



ELSEVIER

Theoretical Computer Science 203 (1998) 253–269

---

---

Theoretical  
Computer Science

---

---

## On mixed connectivity certificates

Shimon Even<sup>a,\*</sup>, Gene Itkis<sup>b,2</sup>, Sergio Rajsbaum<sup>c,3</sup>

<sup>a</sup> *Technion-Israel Inst. of Technology, Department of Computer Science, Haifa 32000, Israel*

<sup>b</sup> *NDS, PO 23012, Harzvim, Jerusalem 91235, Israel*

<sup>c</sup> *Universidad Nac. Autonoma de Mexico, Instituto de Matemáticas, Ciudad Universitaria, Mexico, D.F. 04510, Mexico*

---

### Abstract

Vertex and edge connectivity are special cases of mixed connectivity, in which all edges and a specified set of vertices play a similar role. Certificates of  $k$ -connectivity for a graph are obtained by removing a subset of its edges, while preserving its connectivity up to  $k$ .

We unify the previous work on connectivity certificates and extend it to handle mixed connectivity and multigraphs. Our treatment contributes a new insight of the pertinent structures, yielding more general results and simpler proofs. Also, we present the first communication-optimal distributed algorithm for finding mixed connectivity certificates. © 1998 Published by Elsevier Science B.V. All rights reserved

*Keywords:* Connectivity certificates; Mixed connectivity; Distributed algorithm

---

## 1. Introduction

### 1.1. Basic concepts

Let  $G = (V, E)$  be a finite undirected graph with no self-loops, and  $x, y \in V$  be a pair of distinct vertices of  $G$ . The *edge connectivity* of  $x$  and  $y$  in  $G$  is the maximum number of edge-disjoint paths connecting  $x$  and  $y$ . Similarly, their *vertex connectivity* is the maximum number of vertex-disjoint paths connecting  $x$  and  $y$ . (Each edge between  $x$  and  $y$  is such a path.)

---

\* Corresponding author. E-mail: even@cs.technion.ac.il.

<sup>1</sup> Parts of this work were done while the author visited U.N.A.M. and Boston University. Partly supported by the Fund for the Promotion of Research at the Technion.

<sup>2</sup> Part of this research was conducted in the Comp. Sci. Dept., Technion, Israel, and in Boston University. Supported by the Israeli Council for Higher Education and NSF grant CCR-90 15276.

<sup>3</sup> Part of this research was conducted while visiting the MIT Laboratory for Computer Science, and CRL Digital Equipment Corporation. Partly supported by DGAPA Projects, U.N.A.M.

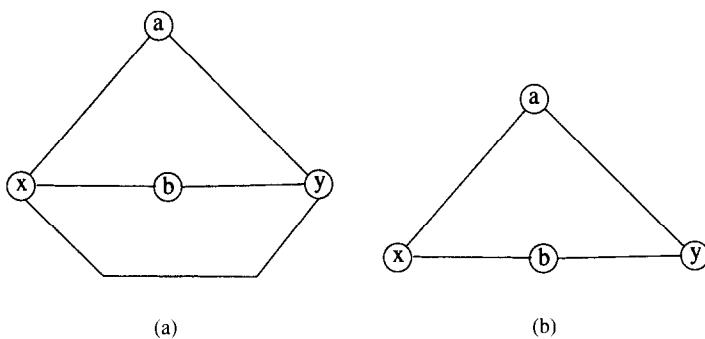


Fig. 1. Global vs. Local connectivity certificates.

Following [6], we consider a generalization of these two particular types of connectivity. Let  $S \subseteq V$ . We say that a family of paths connecting vertices  $x, y$  is *S-independent* if the paths are edge-disjoint and every element of  $S$  appears as an inner vertex in at most one of these paths. The *S-mixed connectivity*  $\lambda_S(x, y; G)$  of  $x$  and  $y$  in  $G$  is the maximum number of *S-independent* paths connecting  $x$  and  $y$  in  $G$ . The cases of  $S = \emptyset$  and  $S = V$  correspond to edge and vertex connectivity, respectively. For brevity, if  $G$  is clear from the context, we omit it. Say that  $x$  and  $y$  are *S-mixed k-connected* if  $\lambda_S(x, y) \geq k$ . This is also referred to as *local connectivity*, as opposed to global: the *global connectivity* of graph  $G$  is  $\lambda_S(G) \stackrel{\text{def}}{=} \min_{x, y \in V} \lambda_S(x, y; G)$ .  $G$  is *S-mixed k-connected* if  $\lambda_S(G) \geq k$ .

For each type of connectivity, a certificate of *k-connectivity* for  $G$  is a sub-graph preserving the connectivity up to  $k$ . Namely,  $G' = (V, E')$ ,  $E' \subseteq E$ , is a *certificate of local S-mixed k-connectivity* for  $G$  if for any two vertices  $x$  and  $y$ ,  $\lambda_S(x, y; G') \geq \min\{k, \lambda_S(x, y; G)\}$ . Similarly,  $G'$  is a *certificate of global S-mixed k-connectivity* for  $G$  if  $\lambda_S(G') \geq \min\{k, \lambda_S(G)\}$ . The *size* of  $G'$  is  $|E'|$ .

Clearly, certificates of local *k-connectivity* are also certificates of global *k-connectivity*; the opposite is generally not true. If  $\lambda_S(G) \geq k$  then a certificate for global *k-connectivity* of  $G$  is also a certificate for local *k-connectivity*. However, if  $\lambda_S(G) < k$ , a certificate for global *k-connectivity* of  $G$  may not be a certificate for local *k-connectivity*. Consider the graph  $G$  shown in Fig. 1(a), and assume  $k = 3$  (The choice of  $S$  is immaterial since there are no parallel edges, and the degree of the vertices is bounded by 3.)  $\lambda_S(G) = 2$ . However,  $\lambda_S(x, y; G) = 3$ . The subgraph  $G'$ , shown in Fig. 1(b), is a certificate for global *k-connectivity* of  $G$ , since  $\lambda_S(G') = \lambda_S(G) = 2$ , but  $\lambda_S(x, y; G') = 2$ .

Unless stated otherwise, we will speak about certificates of local connectivity.

For a *k-connected*  $G$  there is a trivial lower bound of  $k|V|/2$  on the size of a certificate of *k-connectivity*, because the degree of every vertex in a *k-connected* graph is at least  $k$ . Results of Mader [11–13] imply that every edge *k-connected* graph  $G$  contains an edge *k-connected* subgraph with  $O(k|V|)$  edges. Also, Mader’s results imply

a similar result for vertex connectivity of simple graphs. Namely, an edge  $k$ -connected graph has a certificate of global edge  $k$ -connectivity of size  $O(k|V|)$ , and a simple graph, which is vertex  $k$ -connected, has a certificate of global vertex  $k$ -connectivity of size  $O(k|V|)$ . However, as is shown below, if one follows Zykov's point of view [16], as we do, if parallel edges are allowed, the statement does not hold for vertex connectivity.

**Example.** Let  $G = (V, E)$  be a complete graph, each pair of vertices connected by  $p$  parallel edges. Then  $|E| = p|V|(|V| - 1)/2$  and  $\lambda_V(G) = p + |V| - 2$  (since any two vertices have  $p + |V| - 2$  vertex-disjoint paths between them). At the same time, the removal of any edge reduces the graph's global connectivity. In this example, for  $p \geq |V|$ , Mader's  $O(p|V|)$  bound is off by a factor of  $|V|$ .

## 1.2. Applications

Certificates with fewer edges than the original graph are useful for improving the efficiency of a number of graph algorithms. One may perform a preprocessing step to find a sparse certificate, and then run the algorithms on the certificate. For example, this method has been used to improve the sequential time complexity of testing simple undirected graphs for  $k$ -connectivity [2, 7, 14]; to improve the running time for finding three independent spanning trees [3]; to design efficient fault tolerant protocols for distributed computer networks [9], and dynamic algorithms [5].<sup>4</sup>

Sparse certificates are of special utility for the *distributed model* of computation. This model consists of a graph  $G$  with the vertices representing processors and the edges representing bidirectional communication links. There is no common memory and all communication is through messages sent along the links. The messages take a finite but arbitrary time to traverse a link. The *communication complexity* of the algorithm is the number of messages sent by the algorithm, assuming each message contains  $O(\log |E|)$  bits. For the sake of discussing the *time-complexity* of asynchronous distributed algorithms, it is customary to assume that each message is transmitted and processed within one unit of time.

The vertex (edge) connectivity of the graph is related to the number of processors (links) failures that can be tolerated by the distributed system before the network is disconnected.  $S$ -mixed connectivity allows to deal with networks where any link can fail but only processors in  $S$  are subject to failure. The number of messages sent by the distributed algorithm often depends critically on  $|E|$ . In such cases, if the

<sup>4</sup> An anonymous referee noted: "It is very easy to prove that the mixed  $k$ -connectivity certificates of this paper can be maintained dynamically in  $O(|V|)$  times per update during edge insertions and deletions, for constant  $k$ . This can be obtained by simply applying sparsification [5]". It is actually possible to do better: Using a more direct approach, amortized cost of  $O(1)$  per insertion and  $O(|V|)$  per deletion, (i.e. independent of  $k$ ) has been achieved [10].

algorithm is executed on a sparse certificate then the message complexity is reduced while preserving the number of faults that can be tolerated.

### 1.3. Previous work

Given a  $k$ -(vertex or edge) connected graph, the problem of finding a  $k$ -connected spanning subgraph, with the minimum number of edges, is NP-hard for any fixed  $k \geq 2$ ; [8] cites personal communication with F. Chung and R. Graham). Indeed, for  $k = 2$  the reduction from the Hamiltonian Cycle problem is trivial.

However, finding good approximations is possible for all  $k$ : Nagamochi and Ibaraki [14] find, in time  $O(|E|)$ , edge and vertex connectivity certificates of size  $< k|V|$  (which is within a factor of 2 from the trivial lower bound described in Section 1.1). Their algorithm, as well as others, consists of finding a sequence of forests  $F_1, F_2, \dots$  in the graph. Each forest  $F_j$  is maximal in the remaining graph  $G - \bigcup_{i=1}^{j-1} F_i$  (e.g. each connected component of the remaining graph is spanned by a tree of  $F_j$ ). This maximality alone suffices to show that  $G_k \stackrel{\text{def}}{=} \bigcup_{i=1}^k F_i$  is an edge  $k$ -connectivity certificate of size  $< k|V|$ . Moreover, if  $G$  is simple and the forests are grown according to a certain rule (see Section 4), then  $G_k$  is a vertex  $k$ -connectivity certificate as well.

Graph  $G$  is called  $S$ -simple if it has no parallel edges incident to vertices of  $S$ . Frank, Ibaraki and Nagamochi [6] show that for all  $S \subseteq V$  the algorithm of [14] applied to  $S$ -simple graphs produces certificates of  $S$ -mixed connectivity.

Cheriyani et al. [2] introduce a more flexible way of constructing certificates of vertex connectivity of size  $< k|V|$ , and use it in their distributed and parallel algorithms. They show that for simple graphs, constructing  $F_i$  in a scan-first-search manner (see Section 4.3) is sufficient to produce certificates of global vertex  $k$ -connectivity. Their distributed algorithm uses the synchronizers of [1] and runs in time  $O(k|V| \log^3 |V|)$  using  $O(k|E| + k|V| \log^3 |V|)$  messages. Every certificate obtained by the algorithm of [14] can be obtained using the scan-first-search; but as we show in the sequel, the converse does not hold. Thus, the results of [6] do not apply to the certificates of [2].

A new distributed algorithm by Thurimella [15] for computing  $O(k|V|)$  size certificates has time complexity  $O(k(D + |V|^{0.614}))$ , where  $D$  is the diameter of the network. The communication complexity of the algorithm was not analyzed.

In all previous results quoted above, the graphs are assumed to be  $S$ -simple;  $S = \emptyset$  and  $S = V$  for edge and vertex connectivity, respectively.

### 1.4. Our results

We present a general scheme for generating  $S$ -mixed  $k$ -connectivity certificates. It consists of an optimum reduction from general graphs, allowing parallel edges, to  $S$ -simple graphs, and a scheme to generate certificates of size  $< k|V|$  for  $S$ -simple graphs.<sup>5</sup> The scheme for  $S$ -simple graphs includes as special cases the results of

<sup>5</sup>Note that an optimum reduction may not produce an optimum certificate, since the certificate for the  $S$ -simple graph may not be optimum.

[2, 6, 14] and has a simpler correctness proof. This implies that the certificates of [2] are of local connectivity. We believe that the generality of this scheme as well as the simplicity of the proof contribute towards a better understanding of connectivity certificates.

For the distributive model, we present the first communication-optimal algorithm which is an implementation of the scheme above. For simple graphs it generates certificates of size  $<k|V|$  in time  $(2k+2)|V|$  using  $\leq 4|E|$  messages. In addition to the improved complexity, our algorithm is simpler and works in a more restricted single server model. A *single server* algorithm has the following property. At any time there is exactly one vertex which is active, and all activities in the network, while this vertex is in charge, are restricted to its immediate neighborhood. In the course of the computation the server travels in the network along its edges.

The rest of the paper is organized as follows. After describing some notation and basic notions in Section 2, in Section 3 we present a method for reducing the problem of finding a global connectivity certificate in a graph which may have parallel edges to  $S$ -simple graphs. In Section 4 we describe the scheme to generate certificates of size  $<k|V|$  for  $S$ -simple graphs. The reader uninterested in the problem with parallel edges can skip directly to this section, which includes at the end, Section 4.4, the distributed algorithm.

## 2. Notation and basic notions

Let  $V(E')$  be the set of vertices which are the endpoints of the edges of  $E' \subseteq E$ . For disjoint sets  $X, Y \subseteq V$ , let  $E(X, Y) \subseteq E$  denote the set of edges of  $G$  with one endpoint in  $X$  and the other in  $Y$ ;  $E(X)$  is the set of edges with an endpoint in  $X$ . We write  $E(x, Y)$  and  $E(x)$  if  $X = \{x\}$ . A path *linking* sets  $X$  and  $Y$  is a path with one endpoint in  $X$  and the other endpoint in  $Y$ ;  $X$  and  $Y$  are then said to be *linked*.  $\Gamma_G(v) \stackrel{\text{def}}{=} V(E(v)) - \{v\}$  is the set of neighbors of  $v$  in  $G$ .

Following the definitions in [6], let  $\{Z, A, B\}$  be a partition of  $V$  such that  $Z \subseteq S$ ,  $A \neq \emptyset$  and  $B \neq \emptyset$ . We say that the pair  $C = (Z, E(A, B))$  is an *S-mixed cut*. If  $\emptyset \neq A' \subseteq A$ ,  $\emptyset \neq B' \subseteq B$  then the cut  $C = (Z, E(A, B))$  *separates*  $A'$  and  $B'$ . The *size* of  $C$  is defined to be  $|C| \stackrel{\text{def}}{=} |Z| + |E(A, B)|$ . As was observed in [6], following [4], using the max-flow min-cut theorem, it is easy to derive the following theorem, which is in the spirit of Menger's theory.

**Theorem 1.** *For any  $a, b \in V$ , the minimum size of an S-mixed cut separating  $a, b$  is  $\lambda_S(a, b; G)$ .*

Henceforth, unless otherwise specified, we discuss  $S$ -mixed cuts and connectivity. So, for brevity, we omit the “ $S$ -mixed”.

### 3. Reduction to $S$ -simple graphs

Let  $\text{simple}_S(G)$  be the maximal  $S$ -simple subgraph of  $G$ . To obtain  $\text{simple}_S(G)$  from  $G$ , for each  $\{a, b\} \cap S \neq \emptyset$  replace all parallel edges between  $a$  and  $b$ , if there are any in  $G$ , by a single edge  $a-b$  in  $\text{simple}_S(G)$ .

We will reduce the construction of a global connectivity certificate for a general graph  $G$  (with arbitrary parallel edges) to finding a local connectivity certificate for  $\text{simple}_S(G)$ . The following lemma will be useful:

**Lemma 1.** *Let  $G'$  be a certificate of local  $k$ -connectivity for  $G$ . Then any cut  $C'$  in  $G'$  is either a cut in  $G$  or  $|C'| \geq k$ .*

**Proof.** Let  $G' = (V, E' \subseteq E)$  be a certificate of local  $k$ -connectivity for  $G = (V, E)$ , and let  $C' = (Z, E'(A, B))$  and  $C = (Z, E(A, B))$  be cuts in  $G'$  and  $G$ , respectively. If  $E'(A, B) = E(A, B)$ , then  $C' = C$ , so  $C'$  is a cut in  $G$ . Otherwise, there exists  $a \stackrel{e}{-} b \in E(A, B) - E'(A, B)$ . Let  $\tilde{C}$  be a minimum cut separating  $a$  and  $b$  in  $G'$ , so  $\lambda_S(a, b; G') = |\tilde{C}| \leq |C'|$ . Clearly,  $\lambda_S(a, b; G') < \lambda_S(a, b; G)$  (since adding  $e$  increases the size of any cut separating  $a$  and  $b$ ). But since  $G'$  is a certificate of local  $k$ -connectivity for  $G$ ,  $\lambda_S(a, b; G') \geq \min\{k, \lambda_S(a, b; G)\}$ , and therefore,  $\lambda_S(a, b; G') \geq k$ . Thus,  $|C'| \geq k$ .  $\square$

#### 3.1. Global connectivity

Next we define the  $S$ -degree of a vertex. In this definition we want  $d_S(v; G)$  to be a locally determined upper bound on the least connectivity of vertex  $v$  to other vertices. Thus, if  $V - v \subseteq S$ , and  $\Gamma_G(v) = V - v$ , then the connectivity between  $v$  and any other vertex  $u$  is bounded by  $|V| - 2 + |E(v, u)|$ , and the least of these is  $|V| - 2 + \min_{w \neq v} |E(v, w)|$ . If  $V - v \not\subseteq S$  then the connectivity between  $v$  and another vertex  $u \in V - S$  is bounded by  $|E(v, V - S)|$ , plus  $|\Gamma_G(v) \cap S|$ . If  $\Gamma_G(v) \neq V - v$  then again, the connectivity between  $v$  and  $u \in (V - v) - \Gamma_G(v)$  is bounded by  $|E(v, V - S)| + |\Gamma_G(v) \cap S|$ . Thus, we define the  $S$ -degree of a vertex  $v$ ,  $d_S(v; G)$ , to be  $|V| - 2 + \min_{w \neq v} |E(v, w)|$  if  $\Gamma_G(v) = V - v \subseteq S$  and  $|E(v, V - S)| + |\Gamma_G(v) \cap S|$  otherwise. The  $S$ -degree of graph  $G$  is  $d_S(G) \stackrel{\text{def}}{=} \min_{v \in V} \{d_S(v; G)\}$ . (The traditional definition of degree coincides with  $d_{\emptyset}$ .) Clearly,  $d_S(G) \geq \lambda_S(G)$ .

Let  $m = \min\{k, d_S(G)\}$ , and  $G' = (V, E' \subseteq E)$ . The following procedure *Incr\_Deg* ( $G', G, m$ ) increases  $d_S(G')$  to be  $\geq m$  by adding edges of  $G$  to  $G'$ :

**Procedure Incr\_Deg**( $G', G, m$ ):

1. for every  $v \in V$ , starting with those  $\in V - S$ , do
2.   while  $d_S(v; G') < m$  do
3.     if  $\Gamma_G(v) - \Gamma_{G'}(v) \neq \emptyset$  then add some  $e \in E(v, \Gamma_G(v) - \Gamma_{G'}(v))$  to  $E'$
4.     else if  $\Gamma_{G'}(v) = V - v \subseteq S$  then
5.       for every  $u \in V - v$  such that  $|E'(v, u)| = \min_{w \neq v} |E'(v, w)|$
6.        add some  $e \in E(v, u) - E'$  to  $E'$
7.     else add some  $e \in E(v, V - S) - E'$  to  $E'$

**Remark.** The purpose of starting, in line 1, with the vertices in  $V - S$ , is to make this Procedure optimal. It adds the least number of edges to make  $d_S(G') \geq m$ . As noted before this does not necessarily make the resulting certificate optimal, for the certificate of the  $S$ -simple subgraph may not be optimal. Due to the limited significance of this optimality, we omit its proof. All other claims we make, concerning Procedure *Incr\_Deg*, remain valid if the preference of the vertices in  $V - S$  is removed.

**Lemma 2.** *After executing  $\text{Incr\_Deg}(G', G, m)$ ,  $d_S(G') \geq m$ .*

**Proof.** It suffices to show that if  $d_S(v; G') < m$  (see line 2) then  $d_S(v; G')$  is increased in lines 3–7.

If the condition of line 3 holds, then  $\Gamma_{G'}(v) \neq V - v$  and by definition  $d_S(v; G') = |E'(v, V - S)| + |\Gamma_{G'}(v) \cap S|$ . Thus, the addition of  $e$ , per line 3, increases one of the terms. If the new  $\Gamma_{G'}(v)$  is still  $\neq V - v$ , or  $V - v \not\subseteq S$ , then clearly  $d_S(v; G')$  has been increased. If after the edge is added  $\Gamma_{G'}(v) = V - v \subseteq S$ , then the new  $d_S(v; G') = |V| - 2 + \min_{w \neq v} |E'(v, w)|$ , while the previous  $d_S(v; G')$ , using the previous  $\Gamma_{G'}(v)$ , was  $|\Gamma_{G'}(v)| \leq |V| - 2$ . Since now  $\Gamma_{G'}(v) = V - v$ , the new value of  $\min_{w \neq v} |E'(v, w)|$  is positive. This concludes the proof that if the condition of line 3 holds then  $d_S(v; G')$  increases. In the remainder of the proof we may assume that  $\Gamma_G(v) = \Gamma_{G'}(v)$ .

Now, suppose the condition in line 4 holds:  $\Gamma_G(v) = \Gamma_{G'}(v) = V - v \subseteq S$ . So  $d_S(v; G') = |V| - 2 + \min_{w \neq v} |E'(v, w)|$  and  $d_S(v; G) = |V| - 2 + \min_{w \neq v} |E(v, w)|$ . But  $d_S(v; G') < m \leq d_S(v; G)$  implies  $\min_{w \neq v} |E'(v, w)| < \min_{w \neq v} |E(v, w)|$ . So, for every  $u$  as in line 5, there is a new edge which can be added (line 6). Thus the for loop of lines 5–6 increases  $d_S(v; G')$ .

Finally, assume neither the condition of line 3, nor that of line 4 holds, so either  $\Gamma_{G'}(v) = \Gamma_G(v) \neq V - v$  or  $V - v \not\subseteq S$ . In both cases,  $d_S(v; G') = |E'(v, V - S)| + |\Gamma_{G'}(v) \cap S| < d_S(v; G) = |E(v, V - S)| + |\Gamma_G(v) \cap S|$ , and so  $|E'(v, V - S)| < |E(v, V - S)|$ . Thus, line 7 is applicable and it increases  $d_S(v; G')$ .  $\square$

*Reduction.* To obtain a certificate of global  $S$ -mixed  $k$ -connectivity for  $G$  first obtain a certificate  $G'$  of local  $S$ -mixed  $m$ -connectivity for  $\text{simple}_S(G)$ , and then apply  $\text{Incr\_Deg}(G', G, m)$  to turn  $G'$  into the desired certificate.

The definition of  $m$  and the fact that  $\lambda_S(G) \leq d_S(G)$  imply that  $\min\{m, \lambda_S(G)\} = \min\{k, \lambda_S(G)\}$ . Therefore, there is no difference between certificates for global  $k$ - and  $m$ -connectivity. Now, the following theorem, together with Lemma 2, implies the correctness of the above reduction.

**Theorem 2.** *Let  $G' = (V, E' \subseteq E)$ ,  $d_S(G') \geq m$ , and let  $G'$  contain as a subgraph a certificate of local  $m$ -connectivity for  $\text{simple}_S(G)$ . Then  $\lambda_S(G') \geq \min\{m, \lambda_S(G)\}$ .*

**Proof.** The Theorem holds trivially if  $\lambda_S(G') \geq m$ . If  $\lambda_S(G') < m$  we need to show that  $\lambda_S(G') \geq \lambda_S(G)$ . Select a minimum cut  $C = (Z, E'(A, B))$  in  $G'$  (i.e.  $|C| = \lambda_S(G') < m$ ) such that for any minimum cut  $C' = (Z', E'(A', B'))$  of  $G'$ ,  $|E(A, B)| \leq |E(A', B')|$ . We show that  $E(A, B) = E'(A, B)$ , so  $C$  is also a cut of  $G$ . Thus,  $\lambda_S(G) \leq |C| = \lambda_S(G')$ .

Suppose  $E'(A, B) \neq E(A, B)$ . Consider the cuts imposed by the partition  $(Z, A, B)$  on  $\text{simple}_S(G)$  and its certificate. Lemma 1 implies that the two cuts are equal, since the latter is contained in  $C$  and thus its size is  $\leq |C| < m$ . The only edges in  $G$  but not in  $\text{simple}_S(G)$  are parallel edges with an endpoint in  $S$ . Therefore, for any  $a \overset{e}{-} b \in E(A, B) - E'(A, B)$ , there exists  $a \overset{e'}{-} b \in E'(A, B)$  and  $\{a, b\} \cap S \neq \emptyset$ . Let  $a \in A, b \in B$ , and w.l.o.g. let  $a \in S$ .

Assume  $|A| = 1$ ; i.e.  $E = \{a\}$ . By definition of  $d_S(a, G')$ ,  $d_S(a, G') \leq |C(Z, E'(a, B))|$ . Since  $d_S(a, G') \geq m$ ,  $|C(Z, E'(a, B))| \geq m$ , contradicting  $|C| < m$ . Thus,  $|A| > 1$ , and  $|B| > 1$  as well. So  $C' \stackrel{\text{def}}{=} (Z \cup \{a\}, E'(A - \{a\}, B))$ , is a minimum cut of  $G'$ :  $|C'| \leq |C|$ , since  $C'$  is obtained from  $C$  by adding one vertex, while removing at least one edge. But obviously  $E(A - \{a\}, B) \subset E(A, B)$ , and  $e, e' \notin E(A - \{a\}, B)$ , so  $|E(A - \{a\}, B)| < |E(A, B)|$ , contradicting the choice of  $C$ .  $\square$

If  $\lambda_S(G') \geq k$  then  $d_S(G') \geq k$ . So, at least for  $k$ -connected graphs, it is necessary to increase the  $S$ -degree of the simplification's certificate (as done by *Incr\_Deg*), in order to turn it into a certificate for  $G$ . Surprisingly, it turns out to be sufficient as well.

### 3.2. Local connectivity

In general the reduction of Section 3.1 may not produce certificates of *local* connectivity. However, in some specific cases, obtaining certificates of local connectivity is easy. If  $\{x, y\} \cap S = \emptyset$  then  $\lambda_S(x, y; G) = \lambda_S(x, y; \text{simple}_S(G))$ . Therefore, if a certificate of local connectivity between vertices of  $V - S$  is required, then a certificate of local connectivity of  $\text{simple}_S(G)$  can be used. A connectivity certificate for a specific pair  $s, t$  can be constructed by defining  $S' = S - \{s, t\}$  and obtaining a certificate of local connectivity for  $\text{simple}_{S'}(G)$ .

For general graphs, a certificate of local connectivity for  $G$  can be constructed as follows. First, find a certificate of local connectivity for  $\text{simple}_S(G)$ . Next, flesh out each edge  $x \overset{e}{-} y$  of the certificate to have  $\max\{k, |E(x, y)|\}$  parallel edges between  $x$  and  $y$ . This could increase the certificate size, unnecessarily, by a factor of  $k$  above the minimum. The problem of efficiently reducing the task of finding sparse certificates of local connectivity to finding sparse certificates for  $S$ -simple graphs remains open.

## 4. Certificates for $S$ -simple graphs

Let  $G = (V, E)$  be an  $S$ -simple graph. Let  $\{F_i\}$  be a sequence of mutually disjoint non-empty sets of edges partitioning  $E$ , and define  $E_k \stackrel{\text{def}}{=} \bigcup_{1 \leq i \leq k} F_i$ ,  $\overline{E}_k \stackrel{\text{def}}{=} E - E_k$ ,  $G_k \stackrel{\text{def}}{=} (V, E_k)$ . Note that  $\overline{E}_0 = E$ . For every  $i \geq 1$ , let  $F_i$  be a maximal forest in  $(V, \overline{E}_{i-1})$ . Then each forest  $F_i$  consists of a set of spanning trees, one for each connected component of  $(V, \overline{E}_{i-1})$ . The next lemma follows directly from the maximality of the forests [14].

**Lemma 3.** *If  $u$  and  $v$  are connected in  $F_j$ , then for each  $i$ ,  $1 \leq i < j$ ,  $u$  and  $v$  are connected in  $F_i$ .*

Lemma 3 is sufficient to prove that  $G_k$  is a certificate of local edge (i.e.  $\emptyset$ -mixed)  $k$ -connectivity of size  $< k|V|$ . We skip the proof since it is a special case of the mixed connectivity results which follow. If  $S \neq \emptyset$ ,  $G_k$  may fail to be a certificate of  $k$ -connectivity, even in the global sense.

Next, we define  $S$ -greedy forests (which are also maximal) and show that the  $S$ -greedy forests yield certificates of local  $S$ -mixed connectivity.

#### 4.1. Greedy forests

The following nondeterministic search procedure produces a maximal forest  $F$  of  $G$ . Initially  $F = \emptyset$ .

During the procedure every vertex is visited at least once. Edges added to  $F$  are incident to the visited vertex.

The vertices of  $G$  are visited as follows. The first vertex to be visited can be chosen arbitrarily. Whenever a visit of a vertex terminates, the next vertex to be visited can be chosen to be any other vertex of  $V(F)$ , or any vertex of a component of  $G$  which has no vertices in  $V(F)$  yet.

The edges are added to  $F$  (one at a time) as follows. During the first visit of a vertex  $v \in S$ , for every neighbor  $x$  of  $v$ ,  $x \notin V(F)$ , add the edge  $v-x \in E$  to  $F$ . ( $S$ -simplicity implies that there is only one edge joining  $v$  and  $x$ .) When visiting  $v \notin S$ , if  $v \stackrel{e}{-} x \in E$  and if  $x \notin V(F)$ , one is allowed to add  $e$  to  $F$ . (Clearly, if  $e$  is added to  $F$  then its parallel edges, if there are any, may never be added to  $F$ .) So, no edges incident to  $v \in S$  are added after its first visit. If  $v \notin S$ , edges incident to  $v$  may be added during several visits.

Some mechanism for termination, once  $F$  is a maximal forest of  $G$ , is necessary, but not specified here. The forests produced by such a procedure are called  $S$ -greedy.

Next, we define greedy forests without referring to any algorithm (as a static counterpart to the above algorithmic construction).

Let  $F$  be a maximal forest in  $G$ , and let  $t: V \rightarrow \{1, 2, \dots, |V|\}$  be a one-to-one numbering of the vertices. The numbering  $t$  induces orientation on edges:  $\vec{F}(t) \stackrel{\text{def}}{=} \{u \rightarrow v: u-v \in F \wedge t(u) < t(v)\}$ . If  $\vec{T}$  is a tree rooted at  $r$ , directed from the root towards the leaves, then for each  $v \neq r$ ,  $\text{parent}(v)$  is the unique  $u$  such that  $u \rightarrow v \in \vec{T}$ ; also  $\text{parent}(r) \stackrel{\text{def}}{=} r$ .

**Definition 1.** A maximal  $F$  is  $S$ -greedy in  $G$  if there exists a numbering  $t$  of the vertices such that

- (1) For every (maximal) tree  $T$  of  $F$ ,  $\vec{T}(t)$  is a rooted tree.
- (2) If  $w-v \in E$  and  $w \in S$  then  $t(\text{parent}(v)) \leq t(w)$ .

Intuitively, Definition 1 reflects the above algorithmic construction of greedy forests as follows. The order in which the vertices of  $G$  are visited for the first time is

specified by  $t$ . If  $\text{parent}(v) = u \neq v$ , then  $v$  has been added to  $V(F)$  when an edge  $u-v$  has been added to  $F$ , while visiting  $u$ . If  $\text{parent}(v) = v$  then  $v$  is the first vertex in its component to be visited. The second item in the above definition reflects the requirement that on the first visit of  $v \in S$  all its non-forest neighbors join the forest. Obviously, an  $S$ -greedy forest  $F$  in  $G$  is also  $S'$ -greedy for all  $S' \subseteq S$ .

4.2. Certificates

Next, we show that if for each  $i \leq k$  the forest  $F_i$  is  $S$ -greedy in  $(V, \bar{E}_{i-1})$ , then  $G_k$  is a  $k$ -connectivity certificate. (Note:  $|E_k| < k|V|$ .)

Theorem 3 below is a generalization of the main theorem of [6], where it has been stated for a specific subclass of  $S$ -greedy forests.

**Theorem 3** (Mixed Connectivity Certificate).

$$\lambda_S(a, b; G_k) \geq \min\{\lambda_S(a, b; G)\}, \quad \text{for all } a, b \in V.$$

We need a couple of lemmas before proving the theorem. Let  $C = (Z, E(A, B))$  be a cut of  $G$ . We use the following obvious fact:

**Fact 1.** Let  $v \in Z, E(v, B) = \emptyset, C' = (Z - \{v\}, E(A \cup \{v\}, B))$ . Then  $|C'| < |C|$ .

Say, a cut  $C' = (Z', E'(A', B'))$  of  $G' = (V, E' \subseteq E)$  narrows a cut  $C = (Z, E(A, B))$  of  $G$  if  $|C'| < |C|$  and  $C'$  separates  $A$  and  $B$  in  $G'$  ( $A \subseteq A', B \subseteq B'$ , so  $Z' \subseteq Z$ ). For example,  $C'$  narrows  $C$  in Fact 1 above.

**Lemma 4.** Let  $C = (Z, E(A, B))$  be a cut,  $|C| > 0$ , and forest  $F$  be  $S$ -greedy in  $G$ . There is a cut  $C' = (Z', \bar{F}, (A', B'))$  in  $(V, \bar{F})$  which narrows  $C$ .

**Proof.** If  $A$  and  $B$  are not linked in  $G$ , then a zero size cut narrows  $C$ . Thus, assume there is a path in  $G$  which connects some vertex  $a$  of  $A$  with some vertex  $b$  of  $B$ . By the maximality of  $F$ , such a path exists in  $F$  as well. If  $F \cap E(A, B) \neq \emptyset$  then  $C' = (Z, \bar{F}(A, B))$  narrows  $C$ , since  $|\bar{F}(A, B)| < |E(A, B)|$ .

Now, suppose  $F \cap E(A, B) = \emptyset$ . Let  $t$  be a numbering of  $F$  as in Definition 1. Clearly, there is some tree  $T \subseteq F$ , which links  $a$  and  $b$ . Thus,  $V(T) \cap Z \neq \emptyset$ . Let  $r$  be the root of  $\vec{T}(t)$ ; w.l.o.g. assume  $r \notin B$ . Let  $w$  be the least vertex (w.r.t.  $t$ ) in  $V(T) \cap Z$ .

If  $\bar{F}(w, B) = \emptyset$ , then by Fact 1, there is a cut in  $\bar{F}$ , narrowing  $(Z, \bar{F}(A, B))$ . Therefore, this cut narrows  $C$  as well, and the lemma follows.

Suppose  $\bar{F}(w, B) \neq \emptyset$  and let  $w \xrightarrow{e} v \in \bar{F}(w, B)$ . Since there is an edge in  $\bar{F}$  connecting  $w$  and  $v$ , by the maximality of  $F$ ,  $v \in V(T)$  as well. Let  $u = \text{parent}(v)$  in  $\vec{T}(t)$ . By Definition 1 item (2),  $t(u) \leq t(w)$ . Let  $\vec{P}$  be the directed path in  $\vec{T}(t)$  from  $r$  (through  $u$ ) to  $v$ . For every vertex  $x$  on  $\vec{P}$ , from  $r$  to  $u, t(x) \leq t(u)$ .  $\vec{P}$  must have a vertex  $z \in Z$ , since  $r \notin B$  and  $F \cap E(A, B) = \emptyset$ . By  $t(z) \leq t(u)$  it follows that  $t(z) \leq t(w)$ . By the minimality of  $t(w)$ ,  $z = u = w$ . Thus,  $w = \text{parent}(v)$  in  $\vec{T}(t)$ . By the  $S$ -simplicity,  $e \in T$ , in contradiction to  $w \xrightarrow{e} v \in \bar{F}(w, B)$ .  $\square$

**Lemma 5.** *Let  $A$  and  $B$  be linked in  $F_k$  and separated by a cut  $C$  in  $G$ . Then  $|C| \geq k$ .*

**Proof.**  $A$  and  $B$  are linked in  $F_i$  for all  $1 \leq i \leq k$  (by Lemma 3). Use Lemma 4 to construct a sequence of  $k + 1$  cuts  $C_j = (Z_j, \bar{E}_j(A_j, B_j))$ ,  $0 \leq j \leq k$ , with  $C_0 = C$  and  $C_j$  narrowing  $C_{j-1}$ . We have  $|C_0| > |C_1| > \dots > |C_k| \geq 0$ , and thus  $|C| \geq k$ .  $\square$

**Proof of Theorem 3.** By Theorem 1 there is cut  $C = (Z, E_k(A, B))$  separating  $a$  and  $b$  in  $G_k$ , such that  $\lambda_S(a, b; G_k) = |C|$ . If  $E_k(A, B) = E(A, B)$  then  $|C| = |(Z, E(A, B))| \geq \lambda_S(a, b; G)$ . Otherwise, if  $E(A, B) - E_k(A, B) \neq \emptyset$  then by Lemma 3,  $A$  and  $B$  are linked in  $F_k$ , and so by Lemma 5 (applied to  $G_k$ ),  $|C| \geq k$ . In either case,  $\lambda_S(a, b; G_k) = |C| \geq \min\{k, \lambda_S(a, b; G)\}$ .  $\square$

### 4.3. Sequential algorithms

A naive use of the greedy search procedure (described in Section 4.1) to construct  $k$  greedy forests one after the other, takes  $O(k|E|)$  time.

When  $S = V$ , this algorithm is called by Cheriyan et al. *scan-first-search* [2]. They prove that the union of these forests constitutes a certificate of vertex  $k$ -connectivity (of size  $< k|V|$ ). This is a special case of our Theorem 3.

Nagamochi and Ibaraki [14] present an algorithm, which we call *NI-search*, to be described shortly. It applies to general graphs (i.e. parallel edges are allowed) and produces a partition of  $E$  into ( $V$ -greedy) forests  $F_1, F_2, \dots$  in a single search of the graph, thus reducing the complexity to  $O(|V| + |E|)$ . Frank et al. show that if  $G$  is  $S$ -simple, then the resulting  $G_k$  is a certificate of local  $S$ -mixed  $k$ -connectivity [6]. As we shall see, each  $F_i$  produced by the *NI-search* is  $V$ -greedy in  $(V, \bar{E}_{i-1})$ , and therefore these results are subsumed by our Theorem 3.

However, there are certificates composed of  $S$ -greedy forests that cannot be produced by *NI-search* (e.g. see Fig. 2, where  $S = V$ ). Hence, results of [6] do not imply that, in general,  $S$ -greedy forests (and in particular scan-first forests) yield mixed connectivity certificates.

*NI-search* is the only previously published sequential algorithm we know of, to build greedy forests in linear time. Our generic strategy, as presented in Section 4.1, is more general and provides greater flexibility for other implementations. A case in point is the method used in [2] to design efficient parallel and distributed algorithms that produce vertex connectivity certificates.

We describe *NI-search*, and prove that it generates  $V$ -greedy forests. This is done in detail for self-containment and because our distributed algorithm can be viewed as an implementation of *NI-search*.

**NI-Search:** *NI-search* assigns  $\text{rank}(e) > 0$  to each edge  $e$ .  $F_i \stackrel{\text{def}}{=} \{e \in E \mid \text{rank}(e) = i\}$ . Each vertex  $v$  keeps  $\text{label}(v) \stackrel{\text{def}}{=} \max_{e \in E(v)} \{\text{rank}(e)\}$ . Initially,  $\text{rank}(e) = 0$  (we say,  $e$  is *unranked*) for all  $e$ , so  $\text{label}(v) = 0$  for all  $v$ . In each step *NI-search* visits

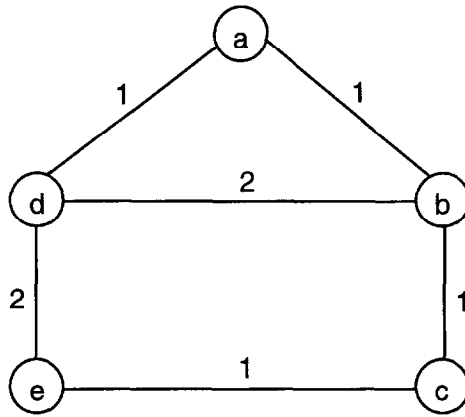


Fig. 2. A certificate made of  $V$ -greedy forests which cannot be obtained by NI-search.

a yet unvisited  $v$  with a highest  $\text{label}(v)$  among the unvisited vertices, and assigns  $\text{rank}(v \xrightarrow{e} w) = \text{label}(w) + 1$  to each unranked  $e \in E(v)$ . NI-search terminates when all vertices have been visited. Notice that if  $\text{label}(v) = i$  then for every  $1 \leq j \leq i$ ,  $v$  has at least one incident edge  $e$  for which  $\text{rank}(e) = j$ .

**Lemma 6.** For each  $i > 0$ ,  $F_i$  produced by NI-search is a  $V$ -greedy forest in  $(V, \overline{E_{i-1}})$ .

**Proof.** First we show that conditions (1) and (2) of Definition 1 hold, and then maximality is proved.

Let numbering  $t$  of the vertices be defined by the order in which they are visited in the NI-search. Let  $T \subseteq F_i$  be a connected component of  $F_i$ . Let  $r \in V(T)$  be the first vertex of  $T$  to be visited:  $t(r) = \min_{v \in V(T)} \{t(v)\}$ . Obviously, the in-degree of  $r$  (in  $\vec{T}$ ) is zero.

For any  $v \in V(T)$ , its in-degree (in  $\vec{T}$ ) is at most one. Indeed, suppose  $u \xrightarrow{e} v \in T$ , and  $t(u) < t(w) < t(v)$ . Then  $w \xrightarrow{e} v \notin T$ : when  $w$  was scanned  $\text{label}(v)$  already was  $\geq i$  (due to  $\text{rank}(e) = i$ ). Therefore,  $\vec{T}(t)$  is a rooted tree and  $F_i$  is a forest. Thus condition (1) holds.

Condition (2) follows directly from the fact that all incident edges of a vertex are ranked during the first visit of the vertex.

Finally, the maximality of  $F_i$  in  $(V, \overline{E_{i-1}})$  is shown by the following sequence of three claims. A tree (in  $F_i$ ) is called *active* if at least one of its vertices has not been visited yet.

**Claim 1.** While NI-search runs, each  $F_i$  contains at most one active tree.

When an edge  $v \xrightarrow{e} w$  is ranked  $i$ , while visiting  $v$ , either  $\text{label}(v) \geq i$  (and the ranking of  $e$  creates no new tree in  $F_i$ ) or  $\text{label}(v) = \text{label}(w) = i - 1$  (thus creating a new tree

in  $F_i$ ). But in the latter case, there is no unvisited  $u \in V$  with  $\text{label}(u) \geq i$  (or  $u$  would be visited rather than  $v$ ), and thus there is no other active tree in  $F_i$ .

**Claim 2.** *If  $u \stackrel{e}{-} v \in F_j$ , then for each  $0 < i < j$ ,  $u$  and  $v$  are connected in  $F_i$ .*

Just before  $e$  is ranked,  $\text{label}(v), \text{label}(u) \geq j - 1$ , so  $v, u \in V(F_i)$  for each  $i < j$ . Also, one of them, say  $u$  has not been visited yet, while the visit at  $v$  is in progress. Thus, just before  $v$  is visited both belong to active trees of  $F_i$ . By Claim 1, both  $u$  and  $v$  are in the unique active tree of  $F_i$ .

**Claim 3.** *Each forest  $F_i$  is maximal in  $(V, \bar{E}_{i-1})$ .*

Let  $u \stackrel{e}{-} v \in \bar{E}_i$ . Thus,  $e \in F_j$  for some  $j > i$ . Therefore, by Claim 2,  $u, v$  are connected in  $F_i$ .  $\square$

Lemma 6 and Theorem 3 yield the following:

**Corollary 1** (Frank, Ibaraki and Nagamochi [6]). *If  $G$  is  $S$ -simple then,  $G_k$  (produced by  $NI$ -search) is a certificate of local  $S$ -mixed  $k$ -connectivity for  $G$  and its size is  $< k|V|$ .*

#### 4.4. Distributed algorithm

In this section we present a new distributed algorithm for finding mixed connectivity certificates of size  $< k|V|$ , for connected  $S$ -simple graphs. The algorithm, described in Fig. 3, is executed in a network identified with a graph  $G$ : each vertex of  $V$  corresponds to a node of the network, and each edge of  $E$  corresponds to a communication link. Each vertex  $v$  maintains variables corresponding to the ones in  $NI$ -search of Section 4.3:  $\text{label}$  and  $\text{rank}(e)$ , for each incident edge  $e$ , all initially 0. In addition, each vertex  $v$  has a boolean variable  $\text{first\_time}$  initially set to *true*, and a list *unvisited* initially including all edges incident to  $v$ . The algorithm is initiated by sending a VISIT message to any one vertex, on a special *nil* edge, and terminates with the RETURN message received back on the same *nil* edge. A vertex  $v$  completes its work after it sends a RETURN message on an edge of  $\text{rank} = 1$ , or on *nil*, if it is the initiator. When every vertex completes its work  $\text{rank}(e) > 0$  for all  $e$ ,  $|E_k| < k|V|$  for any  $k > 0$ , and if  $G$  is  $S$ -simple then  $G_k$  is a certificate of local  $S$ -mixed  $k$ -connectivity.

Note that  $\text{EDGE\_RANKED}(i)$  means a message of type  $\text{EDGE\_RANKED}$  in which the value  $i$  is specified.

The algorithm is of a restricted form. A single center of activity – we call it the *server* – travels from vertex to vertex around the network. When the server is in a vertex  $v$ , messages are sent only between  $v$  and its neighbors. The messages used by the algorithm are: VISIT, RETURN, RANK\_EDGE, and  $\text{EDGE\_RANKED}(i)$ ,  $1 \leq i \leq |E|$ . The server travels with the VISIT and RETURN messages. All the unranked edges incident to a vertex are ranked when the server arrives at it for the first time, and in

**ALGORITHM D** (at vertex  $v$ )

- 
- ```

(1) for RANK_EDGE message arriving at  $v$  on edge  $e$  do
    (1.1) label,  $\text{rank}(e) \leftarrow \text{label} + 1$ 
    (1.2) send EDGE_RANKED( $\text{rank}(e)$ ) on edge  $e$ 
(2) for VISIT message arriving at  $v$  on edge  $e$  do
    (2.1) drop  $e$  from unvisited
    (2.2) If first_time then
        (2.2.1) first_time  $\leftarrow$  false
        (2.2.2) for all edges  $e'$ ,  $e' \neq \text{nil}$ , s.t.  $\text{rank}(e') = 0$  do
            (2.2.2.1) send RANK_EDGE on edge  $e'$ 
            (2.2.2.2) wait for EDGE_RANKED( $i$ ) to arrive on edge  $e'$ 
            (2.2.2.3)  $\text{rank}(e') \leftarrow i$ 
            (2.2.2.4) If  $\text{label} < i$  then  $\text{label} \leftarrow i$  /*OPTIONAL*/
        (2.3) for each  $e' \in \text{unvisited}$  with  $\text{rank}(e') \geq \text{rank}(e)$  in decreasing order of
            rank}(e') do
                (2.3.1) drop  $e'$  from unvisited
                (2.3.2) send VISIT message on  $e'$ 
                (2.3.3) wait for RETURN message on  $e'$ 
    (2.4) send RETURN message on  $e$ 

```
- 

Fig. 3. Single server algorithm  $D$  for  $S$ -simple graphs.

the same way as in the  $NI$ -search. We say that a vertex is *visited* if its *first\_time* is *false*. Thus, a visited vertex has received at least one VISIT, and once it gets its first VISIT, the server moves from it only after the vertex has no unranked incident edges.

**Lemma 7.** *Algorithm D terminates.*

**Proof.** Each edge can be ranked at most once, in Step (1.1) or (2.2.2.3). When VISIT arrives (Step (2.1)) or when sent (Step (2.3.1)) through an edge, the edge is dropped from *unvisited*. Thus, VISIT is never sent again on the same edge. Therefore, RETURN can be sent on each edge at most once. Finally, the server cannot get stuck forever in any vertex. Since the graph is finite, the algorithm terminates.  $\square$

As before, let us denote the set of edges which get  $\text{rank} = j$ , by  $F_j$ .

**Lemma 8.**  $F_j$  is circuit-free.

**Proof.** Assume there is a simple circuit in  $G$  such that all its edges have  $\text{rank} = j$ . Let  $a \xrightarrow{e} b$  be the last edge in this circuit to be ranked. The server is in  $a$  or in  $b$  when this happens. Just before the ranking of  $e$ , since both  $a$  and  $b$  already have an edge ranked  $j$ , for each of them  $\text{label} \geq j$ . By (1.1),  $e$  must get a rank higher than  $j$ . A contradiction.  $\square$

Next, we want to establish the fact that once a vertex  $v$  sends VISIT on some edge  $e'$ , the next time the server is in  $v$  again (if at all) is by a RETURN on  $e'$ , in the opposite direction, as in Step (2.3.3).

For this purpose, let us direct the edges in the direction in which VISIT has traversed them (if at all). As shown in the proof of Lemma 7, VISIT can pass at most once through an edge. Thus, there is no ambiguity in this direction assignment. Let us call such a directed edge *open* if RETURN has not transversed it yet. Denote by  $\vec{G}$  the directed graph which contains all open edges.

**Lemma 9.**  $\vec{G}$  is a simple directed path.

**Proof.** First, let us show that  $\vec{G}$  is acyclic. As Algorithm D runs,  $\vec{G}$  changes. Let  $C$  be the first (simple) cycle to occur, where  $C$  is:

$$v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} v_2 \cdots v_{l-1} \xrightarrow{e_l} v_0.$$

By (2.3),  $\text{rank}(e_1) \leq \text{rank}(e_2) \leq \cdots \leq \text{rank}(e_l)$ . Since they cannot be all equal (see Lemma 8),  $\text{rank}(e_1) < \text{rank}(e_l)$ . Also, when  $e_1$  is chosen (at vertex  $v_0$ , per (2.3)),  $e_l$  is already ranked, and should have been chosen instead.

Now consider the position of the server. Since it moves only with VISIT and RETURN messages, the underlying undirected graph of  $\vec{G}$  is connected. Algorithm D never sends a second VISIT from the same vertex, before RETURN arrives of the previous edge. Thus, the out-degree of every vertex of  $\vec{G}$  is at most one. Since there are no cycles, the in-degree of every vertex is at most one. It follows that  $\vec{G}$  is a simple directed path.  $\square$

Let  $T$  be the component of  $F_1$  which includes vertex  $r$ , the vertex with the *nil* edge. By Lemma 8,  $T$  is a tree. We want to show that it spans  $V$ .

**Lemma 10.** Upon termination of Algorithm D, every vertex of  $T$  is visited.

**Proof.** Since RETURN is sent from  $r$  on *nil*, every neighbor of  $r$  in  $T$  has received a VISIT message and by Lemma 9, has sent back RETURN. By induction on the distance from  $r$ , every vertex in  $T$  has received VISIT and has sent back RETURN.  $\square$

**Lemma 11.**  $T$  spans  $V$ .

**Proof.** Let  $S$  be the set of vertices spanned by  $T$ . If  $S \neq V$ , consider the set of edges,  $(S, \bar{S})$  with one end-vertex in  $S$  and another in  $\bar{S}$ . The set is not empty since  $G$  is connected. The server starts in  $r \in S$  and the only way it can cross to  $\bar{S}$  is when it travels on one of the edges of  $(S, \bar{S})$ . Thus, when the server visits for the first time a vertex with an edge  $e \in (S, \bar{S})$ , none of the vertices in  $\bar{S}$  has been labeled, and none of the edges of  $(S, \bar{S})$  has been ranked. Thus, by (2.2.2) and (1),  $\text{rank}(e) = 1$ , contradicting the definition of  $T$ .  $\square$

Lemmas 10 and 11 imply that upon termination, every vertex of  $G$  has been visited. It is also clear that the ranking of unranked edges is done according to the  $NI$ -search. It remains to be shown that when a vertex is visited for the first time, it has the highest label of all unvisited vertices.

**Lemma 12.** *Assume at some point during the execution of Algorithm D,  $v$  is about to send VISIT on an edge ranked  $i$ . There is no open edge  $e$  for which  $\text{rank}(e) > i$ .*

**Proof.** By Lemma 9,  $v$  must be the last vertex on the directed path which constitutes the current  $\vec{G}$ . If  $e$  is open then it is on the directed path. Thus,  $\text{rank}(e) \leq i$ .  $\square$

**Lemma 13.** *Assume at some point during the execution of the algorithm,  $v$  has sent RANK\_EDGE along  $v \xrightarrow{e'} w$ , but  $v$  has not sent VISIT along  $e'$ . Then there exists an open edge  $e''$ ,  $\text{rank}(e'') \geq \text{rank}(e')$ .*

**Proof.** Consider the moment when  $v$  sent RANK\_EDGE along  $e'$ , and let  $\text{rank}(e') = \rho$ . This happens in (2.2), after  $v$  has received VISIT for the first time, say via  $e$ , and while it executes step (2.2.2). There are two cases.

(1) If  $\rho \geq \text{rank}(e)$ , line (2.3) implies that  $v$  has sent a VISIT along some edge  $e''$ ,  $\text{rank}(e'') \geq \text{rank}(e')$ , but has not received a RETURN on it yet.

(2) Else,  $\rho < \text{rank}(e)$ . When  $v$  received VISIT,  $\text{label} \geq \text{rank}(e)$ . Thus, it already had an incident edge  $v \xleftarrow{e_1} v_1$  with  $\text{rank}(e_1) = \rho$ . Since it received VISIT for the first time from  $e$ , no VISIT has been sent along  $e_1$ . Yet,  $v_1$  has sent RANK\_EDGE along  $e_1$ .

Repeating this argument we find a path:

$$v = v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \dots$$

such that  $v_j$  has sent RANK\_EDGE along  $e_j$  but has not sent VISIT through it, and the rank of all edges in the path is  $\rho$ . Since the graph is finite and since  $F_\rho$  is circuit-free (see Lemma 8), this path must end in a vertex satisfying case (1).  $\square$

**Theorem 4.** *Algorithm D implements NI-search, runs in time  $4|E|$  and sends  $4|E|$  messages.*

**Proof.** It was already shown that all vertices of  $G$  are visited, and once a vertex is visited for the first time, its unranked edges are ranked as in the  $NI$ -search. To show that Algorithm D implements  $NI$ -search it remains to prove that when a vertex is visited for the first time, its label is highest among unvisited vertices. This is proved by contradiction.

Assume  $v$  is about to send a VISIT to an unvisited processor of label =  $i$ , but there is some other unvisited processor  $w$  with label =  $i' > i$ . Then there is an edge  $w' \xrightarrow{e'} w$  with  $\text{rank}(e') = i'$ , and  $w'$  has been visited.

The node  $w'$  and edge  $w' \xrightarrow{e'} w$  satisfy the hypothesis of Lemma 13. Thus, there exists an open edge  $e''$  for which  $\text{rank}(e'') \geq \text{rank}(e')$ . In contradiction to Lemma 12.

For the complexity, observe that each (non-*nil*) edge sees the following sequence of messages: RANK\_EDGE, EDGE\_RANKED( $\cdot$ ), VISIT, RETURN. All of these are constant size except the EDGE\_RANKED( $i$ ), which has  $\lceil \log i \rceil$  bits, where  $i < |E|$ . Thus the total number of messages sent by the algorithm is at most  $4|E|$ . Since in each step a message is sent by the algorithm the time complexity is at most  $4|E|$ .  $\square$

**Remarks.** Line (2.2.2.4) can be omitted without affecting the algorithm correctness, and is included only to guarantee that, analogously to *NI*-search, if the server is at  $v$  then for any unvisited  $u$ ,  $\text{label}(v) \geq \text{label}(u)$ .

Also, note that the search for an unvisited vertex of maximum label in the whole graph, as required in the original *NI*-search, is avoided.

Observe that some time can be saved by parallelizing step (2.2.2). If  $k$ -connectivity is desired only for a fixed  $k$ , then the algorithm can be modified to run in time complexity  $O(k|V|)$  (and each message size is  $\leq \lceil \log k \rceil$ ).

## References

- [1] B. Awerbuch, D. Peleg, Network synchronization with polylogarithmic overhead, FOCS (1990) 514–522.
- [2] J. Cheriyan, M.-Y. Kao, R. Thurimella, Scan-first search and sparse certificates: an improved parallel algorithm for  $k$ -vertex connectivity, SIAM J. Comput. 22 (1) (1993) 157–174.
- [3] J. Cheriyan, S.N. Maheshwari, Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs, J. Algorithms 9 (1988) 507–537.
- [4] G.B. Dantzig, D.R. Fulkerson, On the Max-Flow Min-Cut Theorem of networks, Linear Inequalities and Related Systems, Annals of Math. Study, vol. 38, Princeton University Press, Princeton, NJ, 1956, pp. 215–221.
- [5] D. Eppstein, Z. Galil, G.F. Italiano, A. Nissenzweig, Sparsification – a technique for speeding up dynamic algorithms, FOCS (1992) 60–69.
- [6] A. Frank, T. Ibaraki, H. Nagamochi, On sparse subgraphs preserving connectivity properties, J. Graph Theory 17 (3) (1993) 275–281.
- [7] H. Gabow, A matroid approach to finding edge connectivity and packing arborescences, STOC (1991) 112–132.
- [8] M.R. Garey, D.S. Johnson, Computers and Intractability, a guide to the theory of NP-completeness, Freeman, San Francisco, 1979, p. 198.
- [9] A. Itai, M. Rodeh, The multi-tree approach to reliability in distributed networks, Inform. and Comput. 79 (1) (1988) 3–59.
- [10] G. Itkis, Connectivity certificates made simple and general, manuscript, 1996.
- [11] W. Mader, Über minimal  $n$ -fach zusammenhängende, unendliche Graphen und ein Extremalproblem, Arch. Mat. 23 (1972) 553–560.
- [12] W. Mader, Grad und lokaler Zusammenhang in endlichen Graphen, Math. Ann. 205 (1973) 9–11.
- [13] W. Mader, Connectivity and edge-connectivity in finite graphs, in: B. Bollobas (Ed.), Surveys on Combinatorics, London Math. Soc. Lecture Note Series, vol. 38, London Math. Soc., London, 1979, pp. 293–309.
- [14] H. Nagamochi, T. Ibaraki, A linear-time algorithm for finding a sparse  $k$ -connected spanning subgraph of a  $k$ -connected graph, Algorithmica 7 (1992) 583–596.
- [15] R. Thurimella, Sub-linear distributed algorithms for sparse certificates and biconnected components, Proc. 14th ACM Symp. on Principles of Distributed Computing (PODC'95), pp. 28–37.
- [16] A.A. Zykov, Theory of Finite Graphs, Nauka, Novosibirsk, 1969, see pp. 104–105 (in Russian).