Simple Computation of st-Edge- and st-Numberings from Ear Decompositions

Lena Schlipf
LG Theoretische Informatik
FernUniversität in Hagen, Germany

Jens M. Schmidt*
Institute of Mathematics
TU Ilmenau, Germany

Abstract

We propose simple algorithms for computing *st*-numberings and *st*-edge-numberings of graphs with running time $O(m)$. Unlike previous serial algorithms, these are not dependent on an initially chosen DFS-tree. Instead, we compute *st*-(edge-)numberings that are consistent with *any* open ear decomposition D of a graph in the sense that every ear of D is numbered increasingly or decreasingly.

Recent applications need such *st*-numberings, and the only two linear-time algorithms that are known for this task use a complicated order data structure as black box. We avoid using this data structure by introducing a new and light-weight numbering scheme. In addition, we greatly simplify the recent algorithms for computing (the much less known) *st*-edge-numberings.

1 Introduction

st-Numberings (also known as *(1,1)-orders*) and their relatives *st*-orientations (also known as $(1,1)$-*edge-orders* and *bipolar orientations*) are fundamental tools for problems in graph drawing (such as planarity testing, visibility representations and orthogonal embeddings), routing (such as independent spanning trees) and graph partitioning. The papers [10, 13] list a wealth of applications.

For an edge st of a graph G, an *st-numbering* $<$ of G is a total order $v_1 < \cdots < v_n$ of the vertices such that $s = v_1$, $t = v_n$, and every other vertex has both a larger and smaller neighbor with respect to $<$. Every *st*-numbering defines an *st-orientation* (i.e. an acyclic orientation such that s and t are the only vertices having indegree and outdegree 0, respectively) by orienting every edge to the largest of its two endpoints. Conversely, every *st*-orientation O may be transformed back to some *st*-numbering by topologically sorting O. Since both transformations can easily be computed in linear time, all the results of this paper also hold for *st*-orientations.

Several linear-time algorithms are known for computing *st*-numberings: The first was found in 1976 by Even and Tarjan [7, 8], and then slightly simplified by Ebert [6]. Further simplifications were given in 1986 by Tarjan [14] and in 2002 by Brandes [4]. All

*This research is supported by the grant SCHM 3186/1-1 (270450205) from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation).

these algorithms are inherently based on an initially chosen depth-first search (*DFS*) tree. Maon, Schieber and Vishkin [9] showed how to compute *st*-numberings efficiently in parallel. Their algorithm uses open ear decompositions and implies also a linear-time serial algorithm, but is considerably more involved than any of the above algorithms.

Thus, all simple linear-time algorithms known so far are special in the sense that they compute *st*-numberings that inherently depend on an initially chosen DFS-tree. Let an *st*-numbering be *consistent* with an open ear decomposition D of a graph G if it numbers the vertices of every ear of D either increasingly or decreasingly (consistency of *st*-edge-numberings may be defined similarly). Let T be any initially fixed DFS-tree. Then there is an open ear decomposition that naturally depends on T (see e.g. [12]), and most of the algorithms above compute only *st*-numberings that are consistent with this special open ear decomposition. However, several applications (e.g. the ones in [2, 5, 13, 11]) need more general *st*-numberings that are consistent with an *arbitrary* given open ear decomposition D. The following is a very simple and probably folklore approach to obtain these *st*-numberings by falling back on *st*-orientations (see e.g. [9, Section 3.1] and [13, Application 1]): Compute an open ear decomposition D, orient the first cycle from s to t, and orient every following ear such that acyclicity is preserved; since the reachability relation is always a poset, an appropriate orientation for the next ear is guaranteed to exist.

Although this more general approach for computing *st*-numberings can be made directly into a linear-time algorithm [13, Application 1] (we give a concise description of this algorithm in the last section), this algorithm is not simple, as it uses the complicated *order data structure* of [3, 15] to identify which endpoint of a new ear is of minimal value.

There are two main results in this paper. First, we give the first simple linear-time algorithm that, given any open ear decomposition D, computes an *st*-numbering that is consistent with D. We avoid using the order data structure by using a new and light-weight numbering scheme. Despite its generality, the algorithm does not seem to be more complicated than any known ones.

Second, we consider *st-edge-numberings* (also known as $(1,1)$-*edge-orders*), which are considerably less known than their vertex-counterparts, but were recently used in applications (e.g., for constructing edge-independent spanning trees [11]). In fact, only two results deal with *st*-edge-numberings so far: In [1], it is sketched that, analogous to the situation for the vertex case, *st*-edge-numberings may be computed from ear decompositions (without making the runtime precise). In [11], this was used for a linear-time algorithm, which however needed again the order data structure as black box. We present the first *simple* linear-time algorithm that computes *st*-edge-numberings that are consistent with any given ear decomposition. This makes the application in [11] a lot more practicable and easy to implement.

2 Preliminaries

We use standard graph-theoretic terminology and consider only graphs that are finite, undirected and do not contain self-loops; however, we allow parallel edges.

Definition 1 ([16]). An *ear decomposition* of a graph G is a sequence $(P_0, P_1, \ldots, P_k)$

of subgraphs of G whose edge-sets partition E such that P_0 is a cycle and every P_i, $1 \leq i \leq k$, is either a path that intersects $P_0 \cup \cdots \cup P_{i-1}$ in exactly its endpoints or a cycle that intersects $P_0 \cup \cdots \cup P_{i-1}$ in exactly one vertex. Every P_i is called an *ear*.

An ear decomposition D is called *open* if all of its ears except P_0 are paths. According to Whitney [16], every ear decomposition has exactly $m-n+1$ ears (we set $m := |E|$). For any i, let $G_i := (V_i, E_i) := P_0 \cup \cdots \cup P_i$. For any ear $P_i \neq P_0$, let $inner(P_i) := V(P_i) - V(G_{i-1})$ be the set of *inner vertices* of P_i (for P_0, we define every vertex of P_0 as inner vertex). Hence, every vertex of G is an inner vertex of exactly one ear, which implies that the inner vertex sets of the ears of any ear decomposition partition V. For a vertex v, let $birth_D(v)$ be the index i such that P_i contains v as inner vertex. Whenever D is clear from the context, we will omit the subscript D.

There is a very simple algorithm that computes a structure from which both an ear decomposition and an open ear decomposition (if exists) can be "read off":

Theorem 2 ([12])**.** *For any edge $st \in G$, an ear decomposition and an open ear decomposition of G (if exists, respectively) such that $st \in P_0$ can be computed in time $O(m)$.*

For every ear P_i, $0 < i \leq m-n$, we choose an arbitrary endpoint p_i of P_i as *representative* of P_i and let q_i be the other (not necessarily different) endpoint of P_i (see Figure 1). For P_0, we set $p_0 := q_0 := s$. We denote the vertices of P_i (consecutively along P_i and starting from the representative vertex p_i) by $p_i = v_0^i, v_1^i, \ldots, v_{k_i+1}^i = q_i$ (if P_i is a cycle, we omit q_i in this list); hence, P_i has k_i inner vertices if P_i is a path, and $k_i + 1$ vertices otherwise.

3 st-Numberings from Open Ear Decompositions

Let $D = (P_0, P_1, \ldots, P_{m-n})$ be an open ear decomposition of a graph G such that $st \in P_0$ (e.g., computed by Theorem 2; this step is not much more involved than computing an initial DFS-tree for the classical algorithms). The idea for computing an *st*-numbering from D is now to modify the *st*-orientation method explained in the introduction such that vertex numbers change only in a well-defined way; to achieve the latter we assign an interval to every vertex instead of one number.

For the intervals, we first define the binary order relation *depend* on vertices. Let every inner vertex of P_i *depend* on its representative p_i and take the transitive closure of this relation, so that, for every three vertices a, b and c such that a depends on b and b depends on c, also a *depends* on c. Clearly, the dependence relation is a strict poset. Let the *weight* $w(v)$ of any vertex $v \in V$ be the number of all vertices that depend on v (see Figure 1).

Theorem 3. *Let $D = (P_0, P_1, \ldots, P_{m-n})$ be an open ear decomposition of a graph G. Then an st-numbering of G that is consistent with D can be computed in time $O(m)$.*

Proof. We may compute the weight of all vertices in linear time as follows. Initialize $w(v) := 0$ for all vertices v. For every i from $m-n$ to 1, set $w(p_i) := w(p_i) + |V(inner(P_i))| + \sum_{v \in inner(P_i)} w(v)$. Since all vertices that depend on p_i are inner vertices of some ear P_j satisfying $j \geq i$, this counts the weights correctly.

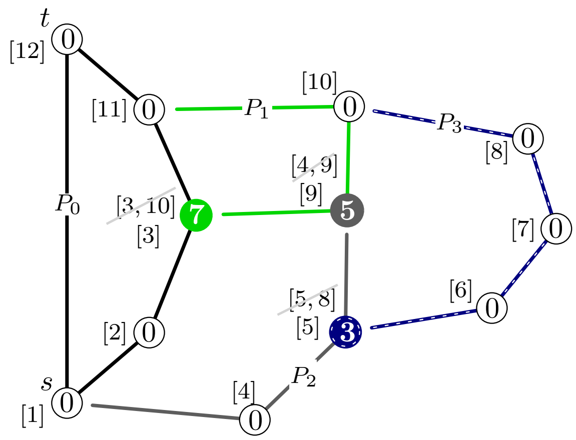

Figure 1: An open ear decomposition D, in which the representatives p_i of ears P_i are drawn solid. For every vertex v, the number at v depicts its weight, and the interval depicts $I(v)$ in the course of the proof of Theorem 3.

The crucial idea is now to use the weight of a vertex v to give v enough slack in the search for its final st-number; in more detail, one st-number for v must remain after all vertices that depend on v have been st-numbered (in particular, every interval $I(v)$ will contain at most $w(v) + 1$ numbers). To this end, we will not assign one number to every vertex v, but an interval $I(v)$ of natural numbers, in which the final number of the desired st-numbering is contained. At any point in time, the intervals of all vertices will be consecutive and pairwise disjoint. Hence, we may define for two vertices v and w that $v < w$ if some number of $I(v)$ is less than some number of $I(w)$; this gives the total order $<$ on V, which eventually will be the desired st-numbering. For an interval $I(v)$, let $I_{\min}(v)$ and $I_{\max}(v)$ be the smallest and the largest number of $I(v)$.

We now state how the intervals are chosen, beginning with the ones for the vertices of P_0 (see also Algorithm 1 for pseudo-code). We set $I(s) := [1, 1 + w(s)]$ and, for every $1 \leq j \leq k_0 + 1$, $I(v_j^0) := [I_{\max}(v_{j-1}^0) + 1, I_{\max}(v_{j-1}^0) + 1 + w(v_j^0)]$ (see Figure 1). Clearly, the intervals are pairwise disjoint and the order $<$ on these intervals is an st-numbering of P_0. Now, given such an st-numbering $<$ of G_{i-1}, we compute an st-numbering of G_i by distinguishing the following two cases for P_i.

(i) Case $p_i < q_i$ (see P_1 and P_3 in Figure 1).

We traverse P_i from q_i to p_i and move the largest values of $I(p_i)$ to intervals of the inner vertices of P_i as follows: $I(v_{k_i}^i) := [I_{\max}(p_i) - w(v_{k_i}^i), I_{\max}(p_i)]$ and, for every $1 \leq j < k_i$, $I(v_j^i) := [I_{\min}(v_{j+1}^i) - 1 - w(v_j^i), I_{\min}(v_{j+1}^i) - 1]$. As this moves $|inner(P_i)| + \sum_{j \in inner(P(i))} w(j)$ values from the interval $I(p_i)$, we update $I(p_i)$ by deleting exactly this many largest values from it. Hence, we have $I_{\max}(p_i) = I_{\min}(v_1^i) - 1$ and, as the interval of p_i was initialized using $w(p_i)$, no interval is empty. In addition, all intervals are pairwise disjoint.

We show that the intervals form an st-numbering of G_i. Deleting the above values of $I(p_i)$ preserves that $I(p_i)$ is consecutive and does not harm the st-numbering of G_{i-1} at all. For the remaining inner vertices of P_i, every vertex has a smaller and a larger neighbor by construction, except for possibly the smaller neighbor of v_1^i and the larger neighbor of $v_{k_i}^i$. However, $p_i < v_1^i$ follows from $I_{\max}(p_i) = I_{\min}(v_1^i) - 1$, and $v_{k_i}^i < q_i$ follows from $p_i < q_i$ and the fact that $I(v_{k_i}^i)$ got at least one number that was previously in $I(p_i)$.

(ii) Case $p_i > q_i$ (see P_2 in Figure 1).
We traverse P_i from q_i to p_i and move the smallest values of $I(p_i)$ to intervals of the inner vertices of P_i as follows: $I(v_{k_i}^i) := [I_{\min}(p_i), I_{\min}(p_i) + w(v_{k_i}^i)]$ and, for every $1 \leq j < k_i$, $I(v_j^i) := [I_{\max}(v_{j+1}^i) + 1, I_{\max}(v_{j+1}^i) + 1 + w(v_j^i)]$. Again, this moves $|inner(P_i)| + \sum_{j \in inner(P(i))} w(j)$ values from the interval $I(p_i)$, and so we update $I(p_i)$ by deleting exactly this many smallest values from it. Hence, we have $I_{\min}(p_i) = I_{\max}(v_1^i) + 1$ and, as the interval of p_i was initialized using $w(p_i)$, no interval is empty. In addition, all intervals are pairwise disjoint.
We show that the intervals form an st-numbering of G_i. Deleting the above values of $I(p_i)$ preserves that $I(p_i)$ is consecutive and does not harm the st-numbering of G_{i-1}. As above, it remains only to show that v_1^i has a larger and $v_{k_i}^i$ a smaller neighbor. However, $v_1^i < p_i$ follows from $I_{\min}(p_i) = I_{\max}(v_1^i)+1$, and $q_i < v_{k_i}^i$ follows from $p_i > q_i$ and the fact that $I(v_{k_i}^i)$ got at least one number that was previously in $I(p_i)$.

Since the above intervals can be set in total time $O(m)$ and the intervals of every open ear are consecutively either increasing or decreasing, we obtain the claim. □

4 st-Edge-Numberings from Ear Decompositions

Two edges are called *neighbors* if they share a common vertex.

Definition 4. For an edge st of a graph G, an *st-edge-numbering* $<$ of G (see Figure 2) is a total order on the edge set $E - st$ such that $m \geq 2$,
– every edge $e \neq st$, except for one incident to s, has a neighbor e' with $e' < e$ and
– every edge $e \neq st$, except for one incident to t, has a neighbor e' with $e < e'$.

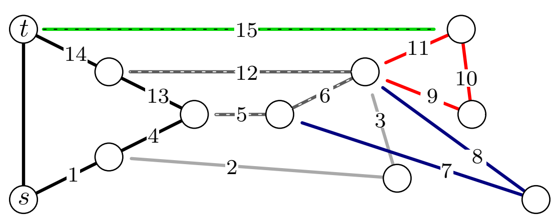

Figure 2: An st-edge-numbering of a 2-edge-connected graph G.

It is known that a graph G has an st-edge-numbering if and only if G has an ear decomposition if and only if G is 2-edge-connected [11]. Let an st-edge-numbering be *consistent* to an ear decomposition D if the edges of every ear are numbered increasingly or decreasingly.

For our algorithm, we first compute an ear decomposition $D = (P_0, P_1, \ldots, P_{m-n})$ of G such that $st \in P_0$ (e.g. by Theorem 2). As for the vertex-variant, we need a notion of

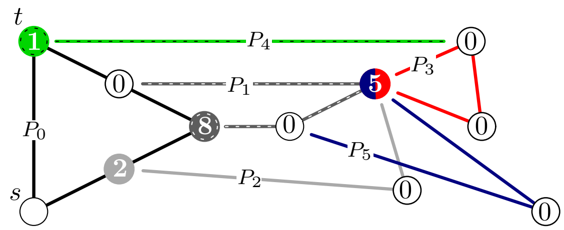

Figure 3: An ear decomposition $D = (P_0, P_1, \ldots, P_5)$ of G, in which the representatives p_i of ears P_i are drawn filled. Notice that $p_3 = p_5$. The vertex numbers depict their weights. While the edge sets $E(P_1)$, $E(P_2)$, $E(P_4)$ depend only on their representatives p_1, p_2 and p_4, respectively, $E(P_3)$ depends on p_1 and p_3, and $E(P_5)$ depends on p_1 and p_5.

dependency. Let every edge of P_i *depend* on the vertex p_i. Recursively, for every edge $e \in P_j$ that depends on some representative $p_i \notin P_0$ (this implies $i \leq j$), let e also *depend* on $p_{birth(p_i)}$ (see Figure 3). Let the *weight* $w(v)$ of any vertex $v \in V$ be the number of all edges that depend on v.

Theorem 5. *Let $D = (P_0, P_1, \ldots, P_{m-n})$ be an ear decomposition of a graph G. Then an st-edge-numbering of G that is consistent with D can be computed in time $O(m)$.*

Proof. We may compute the weight of all vertices in linear time as follows. Initialize $w(v) := 0$ for all vertices v. For every i from $m - n$ to 1, set $w(p_i) := w(p_i) + |E(P_i)| + \sum_{v \in inner(P_i)} w(v)$. Since all edges that depend on p_i are in some ear P_j satisfying $j \geq i$, this counts the weights correctly.

We encode the desired total order $<$ on edges by the function $\pi : E \to \mathbb{N}$. Differently as in the vertex-variant, we will assign these numbers directly rather than approximating them by intervals. In order to avoid having to renumber edges, we will ensure as *Invariant 1* that the numbers of every two neighbored edges e and e' of any ear differ by one plus the number of dependent edges stored at their common incident vertex v, i.e. $|\pi(e) - \pi(e')| = 1 + w(v)$.

We now state the precise numbering scheme and begin with the consecutive edges $e_1, \ldots, e_{k_0}$ of $P_0 - \{st\}$, where e_1 is incident to s (see also Algorithm 2 for pseudo-code). In accordance with Invariant 1, we set $\pi(e_1) := 1 + w(s)$ and $\pi(e_j) := \pi(e_{j-1}) + 1 + w(v^0_j)$ for every further edge e_j (see Figure 2). Clearly, π is an *st*-edge-numbering of P_0, so assume by induction we have one for G_{i-1}. For every vertex v of G_{i-1}, let $low(v)$ and $high(v)$ be the smaller and larger number of the two incident edges of v in $P_{birth(v)}$ (here, we define $low(s) := 0$ and $high(t) := \pi(e_{k_0}) + 1 + w(t)$). Clearly, each value $low(v)$ and $high(v)$ can be computed in constant time after $P_{birth(v)}$ has been numbered. In order to keep track of the numbers that are still available for the incident edges of a vertex v, we maintain the value $end(v)$, and initialize it with $high(v)$. For every vertex $v \in G_i$, we will preserve as *Invariant 2* that $low(v) < end(v) \leq high(v)$ and the numbers in $[low(v) + 1, end(v) - 1]$ are not assigned to any edge, so that initially numbers for all edges that depend on v are available. In particular, the numbers (if any) in $[1, end(s) - 1]$ are smaller than any number that is assigned to an edge, and $high(t) - 1$ is the largest number that will be assigned to an edge. Note that all intervals $[low(v) + 1, end(v) - 1]$ for vertices $v \in G_i$ are pairwise disjoint.

Given the st-edge-numbering for G_{i-1}, we compute one for G_i by distinguishing the following cases for P_i. Let $e_1, \dots, e_r$ be the consecutive edges of P_i such that $e_1 := p_i v_1^i$ and $e_r := v_{k_i}^i q_i$ ($e_1 = e_r$ is possible if P_i is an edge).

(i) Case $low(p_i) \leq low(q_i)$ (see P_1, P_2, P_3 in Figure 3).
We number the edges of P_i decreasingly from e_r to e_1 as follows: $\pi(e_r) := end(p_i) - 1$ and, for every $1 \leq j < r$, $\pi(e_j) := \pi(e_{j+1}) - 1 - w(v)$, where v is the common vertex of e_j and e_{j+1}. Since this uses all values in $[\pi(e_1), end(p_i) - 1]$, we set $end(p_i) := \pi(e_1)$. Clearly, this preserves Invariants 1 and 2.
It remains to show that this obtains an st-edge-numbering of G_i. Since we started with a valid numbering of G_{i-1} and every edge $e \notin \{e_1, e_r\}$ of P_i has clearly smaller and larger neighbors by construction, we aim for finding a smaller neighbor of e_1 and a larger neighbor of e_r. We distinguish two cases. First, let $p_i \neq s$. By Invariants 1 and 2 for p_i, e_1 has a neighbor $e' \in P_{birth(p_i)}$ that satisfies $\pi(e') = low(p_i) < end(p_i) = \pi(e_1)$ and is therefore smaller. Similarly, e_r has a neighbor $e'' \in P_{birth(q_i)}$ that satisfies $\pi(e'') = high(q_i)$. Since $\pi(e_r) < low(q_i) < high(q_i)$, e'' is a larger neighbor of e_r. Now let $p_i = s$. Then the number assigned to e_1 is smaller than any number assigned to an edge in G_i and, hence, e_1 becomes the exceptional edge incident to s that has no smaller neighbor. The previous exceptional edge incident to s has now e_1 as smaller neighbor. The argumentation that e_r has a larger neighbor is the same as in the first case.

(ii) Case $low(p_i) > low(q_i)$ (see P_4, P_5 in Figure 3).
We number the edges of P_i decreasingly from e_1 to e_r as follows: $\pi(e_1) := end(p_i) - 1$ and, for every $1 < j \leq r$, $\pi(e_j) := \pi(e_{j-1}) - 1 - w(v)$, where v is the common vertex of e_j and e_{j-1}. Since this uses all values in $[\pi(e_r), end(p_i) - 1]$, we set $end(p_i) := \pi(e_r)$. Clearly, this preserves Invariants 1 and 2. It remains to show that this obtains an st-edge-numbering of G_i.
Since every edge $e \notin \{e_1, e_r\}$ of P_i has clearly smaller and larger neighbors by construction, we aim for finding a smaller neighbor of e_r and a larger neighbor of e_1. We distinguish between two cases. First, let $p_i \neq t$. By Invariants 1 and 2 for p_i, e_1 has a neighbor $e' \in P_{birth(p_i)}$ that satisfies $\pi(e') = high(p_i) > \pi(e_1)$ and is therefore larger. Similarly, e_r has a neighbor $e'' \in P_{birth(q_i)}$ that satisfies $\pi(e'') = low(q_i)$. Since $\pi(e_r) = end(p_i) > low(p_i) > low(q_i)$, e'' is a smaller neighbor of e_r. Now let $p_i = t$. If $end(t) = high(t)$, e_1 has the largest number that is assigned to an edge in G_i and therefore becomes the exceptional edge incident to t that has no larger neighbor. The previous exceptional edge incident to t has now e_1 as larger neighbor. If $end(t) < high(t)$, then e_1 has a neighbor e' that is incident to t, satisfies $\pi(e') = high(t) - 1 > \pi(e_1)$ and is therefore larger than e_1. The argumentation that e_r has a smaller neighbor is the same as in the first case.

We note that using this approach, the two exceptional edges of Definition 4 may change over time, e.g. whenever P_i is a cycle with representative $p_i = s$. Since every step can be computed in constant time and numbers every ear consecutively, we obtain the claim. □

5 Implementation Details and Discussion

In this section, we give the pseudo-codes of both algorithms presented in the paper, and discuss the linear-time algorithm that computes st-numberings using the order data structure of [15] and [3]. For the pseudo-codes, see Algorithms 1 and 2.

We now give a concise description of the computation of st-numberings using the order data structure. Let $D = (P_0, P_1, \ldots, P_{m-n})$ be an open ear decomposition of a graph G such that $st \in P_0$ (e.g., computed by Theorem 2). Orient the cycle P_0 from s to t. For every next open ear P_i with endpoints p and q, orient P_i from p to q if and only if the orientation of G_{i-1} does not contain a directed path from q to p, and from q to p otherwise.

For $0 \leq i \leq m-n$, consider the order relation on the vertices of the oriented subgraph G_i with respect to reachability. For $i = 0$, this relation is clearly a poset. By construction, it is also a poset for all $0 \leq i \leq m-n$. Thus, all orientations are cycle-free and the orientation O of G_{m-n} is an st-orientation of G (from which an st-numbering consistent with D can be easily obtained as shown in the introduction).

Most algorithms for st-numberings are in fact based on this approach and differ only in the various ways how reachability is computed (a common trick is to use special ear decompositions that allow to choose the orientations in dependence of the orientation of former ears). The following more general approach uses an order data structure instead.

Theorem 6. *Let $D = (P_0, P_1, \ldots, P_{m-n})$ be an open ear decomposition of a graph G. Then an st-numbering of G that is consistent with D can be computed in time $O(m)$.*

Proof. We refine the approach above slightly by maintaining an st-numbering $<_i$ for the vertices of every G_i in the following way. For G_0, let $<_0$ be the total order that orders the vertices of the path $P_0 - st$ consecutively from s to t. For every $1 \leq i \leq m-n$, let p and q be the endpoints of P_i such that $p <_{i-1} q$ and orient P_i from p to q (this strictly refines the approach above). Now obtain $<_i$ from $<_{i-1}$ by adding the set of inner vertices of P_i immediately after p, ordered by the orientation of P_i. Then $<_{m-n}$ is an st-numbering of G, as shown above.

It remains to show that the running time is $O(m)$. We use the *incremental order data structure*, which maintains a total order subject to the operations of (i) *inserting* an element after a given element and (ii) *comparing* two distinct given elements by returning the one that is smaller in the order. Tsakalidis [15] and Bender et al. [3] showed such a data structure with amortized constant time per operation (the latter result also supports additional *deletions* of elements). We use the order data structure to maintain the orders $<_{i-1}$. For every new open ear P_i, this allows to compute the minimum of p and q with respect to $<_{i-1}$ in amortized constant time and thus to augment $<_{i-1}$ in amortized time proportional to $|V(P_i)|$. $\square$

Theorem 6 is not meant to give the simplest algorithm known for computing an st-numbering, as the order data structure itself is somewhat involved. Its beauty stems rather from the fact that it directly transforms the above approach with st-orientations into a linear-time algorithm in an conceptually easy way. This makes this algorithm also worthwhile for teaching in undergraduate classes; in fact, we experienced positive student feedback for first teaching Theorem 6 (to set the stage) and then Algorithm 1 (to avoid the order data structure).

References

[1] F. Annexstein, K. Berman, and R. Swaminathan. Independent spanning trees with small stretch factors. Technical Report 96-13, DIMACS, June 1996.

[2] G. D. Battista, R. Tamassia, and L. Vismara. Output-sensitive reporting of disjoint paths. *Algorithmica*, 23(4):302–340, 1999.

[3] M. A. Bender, R. Cole, E. D. Demaine, M. Farach-Colton, and J. Zito. Two simplified algorithms for maintaining order in a list. In *Proceedings of the 10th European Symposium on Algorithms (ESA'02)*, pages 152–164, 2002.

[4] U. Brandes. Eager st-Ordering. In *Proceedings of the 10th European Symposium of Algorithms (ESA'02)*, pages 247–256, 2002.

[5] J. Cheriyan and S. N. Maheshwari. Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs. *Journal of Algorithms*, 9(4):507–537, 1988.

[6] J. Ebert. st-Ordering the vertices of biconnected graphs. *Computing*, 30:19–33, 1983.

[7] S. Even and R. E. Tarjan. Computing an st-Numbering. *Theor. Comput. Sci.*, 2(3):339–344, 1976.

[8] S. Even and R. E. Tarjan. Corrigendum: Computing an st-Numbering (TCS 2(1976):339-344). *Theor. Comput. Sci.*, 4(1):123, 1977.

[9] Y. Maon, B. Schieber, and U. Vishkin. Parallel ear decomposition search (EDS) and st-numbering in graphs. *Theoretical Computer Science*, 47:277–298, 1986.

[10] C. Papamanthou and I. G. Tollis. Algorithms for computing a parameterized st-orientation. *Theoretical Computer Science*, 408(2-3):224–240, 2008. Excursions in Algorithmics: A Collection of Papers in Honor of Franco P. Preparata.

[11] L. Schlipf and J. M. Schmidt. Edge-orders. In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP'17)*, pages 75:1–75:14, 2017.

[12] J. M. Schmidt. A simple test on 2-vertex- and 2-edge-connectivity. *Information Processing Letters*, 113(7):241–244, 2013.

[13] J. M. Schmidt. Mondshein sequences (a.k.a. (2,1)-orders). *SIAM Journal on Computing*, 45(6):1985–2003, 2016.

[14] R. E. Tarjan. Two streamlined depth-first search algorithms. *Fund. Inf.*, 9:85–94, 1986.

[15] A. K. Tsakalidis. Maintaining order in a generalized linked list. *Acta Informatica*, 21(1):101–112, 1984.

[16] H. Whitney. Non-separable and planar graphs. *Transactions of the American Mathematical Society*, 34(1):339–362, 1932.

Algorithm 1 Compute an *st*-numbering of a 2-connected graph G

Compute an open ear decomposition $D = (P_0, \ldots, P_{m-n})$ of G.
For every path P_i, let $p_i = v_0^i, \ldots, v_{k_i}^i, v_{k_i+1}^i = q_i$ be the vertices of P_i.
for all $v \in V(G)$ **do** $w(v) := 0$
for $i \leftarrow m - n$ to 1 **do** $\triangleright$ compute the weights
  $w(p_i) := w(p_i) + |inner(V(P_i))| + \sum_{v \in inner(P_i)} w(v)$
$I(s) := [1, 1 + w(s)]$ $\triangleright$ initialize intervals for P_0
for $j \leftarrow 2$ to $|V(P_0)|$ in the path $P_0 - \{st\}$ **do**
  $I(v_j^0) := [I_{\max}(v_{j-1}^0) + 1, I_{\max}(v_{j-1}^0) + 1 + w(v_j^0)]$
for $i \leftarrow 1$ to $m - n$ **do** $\triangleright$ compute intervals for G_i
  if $p_i < q_i$ **then**
    $I(v_{k_i}^i) := [I_{\max}(p_i) - w(v_{k_i}^i), I_{\max}(p_i)]$
    for $j \leftarrow k_i - 1$ to 1 **do**
      $I(v_j^i) := [I_{\min}(v_{j+1}^i) - 1 - w(v_j^i), I_{\min}(v_{j+1}^i) - 1]$
    $I_{\max}(p_i) := I_{\min}(v_1^i) - 1$
  else $\triangleright$ $p_i > q_i$
    $I(v_{k_i}^i) := [I_{\min}(p_i), I_{\min}(p_i) + w(v_{k_i}^i)]$
    for $j \leftarrow k_i - 1$ to 1 **do**
      $I(v_j^i) := [I_{\max}(v_{j+1}^i) + 1, I_{\max}(v_{j+1}^i) + 1 + w(v_j^i)]$
    $I_{\min}(p_i) := I_{\max}(v_1^i) + 1$

Algorithm 2 Compute an *st*-edge-numbering of a 2-edge-connected graph G

Compute an ear decomposition $D = \{P_0, \dots, P_{m-n}\}$ of G.
for all $v \in V(G)$ **do** $w(v) := 0$
for $i \leftarrow m - n$ to 1 **do** ▷ compute the weights
  $w(p_i) = w(p_i) + |E(P_i)| + \sum_{v \in inner(P_i)} w(v)$
Let $E(P_0) = \{e_1, \dots, e_r\}$.
Set $\pi(e_1) := high(s) := 1 + w(s)$, $low(s) := 0$, $low(t) := \pi(e_r)$ and $high(t) := \pi(e_r) + 1 + w(t)$.
for $j \leftarrow 2$ to r **do** ▷ number $E(P_0)$
  $\pi(e_j) := \pi(e_{j-1}) + 1 + w(v_j^0)$
Set $low(v)$ and $high(v)$ for all $v \in V - \{s, t\}$. Set $end(v) := high(v)$ for all $v \in V$.
for $i \leftarrow 1$ to $m - n$ **do** ▷ number $E(P_i)$
  Let $e_1, \dots, e_r$ be the consecutive edges of P_i from p_i to q_i.
  if $low(p_i) \le low(q_i)$ **then**
    $\pi(e_r) := end(p_i) - 1$
    for $j \leftarrow r - 1$ to 1 **do**
      $\pi(e_j) := \pi(e_{j+1}) - 1 - w(v)$, where v is the common vertex of e_j and e_{j+1}
    Update *high* and *low*-values for $inner(P_i)$ and set $end(p_i) := \pi(e_1)$.
  else ▷ $low(p_i) < low(q_i)$
    $\pi(e_1) := end(p_i) - 1$
    for $j \leftarrow 2$ to r **do**
      $\pi(e_j) := \pi(e_{j-1}) - 1 - w(v)$, where v is the common vertex of e_j and e_{j-1}
    Update *high* and *low* for $inner(P_i)$ and set $end(p_i) := \pi(e_r)$.