08	24	2.19
graph03	5	0.00	14	0.02	14	0.04	5	0.01	14	0.03	16	0.31
graph04	6	0.02	19	0.07	21	0.34	6	0.03	19	0.07	24	4.34
graph05	8	0.00	15	0.03	16	0.08	8	0.00	15	0.03	18	1.03
graph06	8	0.02	21	0.09	23	0.42	8	0.02	21	0.09	26	6.48
graph06pp	5	0.01	11	0.03	12	0.06	5	0.02	11	0.02	13	0.36
graph07	8	0.02	21	0.10	23	0.42	8	0.02	21	0.09	26	6.74
graph08	7	0.04	26	0.16	27	0.56	7	0.04	26	0.16	32	12.19
graph09	8	0.05	29	0.23	32	1.66	8	0.05	29	0.23	37	32.87
graph10	6	0.06	27	0.17	29	0.79	6	0.06	27	0.18	31	10.78
graph11	7	0.06	27	0.20	30	1.28	7	0.07	27	0.20	34	26.86
graph11pp	7	0.07	28	0.19	30	0.98	7	0.07	28	0.21	34	19.69
graph12	5	0.08	25	0.17	27	0.78	5	0.08	25	0.17	31	15.18
graph13	6	0.14	31	0.29	34	2.06	6	0.14	31	0.30	39	61.54
graph13pp	6	0.13	31	0.31	34	2.04	6	0.13	31	0.33	39	56.50
graph14	4	0.22	27	0.22	30	1.33	4	0.23	27	0.21	34	17.49

Table 16: Results of LBN+/LBP+ for frequency assignment instances

instance	LBN (MMD)		LBN (MMD+)		LBN+ (MD)		LBP (MMD)		LBP (MMD+)		LBP+ (MD)	
	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU
celar01-pp	10	0.38	14	0.05	14	0.06	10	0.35	14	0.04	15	8.87
celar02-pp	10	0.00	10	0.01	10	0.00	10	0.00	10	0.00	10	0.02
celar03-pp	10	0.06	13	0.02	13	0.02	10	0.06	13	0.02	14	0.90
celar04-pp	10	0.07	13	0.03	14	0.07	10	0.06	13	0.03	15	1.57
celar05-pp	10	0.07	13	0.03	13	0.02	10	0.08	13	0.03	14	1.52
celar07-pp	13	0.12	15	0.03	16	0.05	13	0.13	15	0.03	16	0.90
celar08-pp	13	0.42	15	0.06	16	0.15	13	0.40	15	0.06	16	23.85
celar09-pp	13	0.18	15	0.03	16	0.09	13	0.18	15	0.04	16	1.54
celar10-pp	13	0.17	15	0.03	16	0.09	13	0.18	15	0.03	16	1.42
celar11-pp	10	0.14	13	0.03	14	0.06	10	0.14	13	0.03	14	1.13
graph01-pp	8	0.01	15	0.02	15	0.03	8	0.01	15	0.03	16	0.26
graph02-pp	6	0.01	20	0.07	21	0.21	6	0.02	20	0.06	24	1.99
graph03-pp	6	0.00	13	0.03	14	0.07	6	0.01	13	0.02	16	0.47
graph04-pp	6	0.02	19	0.06	21	0.32	6	0.02	19	0.07	24	4.22
graph05-pp	8	0.00	15	0.03	16	0.08	8	0.00	15	0.03	18	0.80
graph06-pp	8	0.01	21	0.09	23	0.39	8	0.01	21	0.09	26	6.40
graph07-pp	8	0.02	21	0.08	23	0.40	8	0.01	21	0.09	26	6.36
graph08-pp	7	0.04	26	0.18	27	0.53	7	0.04	26	0.16	32	11.73
graph09-pp	8	0.05	29	0.22	31	1.18	8	0.05	29	0.22	37	33.81
graph10-pp	6	0.06	26	0.17	29	1.08	6	0.06	26	0.18	31	16.21
graph11-pp	7	0.06	28	0.18	30	0.90	7	0.06	28	0.19	34	20.72
graph12-pp	6	0.05	25	0.16	27	0.76	6	0.05	25	0.16	31	15.34
graph13-pp	6	0.13	31	0.28	34	2.01	6	0.12	31	0.29	39	62.66
graph14-pp	5	0.03	27	0.10	30	1.25	5	0.04	27	0.12	34	16.91

Table 17: Results of LBN+/LBP+ for preprocessed versions of frequency assignment instances

instance	LBN (MMD)		LBN (MMD+)		LBN+ (MD)		LBP (MMD)		LBP (MMD+)		LBP+ (MD)	
	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU
fl3795	3	0.78	5	0.36	3	0.73	5	0.32	7	2.87	-	-
fnl4461	3	0.97	5	0.45	3	0.91	5	0.36	5	2.56	-	-
pcb3038	3	3.65	5	0.21	3	3.39	5	0.19	5	0.88	-	-
rl5915	3	0.79	5	0.21	3	0.70	5	0.20	5	0.87	-	-
rl5934	3	0.99	5	0.24	3	0.69	5	0.21	5	0.99	-	-
fl3795-pp	3	1.31	6	0.18	3	1.32	6	0.22	6	0.51	7	4269.18
fnl4461-pp	3	0.66	5	0.22	3	0.70	5	0.21	5	0.61	-	-
pcb3038-pp	3	0.28	5	0.11	3	0.30	5	0.11	5	0.27	8	34030.60
rl5915-pp	3	0.27	5	0.11	3	0.31	5	0.11	5	0.24	9	54819.50
rl5934-pp	3	0.52	5	0.12	3	0.55	5	0.11	5	0.25	9	38137.20

Table 18: Results of LBN+/LBP+ for TSP graphs and preprocessed versions of these

instance	LBN (MMD)		LBN (MMD+)		LBN+ (MD)		LBP (MMD)		LBP (MMD+)		LBP+ (MD)	
	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU	LB	CPU
anna	10	0.02	11	0.02	12	0.22	10	0.01	11	0.02	12	1.67
david	10	0.00	12	0.02	12	0.06	10	0.01	12	0.02	12	0.61
fpsol2.i.1	66	1.42	66	1.67	66	5.03	66	1.33	66	1.76	66	1810.12
games120	8	0.03	19	0.05	21	0.36	8	0.02	19	0.06	24	2.48
homer	14	1.13	22	0.13	25	3.12	14	1.18	22	0.13	26	88.39
inithx.i.1	56	2.21	56	3.79	56	14.75	56	2.15	56	3.75	56	3937.69
inithx.i.2	31	1.90	31	4.55	31	13.01	31	1.11	31	2.44	31	3510.81
inithx.i.3	31	4.92	31	7.16	31	12.65	31	2.92	31	4.08	31	3223.80
le450-15a	24	0.80	73	2.01	80	48.59	24	0.41	73	0.94	94	33607.00
le450-15b	24	1.70	75	1.94	81	38.40	24	0.88	75	0.93	95	25254.92
le450-15c	49	2.06	109	4.97	118	196.08	49	1.37	109	2.62	139	301507.92
le450-15d	51	4.46	109	4.86	117	186.02	51	2.74	109	2.83	141	468771.49
le450-25a	27	20.09	75	1.93	81	40.36	27	11.61	75	0.96	96	29270.39
le450-25b	25	2.19	74	1.81	81	47.98	25	1.26	74	0.95	96	34398.06
le450-25c	52	1.86	111	4.97	121	228.82	52	1.20	111	2.76	144	818568.19
le450-25d	51	3.88	111	5.14	119	214.68	51	2.62	111	2.79	143	868635.44
le450-5a	17	0.53	62	1.23	66	17.13	17	0.30	62	0.64	79	6149.43
le450-5b	17	0.52	62	1.27	67	20.78	17	0.30	62	0.66	79	7466.67
le450-5c	33	1.09	86	2.53	93	48.27	33	0.69	86	1.32	106	30076.05
le450-5d	32	1.10	85	2.57	92	48.04	32	0.68	85	1.28	106	34573.68
miles1000	44	2.46	48	0.25	49	1.01	44	1.33	48	0.13	49	337.91
miles1500	76	1.55	77	0.62	77	0.63	76	0.82	77	0.35	77	113.30
miles250	8	0.02	9	0.02	9	0.07	8	0.01	9	0.01	9	0.78
miles500	21	0.18	22	0.10	22	0.16	21	0.11	22	0.05	22	14.52
miles750	32	0.17	34	0.14	34	0.35	32	0.09	34	0.07	35	160.10
mulsol.i.1	50	0.23	50	0.30	50	0.75	50	0.12	50	0.15	50	103.11
mulsol.i.2	32	0.17	32	0.33	32	1.10	32	0.08	32	0.17	32	68.10
mulsol.i.3	32	0.18	32	0.34	32	1.10	32	0.09	32	0.18	32	73.24
mulsol.i.4	32	0.17	32	0.34	32	1.13	32	0.09	32	0.17	32	71.90
mulsol.i.5	31	0.08	31	0.34	31	1.16	31	0.05	31	0.18	31	85.24
myciel3	3	0.00	5	0.00	5	0.00	3	0.00	5	0.00	5	0.00
myciel4	5	0.00	8	0.00	8	0.01	5	0.00	8	0.00	9	0.01
myciel5	8	0.01	14	0.02	15	0.07	8	0.00	14	0.01	16	0.25
myciel6	15	0.40	26	0.06	27	0.26	15	0.22	26	0.03	29	1.94
myciel7	25	5.76	42	0.29	46	3.39	25	3.17	42	0.16	52	174.52
queen10-10	27	0.02	35	0.13	35	0.32	27	0.01	35	0.07	42	29.74
queen11-11	30	0.04	38	0.20	40	1.31	30	0.02	38	0.10	48	119.72
queen12-12	33	0.06	44	0.27	44	0.78	33	0.03	44	0.15	55	363.91
queen13-13	36	0.11	48	0.41	50	3.45	36	0.05	48	0.21	61	1154.18
queen14-14	39	0.14	53	0.56	55	4.90	39	0.07	53	0.30	67	3282.31
queen15-15	42	0.19	58	0.78	60	7.36	42	0.10	58	0.41	73	7579.01
queen16-16	45	0.28	63	1.05	66	13.44	45	0.14	63	0.53	79	16312.83
queen5-5	12	0.00	12	0.00	12	0.01	12	0.00	12	0.00	14	0.04
queen6-6	15	0.00	15	0.01	16	0.05	15	0.00	15	0.00	18	0.35
queen7-7	18	0.01	20	0.03	21	0.09	18	0.00	20	0.01	22	0.69
queen8-12	25	0.02	33	0.11	34	0.46	25	0.01	33	0.06	39	19.54
queen8-8	21	0.01	25	0.04	26	0.18	21	0.01	25	0.03	28	1.71
queen9-9	24	0.01	29	0.08	30	0.31	24	0.01	29	0.04	35	9.75
school1	88	910.26	122	7.87	132	180.80	88	610.36	122	4.46	149	207254.65
school1-nsh	72	461.76	106	4.24	116	112.63	72	312.79	106	2.37	132	139781.41
zeroin.i.1	50	0.30	50	0.35	50	0.78	50	0.15	50	0.19	50	33.40
zeroin.i.2	31	1.18	32	0.51	32	1.05	31	0.62	32	0.27	32	74.14
zeroin.i.3	31	1.25	32	0.53	32	1.05	31	0.62	32	0.28	32	71.42

Table 19: Results of LBN+/LBP+ for DIMACS colouring instances