The Max-back Ordering

- A section from the unfinished Master Thesis of

Mette Hagenborg Eskesen

Department of Mathematics & Computer Science
University of Southern Denmark, Odense University

September 27, 2001

Given an undirected multigraph $G = (V, E)$, it is well known, that the edge-connectivity $\lambda(G)$ of G can be found using $n - 1$ max-flow computations. In this section we will present an alternative approach, in which we compute $\lambda(G)$ using only degree-comparisons and edge-contractions. Our main tool is a special ordering, here called a *max-back ordering*, of the vertices of G. All graphs considered in this section will be undirected.

Subsequently we will also show how to find a sparse certificate for the edge-connectivity $\lambda(G)$ of G, i.e. a $\lambda(G)$-edge-connected spanning subgraph $G^* = (V, E^*)$ of G, where $E^* \subseteq E$ and $|E^*| \leq \lambda(G)|V|$, using the same special ordering.

The presentation given below is inspired by a presentation given at SDU, Odense University, by Tibor Jordán in September 1996.

We start with a few definitions and some properties of the structures they define.

Definition 1 (Max-back Ordering) *A* max-back ordering *of a given undirected multigraph G is an ordering $v_1, v_2, \ldots, v_n$ of the vertices of G, satisfying the inequality*

$$d(V_i, v_{i+1}) \geq d(V_i, v_j),$$

for all indices i and j, $1 \leq i < j \leq n$. The set V_i is defined by $V_i = \{v_1, v_2, \ldots, v_i\}$.

An example of a max-back ordering can be seen in Figure 1 (a) on page 4.

Lemma 1 *A max-back ordering of a given undirected multigraph $G = (V, E)$ can be found in $O(|V| \log |V| + |E'|)$ time, where E' is the set of edges in the corresponding simple graph.*

Proof of Lemma 1 Let $G = (V, E)$ be an undirected multigraph. It follows from the definition above, that we can find a max-back ordering of G by choosing $v_1 \in V$ arbitrarily and then successive choose $v_{i+1} \in V$ as a vertex maximizing $d(V_i, v_{i+1})$.

We will pursue the above idea, which bears a lot of similarities to the famous *Single-Source Shortest-Path* algorithm by Dijkstra (a description of which can be found in [CLR90]), but first we will review our perception of G. Dealing with multigraphs is usually not very practical when our goal is a polynomial time algorithm, so in the following we will consider G, not as a multigraph, but as a simple weighted graph $G' = (V, E', W)$ with edge-weights depicting the

multiplicity of the edges in G. In this way we can represent G' by an array of $|V|$ adjacency lists, where the weight of the edge $vu \in E'$, is simply stored with vertex u in v's adjacency list – and vice versa.

Having Dijkstra's algorithm in mind, we now implement the algorithm suggested above, using a priority queue Q containing all the vertices v in $V - V_i$, keyed by the sum of the weights of the edges between v and V_i in G'. Initially we insert all the vertices of V in Q, and give them all key-value 0. In each iterative step we then extract the maximum keyed vertex v from Q, insert it into V_i and increase the key-value of each of v's neighbours u by the weight of the edge $vu \in E'$.

The complexity of the algorithm is given by the total complexity of the operations on Q. Implementing Q with a Fibonacci heap, we have $|V|$ *Insert* operations of amortized cost $O(1)$, $|V|$ *ExtractMax* operations taking $O(\log|V|)$ amortized time and at most $|E'|$ *IncreaseKey* operations of amortized cost $O(1)$. Hence a total cost of $O(|V| \times 1 + |V| \times \log|V| + |E'| \times 1)$ as required. □

An ordering fulfilling Definition 1 is sometimes called a *maximum adjacency* ordering, a *maximum cardinality* ordering or a *legal* ordering. Here however, we have chosen the term *max-back* ordering (as it was denoted in Tibor Jordán's presentation) to illustrate the greedy choice of v_{i+1}, as being a vertex *max*imizing the number of edges *back* to the set V_i of previously chosen vertices.

According to Frank, Ibaraki & Nagamochi [FIN93] max-back orderings (though this term was not used), where first investigated by R. E. Tarjan and M. Yannakakis [TY84] in connection with chordal graphs.

Definition 2 (Continuous ordering) *An ordering $v_1, v_2, \ldots, v_n$ of the vertices of a multigraph G is said to be* continuous*, if every connected component C of G possesses the following properties:*

(i) the vertices of C have consecutive indices in the ordering.

(ii) every vertex of C (except the one with the smallest index) is adjacent to a vertex with a lower index.

In particular we see, that if $v_1, v_2, \ldots, v_n$ is a continuous ordering of the vertices of some multigraph G, then v_n and v_{n-1} belong to the same connected component in G, if and only if $d(v_n) > 0$.

Lemma 2 *Every max-back ordering of a given multigraph G, is continuous.*

An illustrative example is given in Figure 1 (b) on page 4.

Proof of Lemma 2 Let G be a multigraph and let $v_1, v_2, \ldots, v_n$ be an arbitrary max-back ordering of the vertices of G. We then need to verify, that the ordering possesses property *(i)* and *(ii)* stated in Definition 2.

Property (i):
Assume by contradiction the existence of a connected component C in G in which the indices of the vertices do not form an interval and choose (without loss of generality) C to contain the smallest index among such components. This gives us two indices i and h, such that $1 \leq i < i+1 < h \leq n$, $v_i \in C$, $v_{i+1} \notin C$ and $v_h \in C$.

Let C' denote the connected component of G containing v_{i+1}. By the choice of C and the fact that $v_i \in C$, C' contains no vertex v_l, with $1 \leq l \leq i$. This implies, that $d(V_i, v_{i+1}) = 0$.

Since v_i and v_h belong to the same connected component, we have the existence of a path between v_i and v_h. In particular we therefore have the existence of a path between V_i and v_h giving us a vertex v_j, $i+1 < j \leq n$, such that $d(V_i, v_j) \geq 1$. Hence $d(V_i, v_{i+1}) < d(V_i, v_j)$, contradicting the definition of a max-back ordering.

Property (ii):
Assume by contradiction the existence of a connected component C in G with vertex-set $\{v_k, v_{k+1}, \ldots, v_l\}$, $1 < k+1 < l \leq n$, where some vertex v_{i+1}, $k < i+1 < l$, do not have an edge to a vertex with a smaller index. Note that $i+1 \neq l$, since the vertex v_l has to have an backwards edge in order for C to be connected.

By choice of v_{i+1} $d(V_i, v_{i+1}) = 0$. But since C is connected, we must have, that $d(\{v_k, v_{k+1}, \ldots, v_i\}, \{v_{i+1}, v_{i+2}, \ldots, v_l\}) > 0$, giving us a vertex $v_j \in \{v_{i+2}, v_{i+3}, \ldots, v_l\}$ such that $0 < d(\{v_k, v_{k+1}, \ldots, v_i\}, v_j) \leq d(V_i, v_j)$. But then $d(V_i, v_{i+1}) < d(V_i, v_j)$ contradicting the definition of a max-back ordering. □

Definition 3 (Max-back forest) *Let $G = (V, E)$ be a multigraph. The* forest corresponding to a given max-back ordering $v_1, v_2, \ldots, v_n$ of G, *is defined as the simple graph $F = (V, E')$, where $E' = \{v_i v_j \mid v_i v_j \in E \ \wedge \ i < j \ \wedge \ d(V_{i-1}, v_j) = 0\}$.*

It is clear from the definition, that given a multigraph $G = (V, E)$ and a max-back ordering $v_1, v_2, \ldots, v_n$ of G we can easily find the corresponding max-back forest by a sequence of greedy choices, in which we, for each vertex $v \in V$, include the edge incident to v, which reaches as far *backwards* in the ordering as possible. In other words, the idea is to go through the vertices of V (in any order) and for each vertex v_i, to include in F an edge $v_i v_j$ from E, $j < i$, minimizing the index j (if more than one edge fulfill the criteria, we only include one of these edges). If in some step, there is no edge incident to v, which reaches backwards in the ordering, no edge is included in F in that step.

That the defined forest indeed *is* a forest, can be verified by establishing, that F contains no cycles and at most $n-1$ edges. The latter, however, follows directly from the above construction of F, since we include at most one edge for each of the vertices $v_2, \ldots, v_n$ and none for the vertex v_1.

To see that F contains no cycles, we assume the existence of a cycle and let v be the vertex with the highest index in that cycle. The choice of v implies that it has at least two adjacent vertices w and u in F, both with indices less than i. However, this contradicts the construction of F, according to which we only include one of the edges wv and uv.

It will be useful to note, that the spanning forest constructed above is maximal in G, i.e. the addition of any edge from E/F to F would create a cycle. This is easily seen, since (by the above construction) the number of edges in F is given by the number of vertices in G minus the number of connected components in G. This number equals the size of a maximum forest in G, hence F is maximum and must therefore also be maximal. An obvious, yet useful, consequence of this observation is that F gives rise to a spanning tree in each of the connected components of G.

An example of a max-back forest is given in Figure 1 (c).

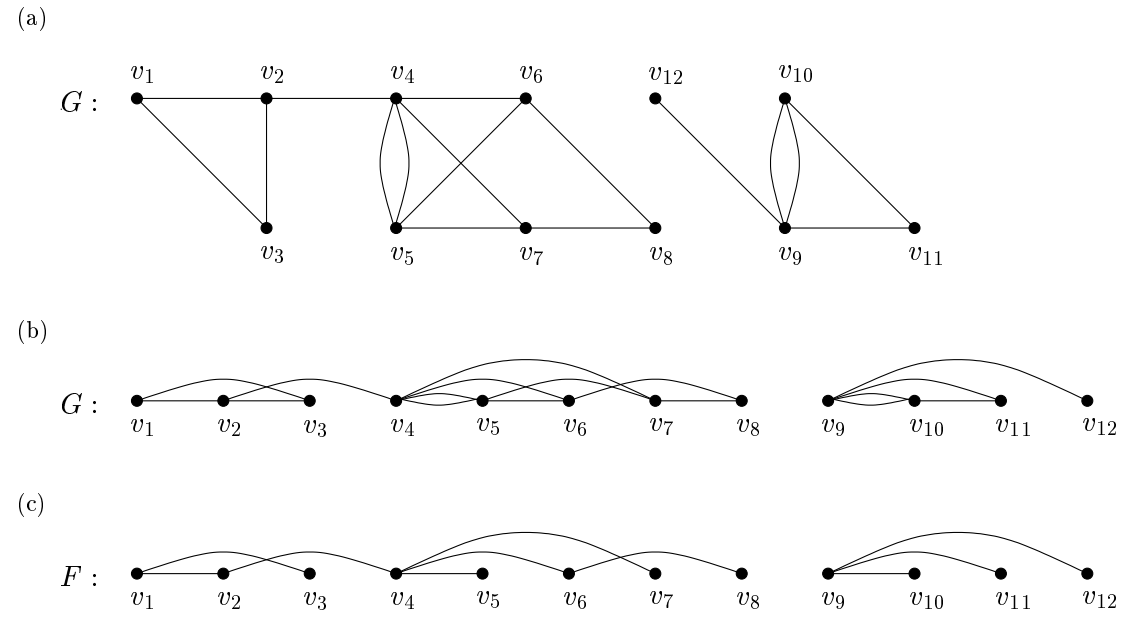

Figure 1: (a) $v_1, v_2, \ldots, v_{12}$ is an example of a max-back ordering of the vertices of the shown graph G. (b) G drawn in a way illustrating that the ordering from (a) is continuous. (c) The forest F corresponding to the max-back ordering in question.

Lemma 3 *If $v_1, v_2, \ldots, v_n$ is a max-back ordering of a given multigraph G, it is also a max-back ordering of the graph $G - E[F]$, where F denotes the forest corresponding to the max-back ordering, and $E[F]$ specifies the edge-set of that forest.*

Proof of Lemma 3 Let G be a multigraph, let $v_1, v_2, \ldots, v_n$ be a max-back ordering of G and assume by contradiction, that $v_1, v_2, \ldots, v_n$ is not a max-back ordering of $G - E[F]$.

The assumption gives us the existence of two indices i and j, $1 < i + 1 < j \leq n$, such that $d_{G-E[F]}(V_i, v_{i+1}) < d_{G-E[F]}(V_i, v_j)$. Consequently $d_{G-E[F]}(V_i, v_j) \geq 1$, implying that $d_G(V_i, v_j) \geq 1$. Since $v_1, v_2, \ldots, v_n$ is a max-back ordering in G and $i + 1 < j$, the latter furthermore gives us that $d_G(V_i, v_{i+1}) \geq 1$.

We can conclude that, in G, both v_{i+1} and v_j have at least one vertex in V_i, i.e. they both have at least one edge going backwards in the ordering. Hence, the definition of a max-back ordering and the forest corresponding to it gives the inequality $d_{G-E[F]}(V_i, v_{i+1}) = d_G(V_i, v_{i+1}) - 1 \geq d_G(V_i, v_j) - 1 = d_{G-E[F]}(V_i, v_j)$, contradicting our choice of i and j. □

Theorem 4 *For any max-back ordering $v_1, v_2, \ldots, v_n$ of a given multigraph G, we have that*

$$\lambda(v_{n-1}, v_n) = d(v_n).$$

Proof of Theorem 4 Let G be a multigraph and let $v_1, v_2, \ldots, v_n$ be an arbitrary max-back ordering of G. Clearly $d(v_n) \geq \lambda(v_{n-1}, v_n)$, since v_n can be the end-vertex of at most $d(v_n)$ edge-disjoint (v_{n-1}, v_n)-paths. For the same reason the theorem is trivial, if $d(v_n) = 0$, so we may assume that $d(v_n) = k > 0$.

Use the notation F_1 to specify the forest of G corresponding to the max-back ordering $v_1, v_2, \ldots, v_n$ and let F_i specify the forest of graph $G - \cup_{j=1}^{i-1} E[F_j]$, $2 \leq i \leq k$, corresponding

to the same max-back ordering. Recall that Lemma 3 ensures the existence of each of the forests $F_1, F_2, \ldots, F_{k-1}$ and F_k.

Since $d_G(v_n) = k$, the definition of the forest corresponding to a max-back ordering gives, that $d_{F_i}(v_n) = 1$, for all $1 \leq i \leq k$. Hence the degree of v_n is nonzero in all of the graphs $G - \cup_{j=1}^{i-1} E[F_j]$, $2 \leq i \leq k$, so Lemma 2 combined with the remark after Definition 2 imply that v_{n-1} and v_n belong to the same connected component in each of these graph (as well as in G). Therefore v_{n-1} and v_n must belong to the same tree in each of the edge-disjoint forests F_i, $1 \leq i \leq k$, giving k edge-disjoint (v_{n-1}, v_n)-paths in G. Consequently $\lambda(v_{n-1}, v_n) \geq k = d(v_n)$ and we are done. □

Definition 4 (G^{vw}) *Let $G = (V, E)$ be a multigraph with distinct vertices $v, w \in V$. We now let G^{vw} denote the multigraph obtained from G, by removing all vw-edges and* identifying v with w, *so that the resulting vertex z is incident with those edges (other than the vw-edges), that where originally incident with either v or w.*

Figure 2: Example of a graph G in which $\lambda(G) < \lambda(G^{vw})$. Notice how the cut $(\{x, y, v\}, \{w\})$ simply does not exist in G^{vw}

Theorem 5 *Let $G = (V, E)$ be an undirected multigraph on at least 3 vertices, then*

$$\lambda(G) = min\{\lambda(G^{vw}), \lambda_G(v, w)\}$$

for all $v, w \in V$.

Proof of Theorem 5 Let $G = (V, E)$ be an undirected multigraph and let $v, w \in V$ be two arbitrary vertices of G. Obviously $\lambda(G) \leq \lambda_G(v, w)$, since $\lambda(G) = min_{x,y \in V} \lambda_G(x, y)$.

Since the operation of identifying v with w eliminates some of the possible cuts in G (all the cuts separating v and w) and does not create new ones, every cut in G^{vw} must be a cut in G. Thus $\lambda(G) \leq \lambda(G^{vw})$ and we have the inequality $\lambda(G) \leq min\{\lambda(G^{vw}), \lambda_G(v, w)\}$. Figure 2 illustrates, how a (minimum) cut in G might not exist in G^{vw}.

Let $(S, \overline{S})$ be a minimum cut in G. If $(S, \overline{S})$ does *not* separate v and w, then $(S, \overline{S})$ is also a minimum cut in G^{vw}. Hence $\lambda(G^{vw}) = \lambda(G) \leq \lambda_G(v, w)$. If $(S, \overline{S})$ *does* separate v and w in G it is not a valid cut in G^{vw}, and we have, that $\lambda_G(v, w) = \lambda(G) \leq \lambda(G^{vw})$. □

Theorem 4 and Theorem 5 combined, more or less gives us the desired algorithm for finding the edge-connectivity of a given multigraph G – without the use of maximum-flow computations. Indeed we are ready to address the main subject of this section.

The idea is to repeatedly choose a pair of vertices v and w, compute the size of a minimum (v, w)-cut and then replace G by the contracted graph G^{vw}. This way we keep track of the size of all potential minimum cuts lost in the process, and after $n - 1$ iterative steps we have

a graph consisting of only a single vertex. By Theorem 5 the edge connectivity of the original graph G is now given by the minimum size of all the minimum cuts found along the way.

A crucial observation is, that the choice of v and w is arbitrary in each step. Consequently we might as well choose them in a way that makes the size of a minimum (v, w)-cut as easy as possible to determine. By Theorem 4 we see, that the aid of a max-back ordering enables us to choose v and w so that $\lambda(v, w)$ is given by $d(w)$. Using this approach the edge-connectivity of G is clearly found as the minimum among all the values $d_{G_i}(w_i)$, that appeared during the process (here G_i and w_i denotes the graph and the chosen vertex w at the i'th iteration). In addition, a global minimum cut in the original graph G is given by the set of vertices, that has been contracted into that particular vertex w, that attains the minimum just mentioned.

As seen in Figure 3, there is no guarantee that a given max-back ordering will remain a max-back ordering through all steps of the algorithm. Furthermore, verifying whether an ordering is in fact a max-back ordering, will take just as long as to find a new max-back ordering. As a consequence hereof, the algorithm given below, will compute a new max-back ordering in each iterative step.

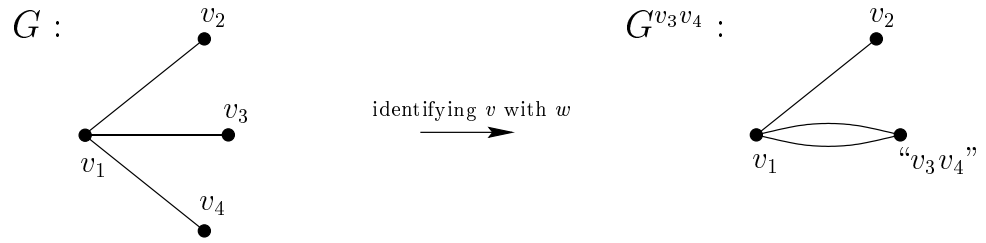

Figure 3: A simple example showing that a max-back ordering of a given graph might not be valid after the contraction of the last two vertices in the ordering. Clearly the ordering $\{v_1, v_2, \text{“}v_3v_4\text{”}\}$ is *not* a max-back ordering of $G^{v_3v_4}$, since $d(v_1, v_2) < d(v_1, \text{“}v_3v_4\text{”})$.

ALGORITHM FOR FINDING $\lambda(G)$:

INPUT: A multigraph G
OUTPUT: $k = \lambda(G)$ and a minimum cut $(S, \overline{S})$ in G

$k := \infty$
$S := \emptyset$
$n := |V(G)|$
while $n > 1$
  find a max-back ordering $v_1, v_2, \ldots, v_n$ of G
  if $d(v_n) < k$
    $k := d(v_n)$
    $S := \{v_1, v_2, \ldots, v_{n-1}\}$
  $G := G^{v_{n-1}v_n}$
  $n := n - 1$
return(k, S)

Figure 4 shows a step-by-step example of the algorithm in use.

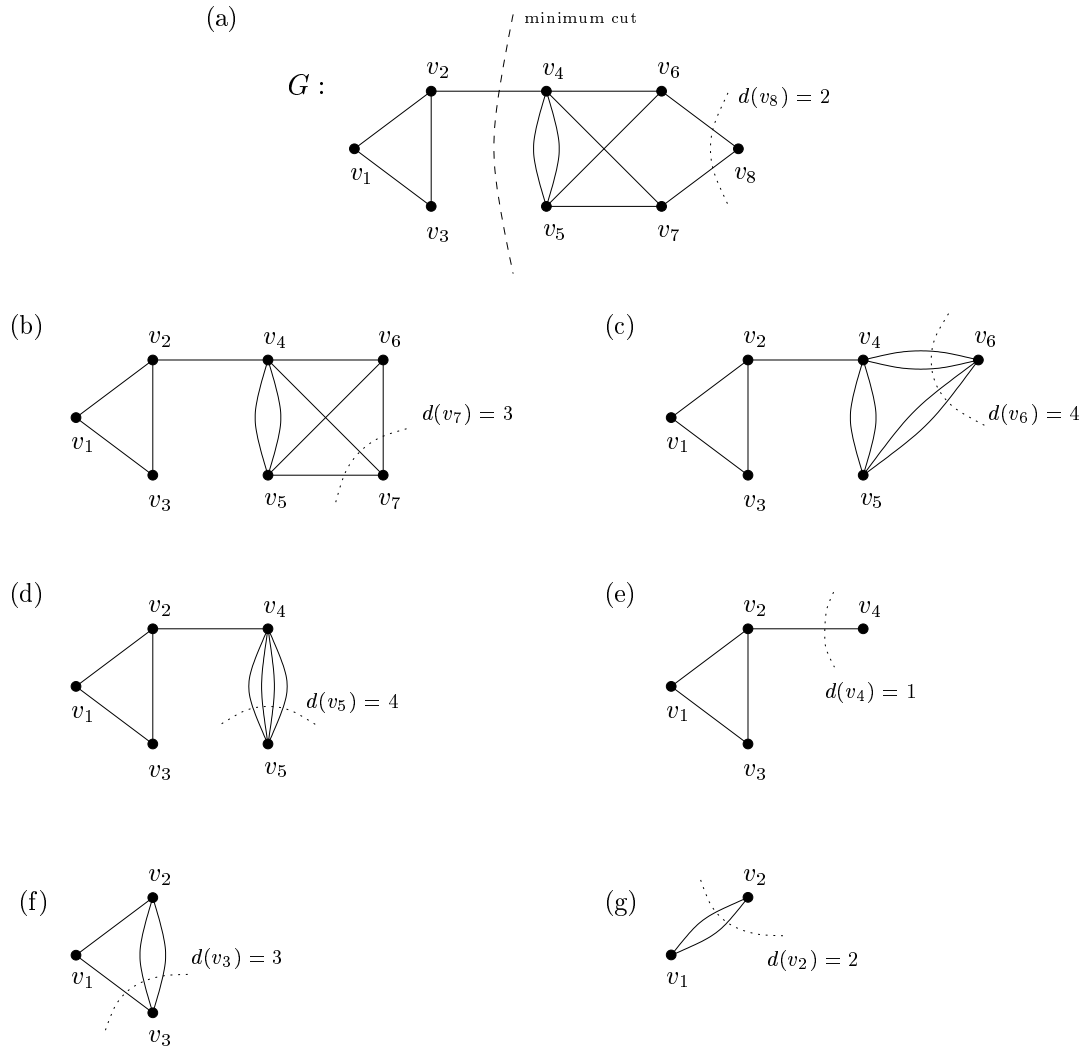

Figure 4: A step-by-step example of how the algorithm for finding $\lambda(G)$ works. The graph and the given max-back ordering have been taken from Figure 1. The ordering is characteristic, since it is valid through all the iterative steps of the algorithm (however the algorithm will reproduce it in each iteration). The algorithm finds, that $\lambda(G) = min\{2, 3, 4, 4, 1, 3, 2\} = 1$, and that $(\{v_1, v_2, v_3\}, \{v_4, v_5, \ldots, v_8\})$ is a minimum cut in the graph – as indicated in Figure (a).

Theorem 6 *Given an undirected multigraph $G = (V, E)$, we can find the edge-connectivity $\lambda(G)$ of G as well as a minimum cut in the graph in time $O(|V|^2 log|V| + |V||E'|)$ time without the use of flow-computations.*

E' denotes the set of edges in the corresponding simple graph.

Proof of Theorem 6 Let $G = (V, E)$ be a multigraph. That the algorithm given above actually finds $\lambda(G)$ along with a minimum cut $(S, \overline{S})$ in G – and does so without the use of flow-computations – should be established by now, so we turn our attention to the complexity of the algorithm.

There is no doubt, that the most expensive operations used in the algorithm is to find a max-back ordering, so evidently the complexity of the algorithm is dominated by the $|V| - 1$

times, this operation is carried out. By Lemma 1 we therefore conclude that the total time complexity of the algorithm is given by $O(\,(|V|-1)\times(|V|log|V|+|E'|)\,)=O(|V|^2log|V|+|V||E'|)$. □

To the best of our knowledge, one of the fastest known flow-algorithms for determining the edge-connectivity of a multigraph $G=(V,E)$ is a $O(\,|V||E'|\,)$ time algorithm presented by Matula in 1987 [AMO93]. Obviously the algorithm presented here is faster for $|E'|\in\Omega(\,|V|log|V|\,)$.

As mentioned in the opening, the concept of a max-back ordering can also be used to find a sparse certificate for the edge-connectivity of a given multigraph. To do so, we will need the following lemma.

Lemma 7 (Sparse certificate) *Let G be a k-edge-connected multigraph. Let F_1 be a maximal spanning forest in G and let F_i be a maximal spanning forest in $G-\cup_{j=1}^{i-1}E[F_j]$, $2\le i\le k$. The graph $H=\cup_{i=1}^{k}F_i$ is then k-edge-connected.*

Proof of Lemma 7 Let $G=(V,E)$ and F_i, $1\le i\le k$, be as described in the theorem. Then let H be the union of the k forests F_i, $1\le i\le k$, and assume by contradiction that H is not k-edge-connected.

The assumption gives us the existence of a cut $(S,\overline{S})$, $S\subset V$, of size at most $k-1$ in H. Since G *is* k-edge-connected, the size of $(S,\overline{S})$ in G is at least k. Consequently we must have two distinct vertices $v\in S$ and $w\in\overline{S}$, such that $vw\in E[G]$, $vw\notin E[H]$ and $\lambda_H(v,w)<k$.

Since $vw\notin E[H]$ we have $vw\in G-\cup_{j=1}^{i-1}E[F_j]$, $2\le i\le k$, placing v and w in a common connected component in each of the graphs G and $G-\cup_{j=1}^{i-1}E[F_i]$, $2\le i\le k$. But then, since the forests F_i, $1\le i\le k$, are maximal, v and w must also belong to a common connected component (a tree) in each of the forests. Thus we get k edge-disjoint (v,w)-paths in H, contradicting the choice of v and w. □

According to Lemma 7, it is possible to find a sparse certificate G^* for the k-edge-connectivity of a given multigraph G, by repeatedly finding an arbitrary maximal forest F in G and then replacing G by the graph $G-E[F]$. After k iterative steps, the sparse certificate G^* is given as the union of the k forests found along the way.

This algorithm seems simple and straightforward, but, as we shall see below, it can be made even simpler if we choose to consider forests belonging to some max-back ordering of G, instead of just arbitrary forests.

By Lemma 3 it will not be necessary to produce more than one max-back ordering of G, since the ordering will remain valid as a max-back ordering throughout the process. An interesting result hereof, is that we might as well create G^* in one step, instead of k steps where we remove only one forest at a time:

Find a max-back ordering $\{v_1,v_2,\ldots,v_n\}$ of G. For each vertex $v\in V$, include in G^* those k edges $(v_i,v)\in E$ with the smallest index $1\le i<j$. If there is fewer than k such edges, just take as many as there are.

Theorem 8 *Let $G=(V,E)$ be a k-edge-connected multigraph. In time $O(|V|\log|V|+|E'|)$ we can find a k-edge-connected spanning subgraph $G^*=(V,E^*)$, such that $|E^*|\le k|V|$.*

E' denotes the set of edges in the simple graph corresponding to G.

Proof of Theorem 8 Let $G = (V, E)$ be a k-edge-connected multigraph, and assume that we have found a max-back ordering $v_1, v_2, \ldots, v_n$ of the vertices of G.

In order to find a sparse certificate for the edge-connectivity of G, we will more or less follow the algorithmic ideas discussed above, but instead of going through the vertices of V in an arbitrary order, we will construct G^* by looking at each edge[†] once, and then determine whether or not it should be a part of E^*. Using this approach, it is of course crucial, that we choose the right order in which to examine the edge-set of G and it is here the max-back ordering will be of tremendous help.

In the text preceding the theorem, we saw that for each vertex v_i in V, we wish to include those k edges incident to v_i, that reaches as far back as possible in the max-back ordering. Hence we start by examining the edges incident to v_1, then those edges incident to v_2 that are not incident to v_1, followed by those edges incident to v_3 that are not incident to v_1 or v_2 and so forth.

Note that in each step we only examine edges, that reaches forward in the max-back ordering, and recall that in G^* each vertex must have at most k edges reaching backwards but an arbitrary number of edges reaching forward.

An edge v_iv_j, $1 \leq i < j \leq n$, is to be included in E^* if we have not yet included k edges incident to v_j. If we have already included k edges incident to v_j, each of those edges reaches a vertex v_k, where $1 \leq k \leq i$, indicating that our algorithm must be correct.

Analyzing the algorithm for finding a max-back ordering (from the proof of Lemma 1), we see that we can construct the max-back ordering and the spanning subgraph $G^* = (V, E^*)$ simultaneously, since the algorithm from Lemma 1 examine each of the edges in E' in the exact same order, as we just did in the proof above. Hence, by Lemma 1, we can find G^* in time $O(|V| \log |V| + |E'|)$.

As for the size of G^*, we recall that G^* is constructed as the union of k spanning forest in G, or equivalently as the union of at most k backward-edges from each vertex in V. It is therefore obvious that the edge-set E^* of G^* cannot have more than $k(|V| - 1)$ edges. □

Using a somewhat similar approach, Nagamochi and Ibaraki [NI92] have presented a linear-time algorithm for finding a sparse certificate for the edge-connectivity $\lambda(G)$ of a given multigraph G. It is easy to verify that their $O(|V| + |E'|)$ algorithm produces a max-back ordering as well as a sparse certificate, so by their result, we can actually get an $O(|V|^2 + |V||E'|)$ algorithm for finding $\lambda(G)$ as opposed to the $O(|V|^2 log|V| + |V||E'|)$ algorithm from Theorem 6.

References

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* PRENTICEHALL, New Jersey, 1993.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* McGraw-Hill Book Company, Massachusetts, 1990.

[FIN93] A. Frank, T. Ibaraki, and H. Nagamochi. On sparse subgraphs preserving connectivity properties. *Journal of Graph Theory*, 17(3):pp. 275–281, 1993.

[†]Having the proof of Lemma 1 in mind, it should be obvious how we can consider G to be a simple weighted graph, with edge-weights depicting the multiplicity of the original edges. We do this in order to keep the algorithm polynomial.

[NI92] H. Nagamochi and T. Ibaraki. A linear-time algirithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7:pp.583–596, 1992.

[TY84] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Computing*, 13(3):pp.566–579, 1984.