Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

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

On the Maximum Cardinality Search Lower Bound for Treewidth

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

ZIB-Report 04–45 (November 2004)

On the Maximum Cardinality Search Lower Bound for Treewidth*

Hans L. Bodlaender† Arie M. C. A. Koster‡

Abstract

The Maximum Cardinality Search algorithm visits the vertices of a graph in some order, such that at each step, an unvisited vertex that has the largest number of visited neighbors becomes visited. An MCS-ordering of a graph is an ordering of the vertices that can be generated by the Maximum Cardinality Search algorithm. The visited degree of a vertex v in an MCS-ordering is the number of neighbors of v that are before v in the ordering. The visited degree of an MCS-ordering ψ of G is the maximum visited degree over all vertices v in ψ. The maximum visited degree over all MCS-orderings of graph G is called its *maximum visited degree*. Lucena [14] showed that the treewidth of a graph G is at least its maximum visited degree.

We show that the maximum visited degree is of size $O(\log n)$ for planar graphs, and give examples of planar graphs G with maximum visited degree k with $O(k!)$ vertices, for all $k \in \mathbb{N}$. Given a graph G, it is NP-complete to determine if its maximum visited degree is at least k, for any fixed $k \geq 7$. Also, this problem does not have a polynomial time approximation algorithm with constant ratio, unless P=NP. Variants of the problem are also shown to be NP-complete.

We also propose and experimentally analysed some heuristics for the problem. Several tiebreakers for the MCS algorithm are proposed and evaluated. We also give heuristics that give upper bounds on the value of the maximum visited degree of a graph, which appear to give results close to optimal on many graphs from real life applications.

1 Introduction

Recent research has shown that the notion of treewidth is not only of theoretical interest, but can also be used to solve problems arising from real life applications in practice (see e.g., [11, 13].) One important issue when using treewidth in implementations is the problem

*The research was partly carried out during a visit by the second author to Utrecht University with support from the project *Treewidth and Combinatorial Optimization* with a grant from the Netherlands Organization for Scientific Research NWO. The second author is supported by the DFG research group "Algorithms, Structure, Randomness" (Grant number GR 883/9-3, GR 883/9-4).

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

‡Zuse Institute Berlin (ZIB), Takustraße 7, D-14195 Berlin, Germany. koster@zib.de

Keywords: Maximum Cardinality Search, Treewidth, Lower bounds, Planar graphs, Graph algorithms
Mathematics Subject Classification (2000): 05C85, 68Q25, 68R10

to find tree decompositions of given graphs of optimal or close to optimal width. Many of the theoretical solutions to this problem seem not to be applicable in practice, e.g., some have very large constant factors hidden in the O-notation (like the algorithm from [3], see [15].) Thus, there is a need for practical algorithms for determining the treewidth and finding tree decompositions. Recent investigations brought us preprocessing methods (e.g., [5, 8, 4]), heuristics that often give close to optimal algorithms (e.g., [1, 7, 10]), lower bound methods [6, 12, 7, 9], and some exact methods [2, 9]. Still, in many cases, exact methods are too slow, and there are large gaps between the bounds given by upper bound and lower bound heuristics. This paper concentrates on the study of a lower bound on the treewidth that is due to Lucena [14]. We analyse this bound both theoretically and experimentally.

The lower bound on treewidth of Lucena [14] is based on the Maximum Cardinality Search (or, in short: MCS) algorithm. This algorithm that visits all vertices of a given graph in order was first proposed in 1984 by Tarjan and Yannakakis for the recognition of chordal graphs [16]. The order in which the Maximum Cardinality Search algorithm must visit the vertices of a graph must fulfill the following property: at each point, a vertex must be visited that has the largest number of visited neighbors among all unvisited vertices. Call any ordering of the vertices of the graph $G = (V, E)$ that fulfills this property an MCS-ordering of G. Note that graphs (with more than one vertex) have several different MCS-orderings: often, there will be more than one vertex with the largest number of visited neighbors, and the MCS-ordering can visit any of these vertices next. In particular, any vertex can be the first vertex to visit. The visited degree of a vertex v in an MCS-orderings is the number of neighbors of v that are before v is the ordering, i.e., the number of visited neighbors of v at the moment that v is visited. The visited degree of an MCS-ordering ψ of G is the maximum over all vertices of their visited degree. The maximum visited degree over all MCS-orderings of G is called its maximum visited degree, and denoted by $MCSLB(G)$. Lucena [14] showed that for every graph G and MCS-ordering ψ of G, the maximum visited degree of ψ is at most the treewidth of G.

Thus, Maximum Cardinality Search provides us with a lower bound heuristic for the treewidth of a given graph. We compare the heuristic with other heuristics for treewidth. For instance, the degeneracy of a graph (the maximum over all subgraphs of the minimum degree) is another lower bound of the graph. The degeneracy can be computed faster, but cannot be larger than the visited degree of an MCS-ordering; in several cases it is smaller.

It is interesting to note that Maximum Cardinality Search also has been used as a heuristic for obtaining upper bounds on the treewidth; an experimental evaluation has been reported in [10].

This paper is organised as follows. In Section 2, we give preliminary definitions and results. In Section 3, we study the value of the maximum visited degree for *planar* graphs. We show that this value can be arbitrary large, but also give an $O(\log n)$ upper bound on the maximum visited degree for planar graphs with n vertices. Moreover, the presented examples allow to compare the maximum visited degree with the degeneracy measure. Then, we consider the computational complexity to determine the maximum visited degree of a given graph G. We first show in Section 4 that the problem to decide for a given graph G and integer k, whether $MCSLB(G) \leq k$ is NP-complete, even for fixed $k \geq 7$. Some variants of the problem are also shown to be NP-complete, and we show that it is NP-hard to approximate the maximum visited degree with a constant approximation ratio. We have two types of heuristics for the maximum visited degree. In Section 5 some upper bounds on the maximum visited degree are given. In Section 6, we look at different tiebreakers for constructing MCS-orderings (and thus lower bounds on the maximum visited degree), and report on computational experiments using these tiebreakers. Some final remarks are made in Section 7.

2 Preliminaries

In this section, we give some preliminary definitions and results. All graphs we consider in this paper are undirected, and without parallel edges or self-loops. A graph is denoted $G = (V, E)$ with V the set of vertices and E the set of edges. Unless stated otherwise, $n = |V|$ denotes the number of vertices in the considered graph. The degree of a vertex $v \in V$ in graph G is denoted $d_G(v)$ or $d(v)$.

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

- $\bigcup_{i \in I} X_i = V$,
- for all $\{v, w\} \in E$, there is an $i \in I$ with $v, w \in X_i$, and
- for all $i_0, i_1, i_2 \in I$: if i_1 is on the path from i_0 to i_2 in T, then $X_{i_0} \cap X_{i_2} \subseteq X_{i_1}$.

The *width* of tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i| - 1$. The treewidth of a graph G, $tw(G)$, is the minimum width among all tree decompositions of G.

Consider an ordering $\psi = (v_1, v_2, \ldots, v_n)$ of the vertices of a graph $G = (V, E)$. The *current visited degree of* v_i *at time* j, denoted $cvd_j^{\psi}(v_i)$, is $|\{\{v_k, v_i\} \mid k < j\}|$, i.e., the number of neighbors of v_i in the set $\{v_1, \ldots, v_{j-1}\}$. The ordering ψ is an *MCS-ordering* ('maximum cardinality search ordering') if for each i, v_i has the maximum current visited degree at time i among all vertices in $\{v_i, v_{i+1}, \ldots, v_n\}$, i.e., $cvd_i^{\psi}(v_i) = \max_{i \leq j \leq n} cvd_i^{\psi}(v_j)$. The *visited degree* of v_i is the current visited degree of v_i at time i, and the visited degree of the MCS-ordering ψ, $mcslb_{\psi}(G)$, is the maximum over all vertices of their visited degree. The *maximum visited degree* of G, or also, MCSLB(G) is the maximum visited degree over all MCS-orderings of G.

In several proofs, we look at maximum cardinality search in an 'operational' way, i.e, we follow the steps of an algorithm that constructs an MCS-ordering. At a certain point during the execution of the algorithm, vertices are either already visited, or unvisited (in which case, the algorithm will visit them later.) The current visited degree of an (unvisited) vertex is the number of visited neighbors at that point. So, the visited degree of a vertex is the number of visited neighbors at the moment it is visited. Motivation for our work was the following result.

Theorem 1 (Lucena [14]) *For each graph G, the treewidth of G is at least its maximum visited degree.*

The following easy lemma is used several times in our proofs.

Lemma 2 *Let ψ be an MCS-ordering of G and suppose v has visited degree k in ψ. Then v has distinct neighbors $w_1, \ldots, w_k$, such that the visited degree of w_i is at least $i-1$, and each w_i is visited before v.*

Proof: Let w_i be the ith visited neighbor of v. Each w_i, $i \leq k$ is visited before v. Just before w_i is visited, v has current visited degree exactly $i-1$. As the MCS visits w_i instead of v at that point, w_i must have visited degree at least $i-1$. □

Corollary 3 *Let ψ be an MCS-ordering of G. The visited degree of a vertex is at most the maximum degree of its neighbors.*

Proof: Each neighbor of v that is visited before v has visited degree at most its degree minus one. Now, use Lemma 2. □

The *degeneracy* of a graph $G = (V, E)$ is the maximum over all induced subgraphs H of G of the minimum degree of a vertex in H and is denoted by $\delta D(G)$. It is easy to see that the degeneracy can be computed in linear time, and that it is a lower bound for the treewidth, see e.g., [10]. In [6], the *contraction degeneracy* $\delta C(G)$ of a graph G is introduced as the maximum over all minors H of G of the minimum degree of a vertex in H. It also holds that $\delta C(G) \leq tw(G)$, but in contrast to the degeneracy it is NP-hard to compute, cf. [6].

3 MCSLB(G) for planar graphs

In this section, we consider the maximum visited degree of planar graphs G. We show that for each planar graph G with n vertices, MCSLB(G) $= O(\log n)$. We also give examples that show that there are planar graphs with arbitrary large values of MCSLB(G). We start the section by giving examples of such graphs. After that, we give the proof of the upper bound. The examples that show that MCSLB(G) can be arbitrarily large can also be used for a comparison with the contraction degeneracy. At the end of this section the same examples are used to show that the maximum visited degree is not closed under taking of induced subgraphs and minors.

3.1 Planar graphs with large MCSLB(G)

Theorem 4 *For every k, there is a planar graph G_k, such that*

(i). The maximum visited degree of G equals k.

(ii). The treewidth of G_k is k.

(iii). There is a vertex v in G_k such that every MCS-ordering that starts in v has visited degree 2.

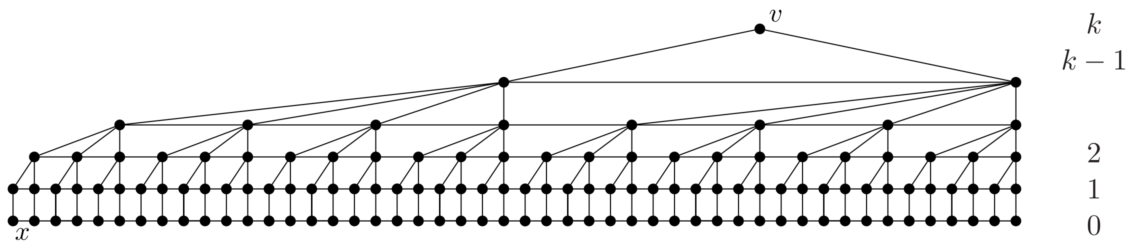

Figure 1: The construction for Theorem 4 for $k = 5$

(iv). G_k has $2(k-1)! \cdot (2 + \sum_{j=2}^{k-1} \frac{1}{j!}) + 1$ vertices.

Proof: We have $k+1$ layers. Number these layers from k to 0. On layer k, there is one vertex. For i, $1 \leq i \leq k-1$, there are $2(k-1)!/i!$ vertices on layer i. On layer 0, there are $2(k-1)!$ vertices. Hence, G_k has the number of vertices as stated in the theorem.

On each layer, we have a path, connecting all vertices on the layer. The vertex on layer k is adjacent to both vertices on layer $k-1$. Each vertex on a layer i, $0 \leq i \leq k-2$ is adjacent to one vertex on layer $i+1$; each vertex on a layer i, $1 \leq i \leq k-1$ is adjacent to k successive vertices on layer $i-1$. An example of the construction is shown in Figure 1.

We can start an MCS with a vertex in layer 0, (e.g., the vertex labeled x in Figure 1) and then visit the vertices layer by layer. The first visited vertex in layer i has visited degree i, and each next vertex in layer i has visited degree $i+1$. As vertices in layer $i+1$ have current visited degree at most $i+1$ when we are visiting the vertices in layer i, we can first visit all vertices in layer i, before going to a vertex in layer $i+1$. In this way, we get a maximum visited degree of k for the second visited vertex in layer $k-1$. So, $\mathrm{MCSLB}(G) \geq k$.

When we start the MCS with the vertex in layer k, labeled v in Figure 1, then any MCS will give maximum visited degree two. We must visit the vertices layer by layer, but now we go from higher numbered layers to lower numbered layers. Suppose we visited all vertices in layer i, $k \geq i > 1$. The next step will be a visit to a vertex in layer $i-1$. Now, as long as we did not yet visit all vertices in layer $i-1$, there will be at least one vertex in layer $i-1$ with current visited degree two, while all vertices in layer $i-2$ have current visited degree one. So, we first must visit all vertices in layer $i-1$, all with visited degree one or two, before we go to layer $i-2$.

The treewidth of the graph is exactly k. The treewidth of G_k is at least k, by Lucena's theorem: it has an MCS-ordering with maximum visited degree k. Let G'_k be the graph obtained by removing the vertex on layer k from G_k. As the vertex on layer k is a simplicial vertex of degree two, the treewidth of G_k is the maximum of two and the treewidth of G'_k. The treewidth of G'_k is at most k, as G'_k is a minor of an r by k grid for $r = 2(k-1)!$, and the treewidth of such a grid is k. We now have $\mathrm{MCSLB}(G) = k$, using the results proved above and Theorem 1. □

Note that the number of vertices in G_k is $\Theta(k!)$. Small variations of the construction give slightly different characteristics. The construction in Figure 2 has somewhat less vertices

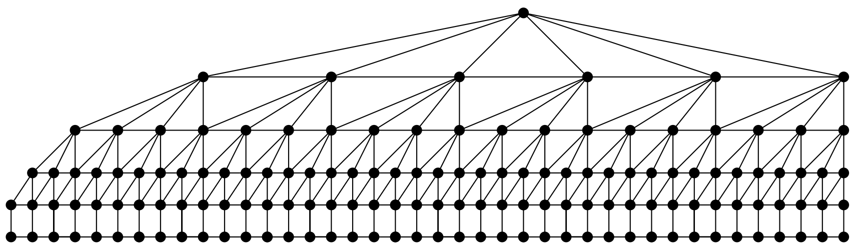

Figure 2: A construction with less vertices with $k = 6$

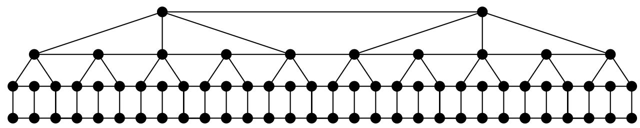

Figure 3: A construction with a bipartite planar graph

but the condition that every MCS-ordering starting in v gives a maximum visited degree of 2 must be weakened. In Figure 3, a similar construction is given, but now the graph is also bipartite. Again, the condition that every MCS-ordering starting in v gives a maximum bound of 2 must be weakened.

3.2 Planar graphs have MCSLB$(G) = O(\log n)$

We will now give the upper bound proof for planar graphs. A vertex w is said to be the *last successor of* a vertex v in an MCS-ordering, if w is a neighbor of v, and w is visited after v and after every other neighbor of v. Note that each vertex has at most one last successor.

Theorem 5 *If G is a planar graph with n vertices with maximum visited degree $k \geq 6$, then $n \geq 2^{\lceil \frac{k-1}{4} \rceil} - 1$.*

Proof: Suppose G is a planar graph, and let π be an MCS-ordering of G with visited degree k. Without loss of generality, we may suppose that G does not contain a proper subgraph H with MCSLB$(H) \geq k$. Let v be the first vertex visited by π with visited degree k.

Write $V = \{v_1, \ldots, v_n\}$, with v_i the ith vertex visited by π. Denote $G_{>i}$ as the subgraph, induced by the vertices $\{v_{i+1}, \ldots, v_n\}$. Similarly, denote $G_{\leq i}$ as the subgraph, induced by vertices $\{v_1, \ldots, v_i\}$.

Claim 5.1 *For each i, $G_{>i}$ and $G_{\leq i}$ are connected.*

Proof: Clearly, at each moment i during an MCS, the graph induced by the visited vertices $G_{\leq i}$ is connected.

Suppose $G_{>i}$ is not connected. Take a connected component from $G_{>i}$ that does not contain v. If we remove all vertices from that connected component from G and from π, we obtain

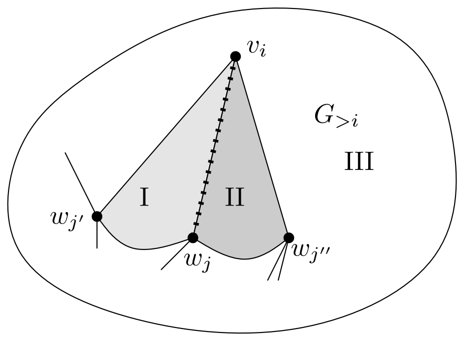

Figure 4: Illustration to the upper bound proof for planar graphs

a graph with an MCS-ordering where v still has visited degree k. This contradicts the minimality of G. $\diamond$

Claim 5.2 *Consider an i, $1 \leq i \leq n$, with v_i a vertex with visited degree $\ell \geq 6$ in π. There are two vertices with visited degree at least $\ell - 4$ in π for which v_i is the last successor.*

Proof: Consider a drawing of G, and the induced drawing of $G_{\leq i}$. By Proposition 5.1 there must be one face in this drawing of $G_{\leq i}$ that contains the positions of all vertices of $G_{>i}$. Without loss of generality, we may suppose this is the external face.

Consider a neighbor w of v_i that is visited before v_i. If v_i is not the last successor of w, then w must have a neighbor in $G_{>i}$, and hence w must lie at the external face of $G_{\leq i}$.

Now, consider the last four neighbors of v_i that are visited before v_i, and call them w_1, w_2, w_3, and w_4. Just before w_i is visited, v_i has current visited degree $\ell - 5 + i \geq 2$ ($1 \leq i \leq 4$.) As in Lemma 2, each w_i has visited degree at least $\ell - 4$. We claim that at most two of these can be at the external face of $G_{\leq i}$. Suppose three of these are at the external face of $G_{\leq i}$, cf. Figure 4. Then, for at least one of these, say w_j, the edge $\{v_i, w_j\}$ is a separator of $G_{\leq i}$. Just after w_j is visited, one component of the graph $G_{\leq i} - \{v_i, w_j\}$ obtained by removing v_i, w_j and incident edges from $G_{\leq i}$ does not contain visited vertices. The first visited vertex in that component must be visited before v_i (as it is in $G_{\leq i}$), and it has visited degree one (as w_j is the only visited vertex it can be adjacent to). This contradicts the property of MCS-ordering, as v_i has current visited degree at least two after w_j is visited.

So, at most two vertices from $\{w_1, w_2, w_3, w_4\}$ do not have v_i as their last successor. As each of these has visited degree at least $\ell - 4$, the proposition follows. $\diamond$

Now, consider the following directed subgraph of G. For each vertex $v \in V$ with a last successor, we take an arc from v to its last successor. As each vertex has at most one last successor, these arcs form a forest. Consider the subtree of the forest with v as root. v has visited degree k, and each vertex with $\ell \geq 6$ has at least two children with visited degree at least $\ell - 4$. With induction, it follows that if $k \in \{4r-2, 4r-1, 4r, 4r+1\}$, $r \geq 1$, then the forest contains at least $2^r - 1$ vertices. (This holds trivially if $k \leq 5$. Otherwise, we have two disjoint subtrees each with at least $2^{r-1} - 1$ vertices, and the root vertex.)

As G contains at least as many vertices as a subtree of it, we have that $n \geq 2^r - 1$ with $r = \lceil \frac{k-1}{4} \rceil$. □

Thus, we can conclude that the MCS-lower bound for treewidth is $O(\log n)$ on planar graphs. As planar graphs can have treewidth $\Omega(\sqrt{n})$, this shows that the MCS-lower bound may be not very effective for planar graphs. (E.g., an r by r grid has treewidth r, but the MCS-lower bound will be two for these grids.)

3.3 MCSLB(G) vs. (contraction) degeneracy

It is interesting to compare the degeneracy $\delta D(G)$ and contraction degeneracy $\delta C(G)$ of graphs with the maximum visited degree, as all three are lower bounds for the treewidth.

Theorem 6 *(i). For every graph G and every MCS-ordering ψ, the maximum visited degree of ψ is at least the degeneracy of G, $mcslb_\psi(G) \geq \delta D(G)$.*

(ii). For each k, there is a graph $G = (V, E)$ for which each MCS-ordering ψ gives $mcslb_\psi(G) = 2$, but for which $\delta C(G) \geq k$.

(iii). For each k, there is a graph $G = (V, E)$ and a vertex $v \in V$, such that each MCS-ordering ψ of G starting at vertex v gives $mcslb_\psi(G) = k$, but for which $\delta C(G) \leq 5$.

Proof:

(i). Let G be a graph with degeneracy k. Let H be a subgraph of G with minimum vertex degree k. Let ψ be an MCS-ordering of G. Suppose v is the vertex from H that is last visited by ψ. v has k neighbors in H that are visited before it, so has visited degree at least k.

(ii). Let G be obtained by taking a clique with $k+1$ vertices and then subdividing each edge. Hence, the clique on $k+1$ vertices is a minor of G with minimum vertex degree k, and thus $\delta C(G) \geq k$. It is not hard to see that each MCS-ordering of G has visited degree two: the neighbors of an unvisited clique vertex can have current visited degree at most one. Hence, as soon as two of them are visited, the clique vertex must be visited.

(iii). Suppose we execute the MMD+ algorithm on a planar graph. Contracting edges in planar graphs gives again a graph that is planar. As each planar graph has a vertex of degree at most five, $\delta C(G) \leq 5$, see also [6]. So, the result follows when we use the graph G_k from Theorem 4. □

3.4 MCSLB(G) and induced subgraphs/minors

The notion maximum visited degree is not closed under taking of induced subgraphs, and hence also not under taking of minors, as is shown in the following proposition. More elaborate examples follow from the proof of Theorem 9.

Proposition 7 *There are graphs G and H, such that H is an induced subgraph of G, and the maximum visited degree of G is smaller than the maximum visited degree of H.*

Proof: A possible example is the following. We let H be the graph shown in Figure 1. For G, we add two vertices, and make these adjacent to all vertices on the lowest and one-but-highest level. A simple but tedious case analysis shows that G has maximum visited degree 4, while we have argued in Theorem 4 that H has maximum visited degree 5. □

4 Complexity

In this section, we will show that the problem to decide for a given graph whether its maximum visited degree is at least k is NP-complete, even for fixed $k \geq 7$. A corollary of the proof is that this problem also does not have a polynomial time constant approximation algorithm with a fixed ratio, unless P=NP. A variant of the proof shows that the problem to decide whether there exists an MCS-ordering with a visited degree *at most* k is also NP-complete.

4.1 Complexity for MCSLB(G) with prescribed start

We first consider a version of the problem, namely, for the case that the starting vertex is specified. NP-completeness of this version would also directly follow from the NP-completeness of the problem without starting vertex, but the proof is somewhat simpler and helpful to prove the unrestricted case. The problem is formally described as:

MAX MCS-LB WITH PRESCRIBED START
Instance: Graph $G = (V, E)$, vertex $v_0 \in V$, integer $k \leq |V|$.
Question: Is there an MCS-ordering ψ starting at v_0 with $mcslb_\psi(G) \geq k$?

Theorem 8 MAX MCS-LB WITH PRESCRIBED START *is NP-complete, even when k is a constant that is at least 6.*

Proof: It is trivial that MAX MCS-LB WITH PRESCRIBED START belongs to NP. To proof NP-hardness, we use a transformation from 3-SATISFIABILITY.

Suppose we are given a set of clauses C, each with three literals, over the set of Boolean variables $\{x_1, \ldots, x_n\}$. Suppose also that we are given an integer $k \geq 6$. Note that our construction is exponential in k, so we assume k to be a fixed constant.

We build a graph $G'_{C,k}$ with the property that if the set of clauses C is satisfiable, then $G'_{C,k}$ has an MCS-ordering ψ starting at specified vertex v_0 with $mcslb_\psi(G'_{C,k}) \geq k$. and if the set of clauses is not satisfiable, then for every MCS-ordering ψ of $G'_{C,k}$ that starts in v_0, $mcslb_\psi(G'_{C,k}) \leq 5$. The construction is not the most efficient one, but a more efficient one would need an (even) more detailed description.

The graph $G'_{C,k}$ consists of a number of parts. We have a relatively simple *start part*, given in Figure 5.

For each variable x_i, we have a *variable part*, which is shown in Figure 6. These variable parts are put in series, attached to each other at the vertices marked $a_{i,1}$ and $a_{i,2}$; the first variable part is attached to the start part.

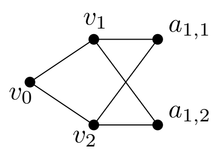

Figure 5: The start part

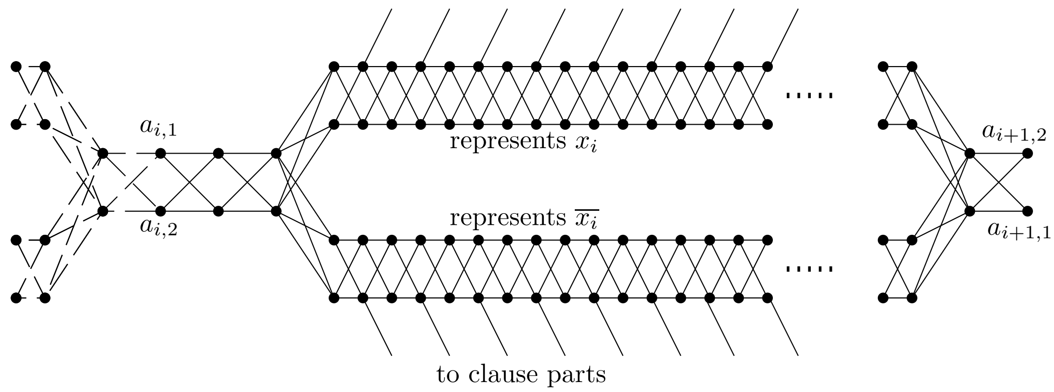

Figure 6: The variable part of variable x_i

The variable part can be seen to consist of four subparts: the beginning of the part, with the attachment to the variable part of the previous variable (or, in case of variable x_1, to the start part), the ending of the part with the attachment to the next variable part (or, in case of variable x_n, to the 'harvest part', to be described later); and two long components, one representing the variable to be set to false, and one representing the variable to be set to true. Associate the top long component with x_i, i.e., the case that x_i is true, and the bottom long component with $\overline{x_i}$, i.e., the case that x_i is false. (In the proof, the component of a literal set to true will be visited much later than the component of a literal set to false.)

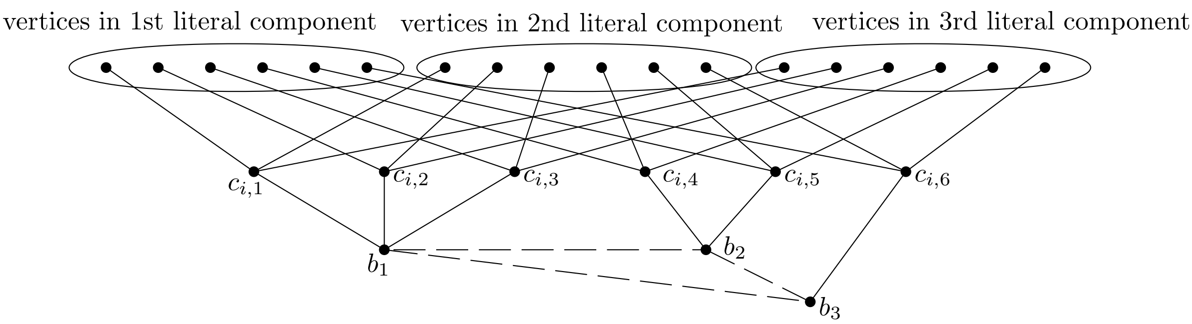

Figure 7: A clause part

A *clause part* consists of six vertices as shown in Figure 7. Each is adjacent to one vertex from the set $\{b_1, b_2, b_3\}$. These vertices are called the *bad* vertices: their role will be described later.

For each clause we have a clause part. The vertices in the clause part are made adjacent to vertices in the corresponding literal components, e.g., a clause part of a clause $C_i = \{x_2, \overline{x_3}, x_4\}$ will have edges to vertices in the top part of the variable part of x_2, the bottom part of the variable part of x_3, and the top part of the variable part of x_4. To each of these parts, there are six edges, as indicated by the figures. The literal components are made long enough such that each vertex in the component is adjacent to at most one vertex in a clause part, and that vertices in the literal component adjacent to a clause part are not neighboring each other.

The *harvest part* consists of $k+3$ layers, numbered layer -1 till layer $k+1$. Layer -1 contains two vertices which we call g_1 and g_2. These are the *good vertices*: visiting these 'early' can give us a large maximum visited degree for some other vertices by the MCS. Layer $k+1$ consists of the three 'bad' vertices b_1, b_2, and b_3. As discussed before, these vertices are adjacent to vertices in clause components.

Layer k contains 16 vertices, and layer $k-1$ contains 12 vertices. Layer $k-2$ also contains 12 vertices. For i, $1 \leq i \leq k-3$, denote $\alpha_i = 12 \cdot (k-2)!/i!$. Write $\alpha_0 = 12 \cdot (k-2)!$. For i, $0 \leq i \leq k-3$, layer i contains α_i vertices. Note that the number of vertices in layer i is exactly $i+1$ times the number of vertices of layer $i+1$. Write $\alpha_{k-1} = 12$, $\alpha_k = 12$, $\alpha_{k+1} = 16$. We denote the vertices in layer i as $h_{i,j}$, $0 \leq j < \alpha_i$.

We have several different types of edges in the harvest part. Note that there are only edges between vertices in the same layer and between vertices in neighboring layers.

clique edges: b_1, b_2, and b_3 form a clique.

top edges: Each vertex in layer k is adjacent to b_1, b_2, and b_3.

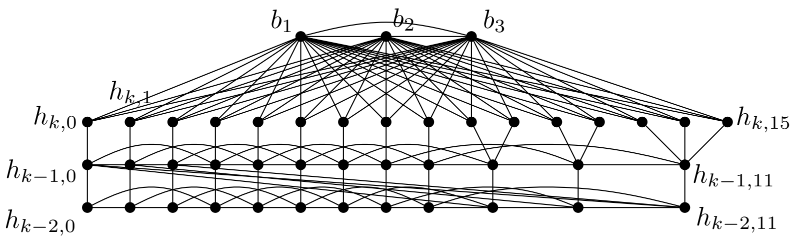

Figure 8: The top four layers of the harvest part)

tree edges: $h_{k-1,11}$ is adjacent to three vertices in layer k: $h_{k,15}$, $h_{k,14}$, and $h_{k,13}$. $h_{k-1,10}$ is adjacent to two vertices in layer k; $h_{k,12}$ and $h_{k,11}$. $h_{k-1,9}$ is also adjacent to two vertices in layer k; $h_{k,10}$ and $h_{k,9}$. For j, $0 \leq j \leq 8$, $h_{k-1,j}$ is adjacent to $h_{k,j}$.

Moreover, for each j, $0 \leq j \leq 11$, we have an edge $\{h_{k-1,j}, h_{k-2,j}\}$. Figure 8 shows the graph formed by levels $k-2$, $k-1$, k, and $k+1$.

A third set of tree edges is defined for i, $1 \leq i \leq k-2$. We give each vertex in layer i, i edges to vertices in the layer below, in the following manner. For j, $0 \leq j \leq \alpha_i - 1$, $h_{i,j}$ has edges to each vertex of the form $h_{i-1,j+\alpha_i \cdot \beta}$, $0 \leq \beta \leq i-1$. Note that each vertex in layers 1 till $k-2$ has exactly one incident tree edge to a vertex in the layer above.

bottom edges: We take edges $\{g_1, h_{0,0}\}$, $\{g_1, h_{0,2}\}$, $\{g_2, h_{0,0}\}$, $\{g_2, h_{0,2}\}$, cf. Figure 9.

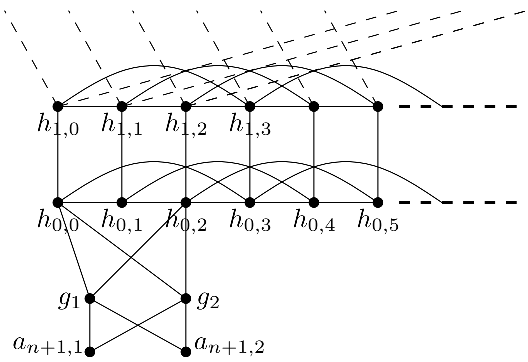

Figure 9: The bottom three layers of the harvest part and $a_{n+1,1}$, $a_{n+1,2}$

path edges: For i, $0 \le i \le k-1$, the vertices in the ith layer are interconnected by a path, i.e., we have edges $\{h_{i,j}, h_{i,j+1}\}$ for all i, $0 \le i \le k-1$ and j, $0 \le j < \alpha_i - 1$.

In addition, for i, $0 \le i \le k-2$, we have an edge from the last vertex in layer i to the first vertex of the next layer, i.e., an edge $\{h_{i,\alpha_i-1}, h_{i+1,0}\}$. All path edges form together a Hamiltonian path in the subgraph formed by layers 0 till $k-1$.

distance three edges: For i, $0 \le i \le k-1$, vertices in layer i are adjacent to vertices at distance three in the same layer, i.e., we have edges $\{h_{i,j}, h_{i,j+3}\}$, for $0 \le j \le \alpha_i - 4$.

We have also some edges that behave similar to the edges of distance three, but are between vertices in different layers, as follows. For i, $0 \le i \le k-2$, we have edges $\{h_{i,\alpha_i-3}, h_{i+1,0}\}$, $\{h_{i,\alpha_i-2}, h_{i+1,1}\}$, $\{h_{i,\alpha_i-1}, h_{i+1,2}\}$. Note that distance three edges are always between vertices that have distance three on the path formed by the path edges.

Note that each vertex is adjacent to at most two path edges, at most two distance three edges, and a number of tree edges. Tree edges go between different levels; a vertex has at most one tree edge to the layer above it, except for the vertices in layers k and $k-1$; it can have more tree edges to the layer below.

The harvest part can be seen as a modification of the graph of Theorem 4. Where in that graph we have one tree with edges on levels, we here have 12 trees; in addition, we have edges from the last vertex of a layer to the first vertex of the next layer, and edges between vertices at 'distance three'. This makes that when we have b_1, b_2, and b_3 are visited before g_1 and g_2, then we must visit all vertices in the harvest part (except b_1 and b_2) at visited degree three. The modifications also play a role in the proof when we allow a start not at v_0; this is discussed in the proof of Theorem 9.

The harvest part has a similar property as the graph of Theorem 4: when we start visiting it from the vertices g_1 and g_2, then we can reach a maximum visited degree in some vertices in the harvest part of k; while when we start visiting it from the vertices b_1, b_2, b_3, then the maximum visited degree of all vertices in the harvest part will be bounded by a small constant.

Finally, we make $a_{n+1,1}$ and $a_{n+1,2}$ adjacent to each of g_1 and to g_2, as shown in Figure 9. Let $G'_{C,k}$ be the resulting graph.

Claim 8.1 *If there is a truth assignment that satisfies C, then there is an MCS-ordering ψ of $G'_{C,k}$ that starts at v_0 with $mcslb_\psi(G'_{C,k}) \geq k$.*

Proof: Suppose we have a truth assignment that satisfies C. We build the requested MCS-ordering ψ as follows. After v_0, we visit v_1, v_2, $a_{1,1}$, and $a_{1,2}$.

Then, for $i = 1, \ldots, n$, we go from $a_{i,1}$ and $a_{i,2}$ to $a_{i+1,1}$, $a_{i+1,2}$, by going through the ith variable part. If variable x_i is set to true in the satisfying truth assignment, then we visit all vertices in the bottom literal component and no vertex in the top literal component; otherwise, we visit all vertices in the top literal component, and no vertices in the bottom literal component. So, we visit the vertices in the part that corresponds to the false literal. Each of the vertices that we visit in this phase has visited degree exactly two.

Consider a clause vertex $c_{i,j}$. As the ith clause contains a true literal, at most two neighbours of $c_{i,j}$ are visited in this phase of the MCS. Hence, each clause vertex $c_{i,j}$ has current visited degree at most two: there is no need to visit these vertices during this phase already.

After we have reached $a_{n+1,1}$ and $a_{n+1,2}$, we visit g_1 and g_2. After this, we visit the vertices in the harvest part layer by layer. We first visit $h_{0,0}$, $h_{0,2}$, $h_{0,1}$, and $h_{0,3}$, in that order. Then we visit all remaining vertices in layer 0, from left to right. Each of these gets visited degree two: when $h_{0,i}$ is visited, it has visited neighbors $h_{0,i-3}$ and $h_{0,i-1}$.

Then we continue visiting the harvest part layer by layer up to layer $k-1$, visiting each layer from left to right. In this way, the vertices in layer i, $1 \leq i \leq k-2$, get visited degree $i+2$: they have i visited neighbors via tree edges, one visited neighbor via a path edge, and one visited neighbor via a distance three edge. We can also note that when not all vertices in layer i have been visited, vertices in layer $i+1$ have current visited degree at most $i+2$, so we indeed can move through the harvest part with MCS in the layer by layer order.

After the harvest part is visited, we visit all remaining unvisited vertices, in any order fulfilling the MCS rule. The resulting MCS-ordering has visited degree at least k, as the vertices in layer $k-2$ have visited degree k. $\diamond$

Claim 8.2 *b_1, b_2, and b_3 are visited before each g_1 and g_2.*

Proof: When g_1 or g_2 is visited, it can have visited neighbors either in the nth variable part, or in the harvest part. Suppose first, it has only visited neighbors in the harvest part, i.e., $h_{0,0}$ and $h_{0,2}$. As any path from v_0 to $h_{0,0}$ and $h_{0,2}$, not containing g_1 or g_2, must use a vertex in b_1, b_2, b_3, at least one of these is visited. Moreover, after some vertices from $\{b_1, b_2, b_3\}$ are visited, the MCS will visit vertices in layer k of the harvest part. Consider the first visited vertex in the harvest part. It must have visited degree two, so at least two vertices from $\{b_1, b_2, b_3\}$ are visited before it. After that, if not all three vertices in

$\{b_1, b_2, b_3\}$ are visited, the third one has current visited degree at least three, so will be visited before g_1 and g_2. So, in this case, the claim holds.

Now, suppose that when g_1 or g_2 is visited, it has a visited neighbor in the nth variable part. With induction, this applies that for each of the variable parts, we must have visited either the top literal component, or the bottom literal component of each variable part. Thus, each variable has been set either to true, or to false.

As the set of clauses was not satisfiable, there must be a clause of which all three literals are set to false. This implies that the vertices in the corresponding clause component have three visited neighbors in the literal parts before g_1 and/or g_2 is visited. As visiting g_1 or g_2 via the variable parts implies visiting vertices with current visited degree two, whereas the current visited degree of c_{ij}, $1 \leq j \leq 6$ for some C_i is three, the vertices in the clause part will all be visited before g_1 and g_2. After the vertices in the clause part are visited, we must visit b_1, then b_2, b_3, as each of these has visited degree at least three. This concludes the proof of the claim. $\diamond$

Claim 8.3 *If there does not exist a truth assignment that satisfies C, then each MCS-ordering ψ of $G'_{C,k}$ that starts at v_0 has $mcslb_\psi(G'_{C,k}) \leq 5$.*

Proof: Assume that there does not exist a satisfying truth assignment of C. First, observe that every vertices in a start, variable, or clause part either has degree at most four, or only has neighbours of degree at most four. As a consequence, using Corollary 3, none of these vertices can be have visited degree larger than four. So, if we want to obtain a visited degree larger than four, this has to happen in the harvest part.

As we will see, it will be necessary to visit the harvest part first at the bad side (i.e., the top levels, starting at vertices b_1, b_2, and b_3), and this forces us that vertices in the harvest part are visited with their visited degree bounded by three (see below.)

We make a number of observations about possible MCS-orderings starting at v_0. After v_0, we visit v_1 and v_2, and from that point on till all vertices are visited, there are always unvisited vertices with current visited degree at least two. So, the only vertices that can be visited with visited degree less than two are v_0, v_1, and v_2.

Suppose both g_1 and g_2 are not visited yet. Vertices in variable parts have either only neighbours in variable parts, or one neighbour not in a variable part, but then their neighbours have only neighbours in variable parts. Thus, it is impossible to 'enter' a variable part by edges only from clause components — that would mean a vertex with visited degree one. So, one can observe that vertices $a_{i+1,1}$ and $a_{i+1,2}$, $i > 1$ can be visited only after either all vertices in the top literal component or all vertices in the bottom literal component of the ith variable part have been visited. Say we have set a variable to true when we have visited all vertices in the bottom part, and we have set a variable to false when we have visited all vertices in the top part of the corresponding variable part. So, visiting all vertices of a literal component corresponds to setting that literal to false. If all literals of C_i are set false, the current visited degree of c_{ij}, $1 \leq j \leq 6$, equals three and thus these vertices have to be visited first.

From Claim 8.2 we know that b_1, b_2, and b_3 are visited before each g_1 and g_2. We now consider the MCS from the moment that b_1, b_2, and b_3 are visited. We now look to the way the layers k till 1 of the harvest part must be visited. The MCS through this part of the graph can be interleaved with visits to vertices in other parts of the graphs, but we will see that g_1 and g_2 cannot be reached from vertices outside the harvest part until all vertices in layers k till 1 are visited.

A possible manner to go with MCS through layers k till 1 through the harvest part is by visiting these layer by layer, going through each layer from right to left. In this way, each vertex in these layers gets visited degree three. While it is possible to vary a little from this scheme, this is essentially the way that one must visit the harvest part, as we will see now.

First, we note that as long as not all vertices in the layers 1 till k of the harvest part are visited, there is an unvisited vertex in the harvest part with current visited degree (at least) three: consider the highest layer i of the harvest part with unvisited vertices, and from this layer, take the vertex $h_{i,j}$ with largest index j, i.e., we take unvisited vertex $h_{i,j}$ with (i,j) lexicographically maximal. Simple case analysis shows that $h_{i,j}$ has at least three visited neighbors. (Usually, it has one visited neighbor via a tree edge, one via a path edge, and one via a distance three edge.) So, as long as layers 1 till k are not entirely visited, we cannot visit vertices with current visited degree two. In particular, we cannot visit g_1 or g_2 until $h_{0,0}$ or $h_{0,2}$ are visited.

Consider layer i, $1 \leq i \leq k-1$. The only vertex that has three or more neighbors in layer $i+1$ is h_{i,α_i-1}, so that this vertex must be the first vertex in the layer that is visited. After h_{i,α_i-1} is visited, the next vertex on layer i that is visited must be h_{i,α_i-2}: it has two neighbors in layer $i+1$ and one visited neighbor in layer i, while each other vertex in layer i has current visited degree at most two. With induction, it follows that $h_{i,j}$ is visited only after all vertices of the form $h_{i,j'}$, $j' > j$ are visited: so we visit layers from right to left. If $1 \leq i \leq k-2$, then h_{i,α_i-1} must be visited after $h_{i+1,0}$, as $h_{i+1,0}$ is one of the three neighbors of h_{i,α_i-1} in layer $i+1$. As $h_{i+1,0}$ is the last vertex visited in layer $i+1$, layer i must be visited after all vertices in layer $i+1$ have been visited. So, we have a fixed order in which the vertices in layers 1 till $k-1$ are visited: we visit the layers from top till bottom, in order, and each layer from left to right. In this way, each of the vertices in layers 1 till $k-1$ receive visited degree exactly three.

Now we can give a bound on the maximum visited degree that we can obtain on $G'_{C,k}$ in the case the set of clauses was not satisfiable. For each vertex z in the start part, a variable part, a clause part, and layers -1, 0, and k of the harvest part, we have that the degree of z is at most four, or all neighbors of z have degree at most four; so none of these vertices can have a visited degree larger than four, see Corollary 3. We argued above that vertices in layers 1 till $k-1$ obtain visited degree three. Finally, b_1, b_2, and b_3 have only two neighbours with a degree larger than four, so are visited when they have at most five visited neighbours. This completes the proof of Claim 8.3. $\diamond$

From Claim 8.1 and 8.3 and the fact that $G'_{C,k}$ can be constructed in polynomial time for fixed k and given set of clauses C, we can conclude the NP-completeness. $\square$

4.2 Complexity for MCSLB(G) with free start

We now modify the proof of Theorem 8 to obtain an NP-completeness result for the case that the starting vertex is not fixed.

> MAX MCS-LB
> **Instance:** Graph $G = (V, E)$, integer $k \leq |V|$.
> **Question:** Is there an MCS-ordering for G with $mcslb_\psi(G) \geq k$?

Theorem 9 MAX MCS-LB *is NP-complete, even when k is a constant that is at least 7.*

Proof: The problem clearly also belongs to NP. The NP-completeness proof uses a modification of the method used to proof Theorem 8. We do not repeat the description of that proof here, but instead describe the differences and modifications.

Suppose we are given a set of clauses C, each with three literals, over the set of Boolean variables $\{x_1, \ldots, x_n\}$. Suppose also that we are given an integer $k \geq 7$.

First, build the graph $G'_{C,k}$ as in the proof of Theorem 8. In contrast to the case with prescribed start, we have to avoid that the MCSLB starts with an arbitrary vertex, e.g., at the bottom of the harvest part, by this obtaining a high bound. Therefore, we modify the graph by adding a *punishment part* for every edge e in $G'_{C,k}$ except for the edges $\{v_0, v_1\}$, $\{v_0, v_2\}$, $\{b_1, b_2\}$, $\{b_1, b_3\}$, and $\{b_2, b_3\}$, and edges between a vertex in $\{b_1, b_2, b_3\}$ and a vertex in layer k of the harvest part. Note that an edge with a punishment part does not belong to a triangle in $G'_{C,k}$.

Such a punishment part consists of 15 new vertices with edges as shown in Figure 10. The punishment part is at one side connected to its edge $\{v, w\}$, and at the other side to the 'bad' vertices b_1, b_2, and b_3. The idea behind this structure is to make sure that when we do not start at v_0 (or v_1 or v_2), then we must go via a punishment part to b_1, b_2, and b_3 and then visit the vertices in the harvest part giving each vertex in the harvest part a small visited degree.

Consider the punishment part of edge $e = \{v, w\}$. We say an MCS-ordering *enters* the punishment part at the *e-side*, when the first visited vertex in the punishment part is a neighbor of v and/or w, i.e., p_{e_1}, $p_{e,2}$, or $p_{e,3}$, and that vertex is visited after v or w. The MCS-ordering *enters* the punishment part at the *b-side*, when the first visited vertex is a neighbor of b_1, b_2, or b_3, i.e., $p_{e,j}$, $12 \leq j \leq 15$, this vertex visited after its neighbor in $\{b_1, b_2, b_3\}$. Exactly one of the following cases holds for each punishment part: it is entered at the e-side, it is entered at the b-side, or the MCS-ordering started with a vertex in this part.

Let $G_{C,k}$ be the resulting graph. Note that $G'_{C,k}$ is the subgraph of $G_{C,k}$, induced by the vertices that do not belong to a punishment part only.

Claim 9.1 *If there is a truth assignment that satisfies C, then there is an MCS-ordering ψ of $G_{C,k}$ that starts at v_0 with $mcslb_\psi(G'_{C,k}) \geq k$.*

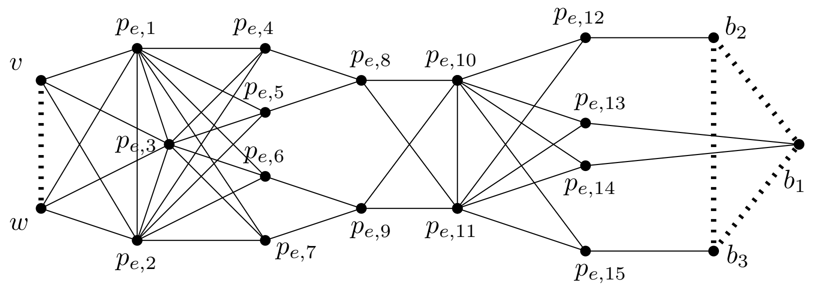

Figure 10: The punishment part for edge $e = \{v, w\}$

Proof: We can start an MCS of $G_{C,k}$ similar as in Claim 8.1. Note that we do not need to visit vertices in punishment parts after we have reached a vertex in the harvest part with visited degree k (in layer $k-2$), as vertices in punishment parts have current visited degree at most two during this initial part of the MCS. $\diamond$

To prove the 'reverse' claim, we need a sequence of claims on the maximum visited degree of the vertices in the various parts.

Claim 9.2 *A vertex in a punishment part has visited degree at most six.*

Proof: All vertices $p_{e,j}$ with $4 \le j \le 15$ have degree at most four or have only neighbours with degree at most three, so these vertices cannot have visited degree larger than four.

$p_{e,1}$, $p_{e,2}$, and $p_{e,3}$ have degree eight, but a case analysis reveals that we cannot get a visited degree larger than six at these vertices. Write $A = \{v, w, p_{e,1}, p_{e,2}, p_{e,3}\}$; note A forms a clique. Write $B = \{p_{e,4}, p_{e,5}, p_{e,6}, p_{e,7}\}$. Consider the first three vertices visited in $A \cup B$. It cannot be the case that all three belong to B: after two vertices are visited in B, vertices in A have two visited neighbours, while unvisited vertices in B can have at most one visited neighbour. Indeed, when two vertices in B are visited, $p_{e,1}$, $p_{e,2}$, and $p_{e,3}$ will have more visited neighbours than vertices in B, so each of these will be visited with at most two visited neighbours in B, so with a visited degree of at most six. $\diamond$

A more refined case analysis shows that we even cannot get a vertex with visited degree larger than five in a punishment part, but this is not needed for our proof.

Claim 9.3 b_1*,* b_2*, and* b_3 *have visited degree at most five.*

Proof: b_1, b_2, and b_3 have two neighbors of degree larger than three. Now use Lemma 2. $\diamond$

Claim 9.4 *After at least two vertices from* $\{b_1, b_2, b_3\}$ *are visited, no vertex will be visited with visited degree at most one.*

Proof: After at least two vertices from $\{b_1, b_2, b_3\}$ are visited, and as long as not all vertices in $G'_{C,k}$ (i.e., outside punishment parts) are visited, there will be an unvisited vertex in $G'_{C,k}$ with current visited degree at least two. (This can be seen by inspection of $G'_{C,k}$.)

After both endpoints of an edge e are visited, and while not all vertices in the punishment part of e are visited, there will be an unvisited vertex in this punishment part of current visited degree at least two. $\diamond$

Claim 9.4 implies that after two vertices from $\{b_1, b_2, b_3\}$ are visited, no punishment part will be entered at the b-side, as entering a punishment part at the b-side implies visiting a vertex with visited degree one.

Claim 9.5 *Let v be a vertex in $G'_{C,k}$, $v \notin \{b_1, b_2, b_3\}$. Suppose v is visited after a neighbor of v in the punishment part of edge $e = \{v, w\}$. Then v has at most two visited neighbors outside the punishment part when it is visited.*

Proof: We consider three cases. In the first two cases, v can have only one such neighbor, namely w.

First, suppose the MCS enters the punishment part from e from the e-side. Then, after w is visited, a vertex $p_{e,j}$, $j = 1, 2, 3$ is visited with visited degree one. After that, the remaining unvisited vertices in $\{v, p_{e,1}, p_{e,2}, p_{e,3}\}$ will have maximum visited degree and will be visited. When v is visited, it cannot have a visited neighbor outside the punishment part of e, as then it would have current visited degree two at the moment that the first vertex in the punishment part was visited (with visited degree one.)

Second, suppose the MCS enters the punishment part from the b-side. This must happen before two vertices from $\{b_1, b_2, b_3\}$ are visited, by Claim 9.4. Note that we cannot move the MCS through a punishment part from the b-side, and reach a vertex in $\{p_{e,4}, p_{e,5}, p_{e,6}, p_{e,7}\}$: after $p_{e,8}$ or $p_{e,9}$, and $p_{e,10}$ or $p_{e,11}$ are reached, we have that $\{p_{e,j} \mid 8 \leq j \leq 15\}$ will either have only visited vertices, or an unvisited vertex of current visited degree at least two. Thus, before a vertex in $\{p_{e,4}, p_{e,5}, p_{e,6}, p_{e,7}\}$ can be reached with a path from the b-side, we must have visited the second vertex from $\{b_1, b_2, b_3\}$, and then Claim 9.4 applies. In particular, the neighbor of v in the punishment part that was visited before v must have been reached via w. Again, this neighbor has visited degree one, so at the time of this visit, v is not adjacent to another visited vertex except w. Simple case analysis shows again that the MCS can only visit vertices of the form $p_{e,j}$, $1 \leq j \leq 7$ before it visits v, which shows that the claim holds in this case.

In the third case, we assume the MCS starts inside the punishment part. Let the first two vertices visited be p_{e,j_1} and p_{e,j_2}. If $j_1 \geq 8$ and $j_2 \geq 8$, then one can see that we cannot visit a vertex in $\{p_{e,4}, p_{e,5}, p_{e,6}, p_{e,7}\}$ before two of $\{b_1, b_2, b_3\}$ are visited (using the MCS rule and case analysis), and then Claim 9.4 applies, hence v is visited before its neighbors in the punishment part. If $j_1 \leq 7$ and $j_2 \leq 7$, then case analysis shows that we visit v and w before $p_{e,8}$ and $p_{e,9}$, and hence, before any other vertex outside the punishment part. Consider the remaining case. Without loss of generality suppose $j_1 \leq 7$ and $j_2 \geq 8$. We look to the third visited vertex p_{e,j_3}. If $j_3 \leq 7$, the analysis is as in the second case. If $j_3 \geq 8$, then as in the first case, the MCS will visit two vertices from $\{b_1, b_2, b_3\}$ before any other vertex outside the punishment part. After these two vertices are visited, Claim 9.4 applies. Thus, the only way a vertex in $\{p_{e,j} \mid 1 \leq j \leq 7\}$ can now be chosen is when it has two visited neighbors. One of these is p_{e,j_1}, the other must be w. After that, the other vertices in $\{p_{e,1}, p_{e,2}, p_{e,3}, v\}$ must be selected (observe their current visited degrees). At the moment the first vertex in $\{p_{e,1}, p_{e,2}, p_{e,3}\}$ is visited, v has current visited degree at most two (with w one of its visited neighbors), and so the claim applies. $\diamond$

Claim 9.6 *Vertices in start, variable, and clause parts receive visited degree at most six.*

Proof: Let v be a vertex in a start, variable, or clause part. Suppose v has d neighbors in $G'_{C,k}$. If v has one or more visited neighbors in punishment parts at the time it is visited, its visited degree is at most five: at most two visited neighbors outside the punishment part, and at most three inside the part. If v has no visited neighbors in punishment parts, then either $d \leq 4$, and hence v has visited degree at most four, or all neighbors of v have degree in $G'_{d,k}$ at most four, hence visited degree at most five, and hence v has visited degree at most six. $\diamond$

Analysing the visited degree of vertices in the harvest part is harder, mainly because the first two visited vertices can belong to a harvest part, thus enabling that the MCS goes through the harvest part in a manner different from that in the proof of Claim 8.3.

Claim 9.7 *Vertices in the harvest part receive visted degree at most six.*

Proof: The analysis of the visited degree of vertices in the harvest part depends on how the MCS is started. We consider a number of different cases. By an argument, similar to the one used for start, variable, and clause parts, we see that vertices in layer -1 in the harvest part cannot have visited degree more than six.

Case 1. The first two vertices visited by the MCS are v_0 and a neighbor of v_0. As long as the MCS does not use vertices in punishment parts, we have an MCS starting in v_0, and moving in $G'_{C,k}$, which gives, by Claim 8.3, a visited degree of at most five.

Using punishment parts cannot help to get a visited degree larger than six. We can go via a punishment part from an edge in a start, variable, or clause part to the bad vertices b_1, b_2, or b_3. Doing so would also give that the MCS visits the harvest part from top to bottom, with the vertices in this part getting visited degree at most three, as in the proof of Claim 8.3.

Case 2. The first two visited vertices belong to $\{b_1, b_2, b_3\}$ or layer k of the harvest part. Each MCS starts with visiting the vertices from some maximal clique of the graph. In this case, this means that b_1, b_2, and b_3 are among the first four visited vertices.

One can now note that as long as not all vertices in layers 0 till k of the harvest part are visited, there is an unvisited vertex in a harvest part with current visited degree at least three. All vertices outside the harvest part will have visited degree at most two. So, we first visit the vertices in the harvest part.

As in the proof of Theorem 8, the MCS visits the harvest part layer by layer, from top to bottom, till layer 0, with the vertices receiving visited degree three.

Case 3. The first two visited vertices are endpoint of an edge with a punishment part. Suppose the first two vertices visited are v, w, and $e = \{v, w\}$ has a punishment part. Now $p_{e,1}$ and $p_{e,2}$ have two visited neighbours.

By construction, e is not part of a triangle in $G'_{C,k}$. (All edges that are part of a triangle in $G'_{C,k}$ are without a punishment part. A start at the endpoint of such an edge is handled in Case 2.)

This implies that $p_{e,1}$, $p_{e,2}$, and $p_{e,3}$ are the only vertices with visited degree (at least) two. So, we must visit $p_{e,1}$, $p_{e,2}$, and $p_{e,3}$, in some order. After these are visited, $p_{e,4}$ till $p_{e,7}$ have three visited neighbours, and are visited; then we must visit $p_{e,8}$ and $p_{e,9}$, etc. Until we visit b_1, all vertices outside the punishment part, except possibly b_1, have current visited degree at most one, while there is at least one vertex that is in the punishment part or is b_1 with visited degree two or more. So, we will visit vertices in the punishment part of e until we visit b_1. We then must visit possibly $p_{e,12}$ or $p_{e,15}$ and then b_2 or b_3.

We have seen that starting at the endpoints of an edge with a punishment part quickly leads to a visit of b_1, b_2, and b_3. This seems to imply that we must go through the harvest part in the same way as in the proof of Claim 8.3. However, the difference with that proof is that v and w are already visited, and thus we need a further case distinction depending on to what part of the graph v and w belong.

Case 3a. v and w belong to start or variable parts and do not have a neighbour in common with b_1, b_2, or b_3. After two vertices from $\{b_1, b_2, b_3\}$ have been visited, we can visit more vertices in the punishment part of e, the third vertex from $\{b_1, b_2, b_3\}$, or vertices of the type $h_{k,i}$. After a vertex $h_{k,i}$ has been visited, we must visit the remaining vertex from $\{b_1, b_2, b_3\}$ when it has not yet been visited. Then, after b_1, b_2, b_3 are visited, we must visit the vertices of the harvest part, in the same way as in the proof of Claim 8.3, layer by layer, from top to bottom. The vertices in layers $k-1$ till 0 thus receive visited degree three.

Case 3b. v and w belong to the the harvest part. As we assumed that the edge $\{v, w\}$ has a punishment part, v and w cannot be b_1, b_2, or b_3. Suppose $\{v, w\} = \{h_{i_0,j_0}, h_{i_1,j_1}\}$, with $i_0 \leq i_1$, and if $i_0 = i_1$, then $j_0 < j_1$.

Notice that as long as not all vertices in layers 0 till k are visited, some unvisited vertices in the harvest part have current visited degree three or more. So, all vertices in these layers of the harvest part must be visited with visited degree *at least* three. This fact will be used a number of times to show that their visited degree is *at most* six.

Consider first the vertices in layers $i_0 - 1$ and below, till layer 0. The argument used in the proof of Claim 8.3 that these layers must be visited from top to bottom, right to left, still applies for these layers. So, vertices in these layers get visited degree three.

Again, Claim 9.5 holds in this case, for all edges except for the edge $\{v, w\}$. We consider a number of subcases.

Case 3b-I. v and w belong to layer k or $k-1$ of the harvest part. As shown above, vertices in layers $k-2$ till 0 are still visited with visited degree three. Vertices in layer k and $k-1$ have at most six neighbors in $G'_{C,k}$, and are visited before their neighbors in punishment parts, so we cannot get a vertex in the harvest part of visited degree larger than six.

Case 3b-II. Edge $\{v, w\}$ is in between layer $k-1$ and $k-2$. So, one of v and w belongs to layer $k-2$, and one of v and w belongs to layer $k-1$. Vertices in layers 0 till $k-3$ get visited degree three, as shown above. Vertices in layer $k-1$ and k have at most six neighbors outside punishment parts. Vertices in layer $k-2$ still must be visited before any vertex in layer $k-3$ is visited. As vertices in layer $k-2$ have at most five neighbors in layers $k-2$ and $k-1$, they get visited degree at most five.

Case 3b-III. v and w belong to layers 0 till $k-2$. We consider the MCS while going through layers k till 0 in the harvest part. Consider the set S of visited vertices that belong to layers k till 0 at a certain point in the MCS. We call the set S *normal*, when there is an i', j' with S consisting of v, w, and all vertices $h_{i'',j''}$ with $i'' > i'$ or $i' = i''$ and $j'' \geq j'$, an i.e., apart from v and w, we have visited a number of consecutive layers, and from the lowest layer we visited a vertex and all vertices right of it.

Note that, as there are no triangles in layers k till 0 in the harvest part, no vertex in the harvest part is adjacent to both v and w.

A visit to a vertex in the harvest part is called *specific*, when the set of visited vertices in the harvest part is normal before the visit but is not normal after the visit. In other words, while we 'usually' visit the vertices layer by layer, each layer from right to left, in a specific visit, we deviate from this scheme.

If there are no specific visits, then all vertices in layers k till 0 have visited degree at most four (to visited neighbors we can have one tree edge, one path edge, one distance three edge, and v or w can be a visited neighbors). Also, visits that lead from a normal set to a normal set give visited degree at most four for the visited vertex.

Consider a specific visit, say to a vertex $h_{i',j'}$. For ease of presentation, we assume that $3 \leq j' \leq \alpha_{i'} - 5$. If this does not hold, then the arguments are the same, but we wrap to the next level. (E.g., if $j = \alpha_{i'} - 1$, then read $h_{i',j+3}$ as $h_{i'+1,2}$. The path and distance three edges between different levels are used in such cases.)

$h_{i',j'}$ has visited degree at least three. Consider the moment that $h_{i',j'}$ is visited. Because the visit to $h_{i',j'}$ is specific, $h_{i',j'-3}$, $h_{i',j'-1}$, and $h_{i',j'+1}$ are not visited. $h_{i',j'}$ can have one visited neighbor via a tree edge to layer $i'+1$. So, $h_{i',j'}$ must be adjacent to v or to w, and $h_{i',j'+3}$ (and hence all vertices in layer $i'+1$ and all vertices $h_{i',j''}$ with $j'' > j'+3$) must be visited.

Suppose without loss of generality, that $h_{i',j'}$ is adjacent to v. There now are again a few different cases.

Case 3b-III-a. The edge from $h_{i',j'}$ to v is a tree edge. In this case, v is in layer $i'-1$. We note that the vertices $h_{i',j''}$ with $j'-3 \leq j'' \leq j'+3$, $j'' \neq j'$ cannot be adjacent to v or w, by construction. The first case has two subcases, again.

Case 3b-III-a.1. $h_{i',j'+2}$ is not yet visited. We claim that $h_{i',j'+2}$ must be the next visited vertex. Note that it has current visited degree at least three: it is the only vertex with this property. Let x be the next visited vertex. x can be adjacent to v or w, to one other visited vertex via a tree edge, so must be adjacent to a visited vertex (not v or w) via a path or distance three edge. So, x must be one of the vertices $h_{i',j'-3}$, $h_{i',j'-1}$, $h_{i',j'+1}$, or $h_{i',j'+2}$. However, due to the construction of the harvest part, none of these vertices can be adjacent to v and w. (This follows, using that $h_{i',j'}$ is adjacent to v or w.) So, the only vertex of current visited degree three or more is $h_{i',j'+2}$ and it is visited next. The next visited vertex must be $h_{i',j'+1}$: this vertex has current visited degree four after the visit to $h_{i',j'+2}$, and it is the only vertex with current visited degree at least four. After the visit to $h_{i',j'+1}$, the set of visited vertices in the harvest part is again normal.

Case 3b-III-a.1. $h_{i',j'+2}$ is visited before $h_{i',j'}$. Recall that the edge from v to $h_{i',j'}$ is also a tree edge in the case we handle. Now, $h_{i',j'+1}$ must be the next visited vertex, as it is again the only vertex of current visited degree four (or more.) After this visit, we are again in a situation with a normal set of visited vertices.

Case 3b-III-b. v is adjacent to $h_{i',j'}$ through a path or distance three edge, and the edge $\{v,w\}$ is a tree edge. Now, the vertices $h_{i',j''}$ with $j'-3 \leq j'' \leq j'+3$, $j'' \neq j'$ cannot be adjacent to w, by construction. Simple case analysis reveal that the next vertices to be visited are $h_{i',j'+1}$ and $h_{i',j'+2}$, if these weren't visited yet; these receive visited degree at most five.

Case 3b-III-c. v is adjacent to $h_{i',j'}$ through a path or distance three edge, and the edge $\{v,w\}$ is a path or distance three edge. The case analysis here is again simple, but tedious. In each of the cases, after a small number of visits (at most four), we have again a normal set of visited vertices, and the maximum visited degree of one of these vertices is at most five.

So, the visited degree in the layers 0 till k of the harvest part is bounded by five.

Case 3b-IV. v or w belongs to layer -1 of the harvest part. This analysis of this case is similar to Case 3b-III. Note that there cannot be specific visits until most vertices in layer 2 are visited. An easy but somewhat tedious case analysis shows again the desired bound on the visited degree in the lower layers of the harvest part.

Case 3c. v and w belong to a variable or clause part and v or w has a neighbour in common with b_1, b_2, or b_3. In this case, after we reached b_1, b_2, or b_3 via the punishment part, we possibly can visit vertices that are a neighbour to a vertex in $\{b_1, b_2, b_3\}$

and neighbour to v or w. Observing the construction of clause and variable parts, there is at most one such vertex. Visiting that vertex does not increase the number of visited neighbours of unvisited vertices in start, clause, or variable parts to two. So, after this visit, we must visit vertices with current visited degree two or more in the punishment part of e, or b_1, b_2, b_3, or vertices in the harvest part, and then must go through the harvest part in the same manner as in previous cases, with most vertices in the harvest part getting visited degree three.

Case 4. At least one of the first two visited vertices belongs to a punishment part. Suppose we visit vertices in the clause punishment part of edge $e = \{v, w\}$.

Consider the first vertex, visited outside the punishment part that are unequal to v or w. Note that $\{v, w\}$ does not belong to a triangle. Thus, when v or w is reached, and as long as not all vertices in the punishment are visited, there is an unvisited vertex in the punishment part with current visited degree two or more. So, this first vertex outside the part and $\{v, w\}$ must belong to $\{b_1, b_2, b_3\}$. Moreover, the next vertex of this type must be a neighbor of this first vertex. We arrive now in a case analysis, similar to Case 3, and we get that the maximum visited degree is six. ◇

Claim 9.8 *If there does not exist a truth assignment that satisfies C, then each MCS-ordering ψ of $G_{C,k}$ that starts at v_0 has $mcslb_\psi(G'_{C,k}) \leq 6$.*

Proof: We have now shown for all types of vertices that the maximum visited degree is at most six. This ends the proof of Claim 9.8. ◇

The NP-completeness result now follows directly from Claim 9.1 and Claim 9.8. □

Corollary 10 *For each constant $c > 1$, there is no polynomial time approximation algorithm for* Max MCS-LB *and* Max MCS-LB with prescribed start *with approximation ratio at most c, unless P=NP.*

Proof: This is a consequence of the proofs of Theorems 8 and 9. From these proofs, we see that it is NP-hard to distinguish the cases that the maximum visited degree is five (six) or that we can get a visited degree of k, for some free to choose constant k. □

The constant of 7 in Theorem 9 possibly can be lowered to 6 with a more refined analysis. We conjecture the problem to be also NP-complete for $k = 4$, $k = 5$, and $k = 6$. The case $k = 3$ is also open, but a polynomial time algorithm for that case may be a better conjecture. Max MCS-LB is trivial in the case that $k = 2$ by the following easy result.

Proposition 11 *The following statements are equivalent.*

(i). G is a forest.

(ii). There is an MCS-ordering that gives maximum visited degree 1 on G.

(iii). Every MCS-ordering gives maximum visited degree 1 on G.

As the construction in the proofs of Theorems 8 and 9 is exponential in k, an approximation algorithm with a logarithmic performance ratio still may be a possibility.

Similar proofs can give NP-completeness results for related problems. We define the problems, but give the results without the proofs.

> Max MCS-LB-v
> **Instance**: Graph $G = (V, E)$, vertex $v \in V$, integer $k \leq |V|$.
> **Question:** Is there an MCS-ordering of G in which v has at least k visited neighbours.

We also can look to the minimisation variants, in which we ask for MCS-orderings whose maximum visited degree is *at most* some given integer k.

Theorem 12 Max MCS-LB-v, Min MCS-LB-v, Min MCS-LB, *and* Min MCS-LB with prescribed start *are NP-complete. For each of these problems, and each constant* $c > 0$, *there is no polynomial time approximation algorithm with approximation ratio at most c, unless P=NP.*

5 Upper bounds on the maximum visited degree

From the results in the previous section, it appears that we must resort to heuristics for obtaining good bounds for the maximum visited degree for given graphs G. With the application to compute lower bounds for treewidth in mind, most interesting are lower bounds for MCSLB(G). These will be looked at in the next section. In this section, we propose some upper bound methods. These can inform us on the quality of the heuristics for lower bounds for the maximum visited degree.

In order to obtain an upper bound on the maximum visited degree of a given graph $G = (V, E)$, we compute for each vertex $v \in V$ an upper bound on the maximum visited degree of v over all MCS-orderings of G. I.e., we let $mcslb_{\max}(G, v)$ be the maximum over all MCS-orderings ψ of G of the visited degree of v in ψ. The maximum of $mcslb_{\max}(G, v)$ taken over all vertices is an upper bound on the maximum visited degree of G.

Our first heuristic maintains for each $v \in V$ an upper bound $u(v)$ for $mcslb_{\max}(G, v)$, and tries to improve (decrease) these upper bounds stepwise. The algorithm has the following invariant:

$$mcslb_{\max}(G, v) \leq u(v) \tag{1}$$

We initialise for each vertex v, $u(v)$ to be the degree of v, $d(v)$. Clearly, $mcslb_{\max}(G, v)$ cannot be larger than $d(v)$, so the invariant holds trivially.

Procedure 1 (ImproveMCSLBMAXv) states a subroutine that tries to improve the value $u(v)$ for some vertex v. The subroutine can decrease the value $u(v)$, and outputs true if and only if $u(v)$ has been changed. The procedure starts by sorting the values $u(w)$ for the

Procedure 1 ImproveMCSLBMAXv (**Graph** G, **Vertex** v)

Compute $UN(v) = \{u(w) \mid \{v, w\} \in E\}$, and sort $UN(v)$.
Suppose $UN(v) = \{u_1, u_2, \ldots, u_d\}$, with $u_1 \leq u_2 \leq \cdots \leq u_d$.
count $= 0$;
for $j = 1$ to $u(v)$ **do**
    if ($u_{d-u(v)+j} \geq$ count) **then**
        count ++.
    end if
end for
if (count $< u(v)$) **then**
    $u(v) =$ count; return **true**
else
    return **false**
end if

set of neighbors of v. Next, as many vertices as possible that satisfy the condition stated in Lemma 2 are selected. This number then defines the new $u(v)$ and we return true on improvement.

Proposition 13 *The procedure ImproveMCSLBMAXv maintains invariant (1).*

Proof: Suppose $mcslb_{\max}(G, v) = k$. By Lemma 2, v has neighbors $w_1, \ldots, w_k$, with $i - 1 \leq mcslb_{\max}(G, w_i) \leq u(w_i)$ for all i, $1 \leq i \leq k$.

We claim that with induction, for each i, $0 \leq i \leq k$, the value of count after the $u(v) - k + i$th iteration is at least i. This trivially holds for $i = 0$. Suppose it holds for $i - 1$, $1 \leq i \leq k$. As for each $i' \in \{i, \ldots, k\}$, $u(w_{i'}) \geq i - 1$, the $k - i + 1$ neighbors of v with largest u-value each have an u-value that is at least $i - 1$, and hence $u_{d-k+i} \geq i - 1$, with u_{d-k+i} as in the procedure. If before the $u(v) - k + i$th iteration, count is at least i, the induction hypothesis trivially holds. Otherwise, count equals $i - 1$ before the $u(v) - k + i$th iteration. Now, $u_{d-u(v)+(u(v)-k+i)} = u_{d-k+i} \geq i - 1$, and hence count is increased by one in the $u(v) - k + i$th iteration, hence equals i after this iteration.

In particular, after the last $u(v)$th iteration, count is at least $k = mcslb_{\max}(G, v)$. So, after running ImproveMCSLBMAXv(G, v), we still have $u(v) \geq mcslb_{\max}(G, v)$. $\Box$

Observe that the procedure ImproveMCSLBMAXv never increases values $u(v)$. When it returns *true*, then $u(v)$ has been changed. Possibly, this can cause that neighbors of v can decrease their value of $u(v)$ with another run of ImproveMCSLBMAXv. This gives rise to the simple scheme of Procedure 2 (MCSLBMAX). The set S can be implemented with e.g., a queue or stack, with an additional mechanic (e.g., Boolean array) that prevents a vertex to be added for a second time to S when it already belongs to S.

The largest value of $u(v)$ over all $v \in V$ is an upper bound on MCSLB(G). In Section 6, we report on the upper bounds obtained with the MCSLBMAX heuristic for several graphs from applications. To improve the running time of the procedure, we always extract the vertex with minimum value of $u(v)$.

It is possible to give a variant of the procedure that gives in some cases better bounds. In

Procedure 2 MCSLBMAX (**Graph** G)

for all $v \in V$ **do**
  $u(v) = d_G(v)$.
end for
$S = V$.
while $S \neq \emptyset$ **do**
  Extract vertex v from S.
  change = ImproveMCSLBMAXv(G,v).
  if change **then**
    Add to S all neighbors of v that are not in S.
  end if
end while

this variant, we distinguish between the cases that a vertex is visited after all its neighbors or not. Define $mcslb'_{\max}(G, v)$ as the maximum over all MCS-orderings of G where v has at least one neighbor that is visited after v of the number of visited neighbors of v when v is visited.

We have a variable $u'(v)$ for each vertex $v \in V$. We initialise $u'(v)$ to $d_G(v) - 1$. $u'(v)$ must be an upperbound on $mcslb'_{\max}(G, v)$, i.e., we want that our procedure maintains the following invariant:

$$\forall v \in V : mcslb'_{\max}(G, v) \leq u'(v) \tag{2}$$

Similar as above, we have a procedure ImproveMCSLBMAXv2 that attempts to decrease the values $u'(v)$, and returns true when this value is changed. The code is identical to ImproveMCSLBMAXv, except that we have

$$UN(v) = \{u'(w) \mid \{v, w\} \in E\}$$

Similar to the proof of Proposition 13, one can show:

Proposition 14 *The procedure ImproveMCSLBMAXv2 maintains invariant (2).*

Procedure 3 (MCSLBMAX2) is similar to MCSLBMAX. We initialise $u'(v)$ to $d_G(v) - 1$ for each $v \in V$, and carry out then the same steps as in MCSLBMAX, but now with variables $u'(v)$, and calls to ImproveMCSLBMAXv2. After this, each variable $u'(v)$ still is an upper bound to $mcslb'_{\max}(G, v)$, and these variables cannot be improved by the use of ImproveMCSLBMAXv2. After this, we compute upper bounds $u(v)$ on $mcslb_{\max}(G, v)$ with help of the values $u'(v)$ for all vertices $v \in V$. Hereto, we can run a variant of ImproveMCSLBMAXv, once for each vertex, as follows:

As in the proof of Proposition 13, one can show that after running MCSLBMAX2, for all $v \in V$, $mcslb_{\max}(G, v) \leq u(v)$.

Our third upper bound heuristic gives a further refinement by looking at which neighbor of v is visited after v. Define for each pair of adjacent vertices v, w, $mcslb_{\max}(G, v, w)$ as the

Procedure 3 MCSLBMAX2 (**Graph** G)

Run MCSLBMAX2(G).
for all $v \in V$ **do**
    Compute $UN'(v) = \{u'(v) \mid \{v, x\} \in E\}$, and sort $UN'(v)$.
    Suppose $UN'(v) = \{u'_1, u'_2, \ldots, u'_d\}$, with $u'_1 \leq u'_2 \leq \cdots \leq u'_d$.
    count = 0;
    for $j = 1$ to d **do**
        if ($u'_j \geq$ count) **then**
            count ++.
        end if
    end for
    $u(v) = count$.
end for
return $u(v)$ for all $v \in V$

maximum of the visited degree of v over all MCS-orderings of G where v is visited before w. For each ordered pair of adjacent vertices v, w, we have a variable $u(v, w)$. We want that these values of these variables maintain the following invariant.

$$mcslb_{\max}(G, v, w) \leq u(v, w) \tag{3}$$

Procedure 4 (ImproveMCSLBMAXe) tries to decrease an value $u(v, w)$, and returns true when the value has been changed.

Procedure 4 ImproveMCSLBMAXe (**Graph** G, **Vertex** v, **Vertex** w)

Compute $UN(v, w) = \{u(x, v) \mid \{v, x\} \in E, x \neq w\}$, and sort $UN(v, w)$.
Suppose $UN(v, w) = \{u_1, u_2, \ldots, u_d\}$, with $u_1 \leq u_2 \leq \cdots \leq u_d$.
count = 0;
for $j = 1$ to $u(v, w)$ **do**
    if ($u_{d-u(v,w)+j} \geq$ count) **then**
        count ++.
    end if
end for
if (count $< u(v, w)$) **then**
    $u(v, w) =$ count; return **true**
else
    return **false**
end if

Proposition 15 *The procedure ImproveMCSLBMAXe maintains invariant (3).*

Procedure 5 (MCSLBMAXe) runs ImproveMCSLBMAXe until no improvements are possible, and then computes a value $u(v)$ for each $v \in V$. As before, we can show that $u(v)$ is an upper bound on $mcslb_{\max}(G, v)$. The largest value of $u(v)$ over all $v \in V$ is again an upper bound on the visited degree of G. Our experiments, discussed in Section 6 show that this third heuristic gives sometimes additional improvements on the upper bound.

Procedure 5 MCSLBMAXe (**Graph** G)

Initialise set S to be the set of all ordered pairs (v, w) with $\{v, w\} \in E$.
while $S \neq \emptyset$ **do**
  Extract a pair (v, w) from S.
  change = ImproveMCSLBMAXe(G,v,w).
  if change **then**
    Add to S all pairs (w, x) with x neighbor of w, not in S.
  end if
end while
for all $v \in V$ **do**
  Compute $UN(v) = \{u(x, v) \mid \{v, x\} \in E\}$, and sort $UN(v)$.
  Suppose $UN(v) = \{u_1, u_2, \ldots, u_d\}$, with $u_1 \leq u_2 \leq \cdots \leq u_d$.
  count = 0;
  for $j = 1$ to d **do**
    if ($u_j \geq$ count) **then**
      count ++.
    end if
  end for
  $u(v) = count$.
end for
return $u(v)$ for all $v \in V$

6 Computational results

In this section, we perform an experimental evaluation of the heuristics for the maximum visited degree, and compare these with the degeneracy and an upper bound for the treewidth. All algorithms have been tested on a large number of graphs from various application areas such as probabilistic networks, frequency assignment, travelling salesman problem and vertex colouring (see e.g. [6] for details). All algorithms have been written in C++, and the computations have been carried out on a Linux operated PC with a 3.0 GHz Intel Pentium 4 processor. All reported CPU times are in seconds. In the tables below, we present the results for some selected instances only. The result of these representative instances reflect typical behaviour for the whole set of instances. The results for the other instances can be viewed at TreewidthLIB [17].

Our experiments are divided in two parts. First, we examine the value of the visited degree of MCS-orderings obtained by different start vertices and tiebreaking rules. Second, we report on upper bounds on the maximum visited degree.

6.1 Start vertices and tiebreakers

Each MCS-ordering ψ provides a lower bound for treewidth. The start vertex of an MCS-ordering influences the final ordering directly. Computational experiments however have shown that the outcome varies only marginally depending on the start vertex. Typically, an overwhelming majority of the start vertices results in the same visited degree, with a few exceptions to lower and/or higher values.

During the ordering process, multiple vertices can have the highest visited degree, e.g., after

			$\delta D(G)$		MCS-LB						MCS-UB	
					default		max-degree		min-degree			
instance	$\|V\|$	$\|E\|$	LB	CPU	LB	CPU	LB	CPU	LB	CPU	UB	CPU
link	724	1738	4	0.01	5	3.35	4	7.22	5	7.88	25	25.13
munin1	189	366	4	0.00	4	0.21	4	0.42	4	0.40	20	1.42
munin3	1044	1745	3	0.01	4	6.28	3	13.43	4	14.24	12	33.36
pignet2	3032	7264	4	0.04	5	67.59	4	144.42	5	164.38	255	10977.05
celar06	100	350	10	0.01	**11**	0.06	10	0.15	10	0.14	**11**	0.10
celar07pp	162	764	11	0.01	12	0.18	11	0.47	12	0.41	18	0.68
graph04	200	734	6	0.01	8	0.28	6	0.61	7	0.60	57	13.77
rl5934-pp	904	1800	3	0.01	4	5.31	4	11.13	4	10.69	32	79.04
school1	385	19095	73	0.04	85	5.22	85	30.80	85	26.99	264	277.46
school1-nsh	352	14612	61	0.02	72	4.22	72	19.14	72	17.02	224	212.19
zeroin.i.1	126	4100	48	0.00	**50**	0.39	48	3.56	**50**	2.74	52	3.77

Table 1: Treewidth lower and upper bounds for selected instances

the start vertex is fixed all neighbors have the same visited degree and can be ordered next. To select the next vertex various tiebreakers can be applied.

In Table 1 we compare three different tiebreakers for selecting the next vertex among all vertices of highest visited degree. For each tiebreaker we report the largest visited degree taken over all possible start vertices. The CPU times are the sum over all possible start vertices. The column 'default' present the results without a specific tiebreaker, i.e., the first vertex with highest current visited degree is selected. The 'max-degree' tiebreaker selects the vertex with maximum degree among the vertices with highest visited degree, whereas the 'min-degree' tiebreaker selects the vertex with minimum degree. The idea behind the maximum degree strategy is to push the visited degree for as much vertices as possible. On the other hand, the minimum degree strategy tries to keep a vertex of high degree as long as possible unvisited such that more and more neighbors are visited before it, and thus, its visited degree increases.

The figures in Table 1 as well as for the remaining instances show that the 'default' tiebreaker outperforms the other tiebreakers with 162 times the best value (out of 165 instances). The 'min-degree' tiebreaker is second best with 141 times the best value, whereas the 'max-degree' obtains only 82 times this value.

For comparison, the degeneracy $\delta D(G)$ is also included in the table as well as the treewidth upper bound computed by the MCS heuristic [10]. As proved in Theorem 6, the visited degree for any MCS-ordering is always at least as good as the degeneracy. The experiments show that in almost half the cases the best visited degree that is obtained is one better than the degeneracy, cf. Figure 11. The computation times of the MCS-LB heuristics are larger than those for the degeneracy, but still very small.

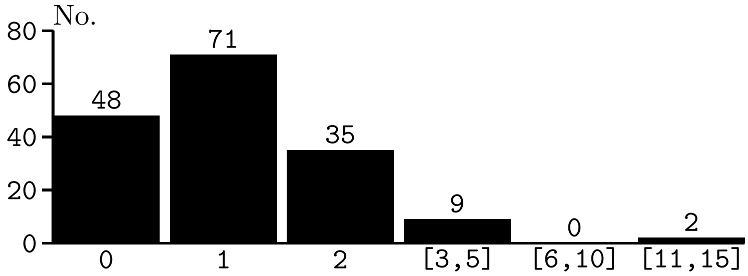

Figure 11: Histogramm for difference between MCSLB and $\delta D(G)$

				MCSLBMAX		MCSLBMAX2		MCSLBMAXe		best	
instance	$\|V\|$	$\|E\|$	$\Delta(G)$	value	CPU	value	CPU	value	CPU	MCSLB	TW-UB
link	724	1738	31	11	0.01	10	0.01	10	0.03	5	13
munin1	189	366	18	10	0.00	9	0.00	8	0.01	4	11
munin3	1044	1745	73	12	0.00	11	0.01	10	0.05	4	7
pignet2	3032	7264	232	17	0.03	16	0.03	14	3.26	5	135
celar06	100	350	31	17	0.00	16	0.00	16	0.01	11	11
celar07pp	162	764	39	26	0.01	25	0.00	24	0.02	12	18
graph04	200	734	15	13	0.00	12	0.00	11	0.01	8	55
rl5934-pp	904	1800	7	7	0.01	6	0.00	6	0.02	4	23
school1	385	19095	282	172	0.02	171	0.04	170	9.53	85	188
school1-nsh	352	14612	232	142	0.01	141	0.02	140	2.69	72	162
zeroin.i.1	126	4100	111	105	0.01	104	0.00	104	0.61	50	50

Table 2: Upper bounds on the MCSLB for selected instances

In some cases the MCS-bound equals the best treewidth upper bound (**bold** values, cf. Table 2 if MCS-UB is larger) and thus the reported value is the treewidth of those graphs. In total 30 instances could be solved to optimality by this lower bound, wheres with the degeneracy only 15 instances could be solved to optimality. In other cases the gap between lower and upper bound is still large, e.g., for instance 'pignet2'. As stated in Proposition 7, MCSLB(G) is not closed under taking subgraphs or minors. In two recent studies on the impact of edge contraction on treewidth lower bounds, we showed that the MCSLB lower bound as well as other lower bounds can be improved substantially by computing them over selected minors [6, 12].

6.2 Upper bounds on MCSLB

In Section 5 we have reported on three ways to compute an upper bound on the maximum visited degree. All three methods as well as the maximum degree $\Delta(G)$, the actual best value achieved (cf. Table 1) and the best treewidth upper bound for selected instances are reported in Table 2. The maximum degree of each graph is reported since the algorithm to compute $u(v)$ is initialised with the degree $d_G(v)$. Table 2 shows that in several cases the final maximum of $u(v)$ over all vertices is significantly smaller than the maximum degree. Only in cases where the maximum degree is close to the treewidth, only minor improvement could be achieved. If we sum over all 165 instances the maximum degree equals 9678, whereas MCSLBMAX gives an summed upper bound of 5099, MCSLBMAX2 4965, and MCSLBMAXe 4896. The difference between the first and second improvement step is at most one, whereas between the second and third improvement step the difference is two in exceptional cases, e.g., instance 'pignet2'.

Compared to the actually computed visited degrees, there is either space for increasing the maximum visited degree or the upper bounds are not tight. Where MCSLBMAXe sums up to 4896, the MCSLB values sum up to only 2551. For some instances, the non-tightness of the upper bounds is supported by the upper bounds for treewidth. For about half the instances this is true.

Regardless whether or not these upper bounds for MCSLB are tight, the results show that they have limited explanatory power. For those probabilistic networks where the gap between lower and upper bound is large, it cannot be closed by computing the best visited degree over all MCS-orderings. For the frequency assignment graphs this could be the case, but the values are in fact useless since they are larger than the treewidth upper bound.

7 Conclusions

In this paper, we analysed the lower bound on the treewidth, introduced by Lucena [14], based on Maximum Cardinality Search. While computing the MCS-ordering with a maximum visited degree is NP-hard, we see that in practice, an arbitrary MCS-ordering gives reasonable results. A method to obtain upper bounds on the maximum visited degree shows that in several cases, an arbitrary MCS-ordering gives a visited degree that is not far from that of the best MCS-ordering.

Comparing the visited degree lower bound with other lower bounds for treewidth, we see that it gives bounds that are at least as good as the degeneracy (termed MMD in some papers), while it still can be computed very fast. In [6], we combine the method with contracting edges, giving a further improvement of the bound. Still, on many graphs, there are large differences between the lower bounds that can be obtained in this way and the actual treewidth: for instance, on planar graphs, the treewidth can be $\Omega(\sqrt{n})$ while an MCS-ordering has visited degree bounded by $O(\log n)$. So, the search for further lower bound heuristics for treewidth remains important and interesting.

Several interesting theoretical questions are left open in this paper. We mention a few. What is the complexity of Max MCSLB when k is 3, 4, 5, or 6? (We conjecture NP-completeness when $k = 4$, $k = 5$, and $k = 6$, and polynomial time solvability when $k = 3$.) Can we find an approximation algorithm for Max MCSLB with performance ratio $O(\log n)$? Can we solve the Max MCSLB problem exactly on interesting graph classes, like planar graphs or permutation graphs?

References

[1] E. Amir. Efficient approximations for triangulation of minimum treewidth. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pages 7–15, 2001.

[2] A. Becker and D. Geiger. A sufficiently fast algorithm for finding close to optimal clique trees. *Artificial Intelligence*, 125:3–17, 2001.

[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 and A. M. C. A. Koster. Safe separators for treewidth. Technical Report UU-CS-2003-027, Institute for Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands, 2003. Extended abstract in proceedings ALENEX'04.

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

[6] H. L. Bodlaender, A. M. C. A. Koster, and T. Wolle. Contraction and treewidth lower bounds. In S. Albers and T. Radzik, editors, *Proceedings 12th Annual European Symposium on Algorithms, ESA2004*, pages 628–639. Springer, Lecture Notes in Computer Science, vol. 3221, 2004.

[7] 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.

[8] F. v. d. Eijkhof and H. L. Bodlaender. Safe reduction rules for weighted treewidth. In L. Kučera, editor, *Proceedings 28th Int. Workshop on Graph Theoretic Concepts in Computer Science, WG'02*, pages 176–185. Springer Verlag, Lecture Notes in Computer Science, vol. 2573, 2002.

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

[10] 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.

[11] 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.

[12] A. M. C. A. Koster, T. Wolle, and H. L. Bodlaender. Degree-based treewidth lower bounds. Technical Report UU-CS-2004-050, Institute for Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands, 2004.

[13] 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.

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

[15] H. Röhrig. Tree decomposition: A feasibility study. Master's thesis, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1998.

[16] 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.

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