Journal of the Operations Research
Society of Japan
2004, Vol. 47, No. 4, 199-223

GRAPH ALGORITHMS FOR NETWORK CONNECTIVITY PROBLEMS

Hiroshi Nagamochi
Kyoto University

(Received September 30, 2003; Revised February 4, 2004)

Abstract This paper surveys the recent progress on the graph algorithms for solving network connectivity problems such as the extreme set problem, the cactus representation problem, the edge-connectivity augmentation problem and the source location problem. In particular, we show that efficient algorithms for these problems can be designed based on maximum adjacency orderings.

Keywords: Graph theory, network flow, algorithm, MA ordering, minimum cut, edge-connectivity, graph augmentation, source location problem

1. Introduction

Connectivity of networks is one of the most fundamental and useful notions for analyzing various types of network problems in the practical applications such as communication networks and VLSI layouts. Many graph algorithms have been developed for solving the network connectivity problems. Needless to say, the minimum cut maximum flow theorem discovered by P. Elias, A. Feinstein and C. F. Shannon [6] and L. R. Ford and D. R. Fulkerson [7] is the basis of most of those algorithms. In particular, a maximum flow algorithm that finds a maximum flow and a minimum cut between two specified vertices has been used as a building block of many algorithms for solving connectivity problems. In the last two decades, development of fast maximum flow algorithms has been an important issue, and the time to solve the maximum flow problem in a network with n vertices and m edges has been reduced to nearly $O(nm)$ (see [1]). Contrary to this, H. Nagamochi and T. Ibaraki [30] devised a new algorithm that finds a global minimum cut without constructing any maximum flow. The algorithm consists of a graph traversal procedure for computing a vertex ordering called a maximum adjacency ordering (MA ordering for short), and can be implemented to run in $O(mn+n^2\log n)$ time, which is the currently best time bound for computing a minimum cut deterministically. Afterwards, many efficient algorithms for solving graph connectivity problems have been obtained by making use of MA orderings. In particular, $O(mn+n^2\log n)$ time algorithms are obtained to the problem such as the extreme set problem, the cactus representation problem, the edge-connectivity augmentation problem, the edge-splitting problem and the source location problem with the edge-connectivity requirement. In this survey, we show how such efficient algorithms can be designed based on MA orderings.

The paper is organized as follows. After introducing basic definitions on connectivities and flows in sections 2 and 3, we show useful properties of MA orderings in section 4. In section 5, we review four basic structural representations of a given network, a Gomory-Hu tree, maximal components, extreme sets and a cactus representation. We then show $O(mn+n^2\log n)$ time algorithms that compute extreme sets and a cactus representation in sections 6 and 7, respectively. Based on these algorithms, we in section 8 prove that the edge-connectivity augmentation problem and the edge-splitting problem can be solved in $O(mn + n^2 \log n)$ time. In section 9, reviewing the recent progress on the source location problem, we show efficient algorithms for solving the problem with the edge- or vertex-connectivity requirement.

2. Preliminaries

Let $\Re$ (resp., $\Re_+$) denote the set of reals (resp., nonnegative reals), and $\mathbf{Z}$ (resp., $\mathbf{Z}_+$) denote the set of integers (resp., nonnegative integers). A singleton set $\{x\}$ may be simply written as x, and " $\subset$ " implies proper inclusion while " $\subseteq$ " means " $\subset$ " or " $=$ ".

Let V be a finite set. For two subsets $A, B \subset V$, we say that a subset $X \subseteq V$ *separates* A and B if $A \subseteq X \subseteq V - B$ or $B \subseteq X \subseteq V - A$ holds. For two subsets $X, Y \subseteq V$, we say that X and Y *intersect* each other if $X \cap Y \neq \emptyset$, $X - Y \neq \emptyset$ and $Y - X \neq \emptyset$ hold, and that X and Y *cross* each other if, in addition, $V - (X \cup Y) \neq \emptyset$ holds. A family $\mathcal{X} \subset 2^V$ is called *laminar* if no two subsets in $\mathcal{X}$ intersect each other. A laminar family $\mathcal{X} \subset 2^V$ is represented by a rooted tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ as follows, where we use term "nodes" for tree representations.

(i) The node set $\mathcal{V}$ of $\mathcal{T}$ consists of nodes each of which corresponds to the set V or a subset $X \in \mathcal{X}$, i.e., $\mathcal{V} = \mathcal{X} \cup \{V\}$, where the root corresponds to V. A node corresponding to a subset $X \subseteq V$ is denoted by u_X.

(ii) For two nodes u_X and u_Y, u_X is a child of u_Y in $\mathcal{T}$ if and only if $X \subset Y$ holds and $\mathcal{X}$ contains no set X' with $X \subset X' \subset Y$. The set of children of a node u is denoted by $Ch(u)$.

Let $G = (V, E)$ be an undirected graph with a vertex set V and an edge set E, where each edge e may be weighted by a nonnegative real $c_G(e) \in \Re_+$. The weight $c_G(e)$ of an edge $e = (u, v)$ may be written as $c_G(u, v)$. A graph G is called *unweighted* if $c_G(e) = 1$ for all edges in E. The vertex set and edge set of a graph G may be denoted by $V(G)$ and $E(G)$, respectively. Edges with the same end vertices are called *multiple edges*. A graph is called *multiple* if it is allowed to have multiple edges; it is called *simple* otherwise. We say that two vertices u and v are *connected* by a path consisting of edges with positive weights. Let $E^+(G)$ denote the set of unordered pairs of vertices $u, v \in V$ such that E contains an edge $e = (u, v)$ with $c_G(e) > 0$. We denote $n = |V|$, $e = |E|$ and $m = |E^+(G)|$. The input size of a multiple graph $G = (V, E)$ is measured by n and e. However, a multigraph $G = (V, E)$ can be represented by an edge-weighted graph in which each $c_G(u, v)$ is defined by the number of multiple edges between u and v. In this case, the input size is $O(n + m)$. Figure 1 shows an integer weighted graph G, where the number of lines between two vertices represents the weight of the edge between them. Note that the graph G can be viewed as a multigraph with multiplicity equal to the edge weight.

For a vertex $v \in V$, let $\Gamma_G(v)$ denote the set of *neighbours* of u (i.e., vertices adjacent to v). For a subset $X \subseteq V$, let $\Gamma_G(X) = \cup_{v \in X} \Gamma_G(v) - X$. For two disjoint subsets $X, Y \subset V$, $E_G(X, Y)$ denotes the set of edges joining a vertex in X and a vertex in Y, and $d_G(X, Y)$ denotes $\sum_{e \in E_G(X,Y)} c_G(e)$. In particular, they may be written as $E_G(X)$ and $d_G(X)$, respectively, if $Y = V - X$. Also $E_G(X, Y)$ and $d_G(X, Y)$ may be written as $E(X, Y)$ and $d(X, Y)$, respectively, if G is clear from context.

A partition $\{X, V - X\}$ of V such that X is a nonempty and proper subset of V is called a *cut* of G. A cut $\{X, V - X\}$ may be denoted by X for convenience. A subset $E' \subseteq E$ such that $E' \supseteq E(X, V - X)$ for some cut X is called a *cut set*. For a cut set E' such that $E' = E(X, V - X)$, we say that cut $\{X, V - X\}$ is *generated* by E'. The cut size of a cut set

E' (resp., a cut X) is defined by $\sum_{e \in E'} c_G(e)$ (resp., $d_G(X)$). We say that cuts X and Y are *intersecting* (resp., *crossing*) if the subsets X and Y are intersecting (resp., crossing). A cut X separating two subsets $A, B \subset V$ is called an (A, B)*-cut*.

For a subset $F \subseteq E$, $G - F$ denotes the graph obtained from G by removing edges in F. For a subset $X \subseteq V$, $G - X$ denotes the graph obtained from G by removing vertices in X together with edges incident to a vertex in X, and G/X denotes the graph obtained from G by contracting vertices in X into a single vertex (deleting any resulting loops and merging any resulting parallel edges into a single edge with the sum of their weights). For a given graph $G = (V, E)$, *a star augmentation* is a graph obtained by adding a new vertex s to G together with new weighted edges between s and some vertices in V. The resulting graph H is denoted by $H = G + b$, where $b : V \to \Re_+$ is a weight function such that, for each $v \in V$, $b(v) = c_H(s, v)$, i.e., the weight of new edge (s, v), where we let $b(v) = 0$ if edge (s, v) is not introduced in the star augmentation.

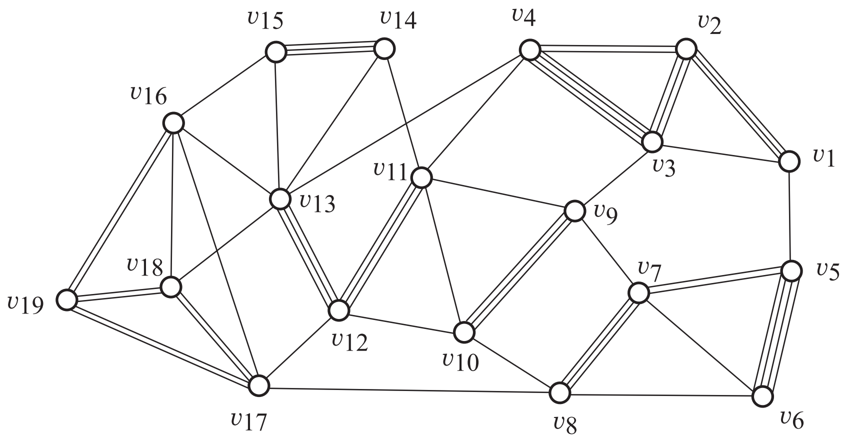


Figure 1: An integer-weighted graph G

Edge-connectivity For two vertices u and v, a (u, v)-cut X with the minimum cut size $d_G(X)$ over all (u, v)-cuts is called a *minimum* (u, v)-cut, and the cut size $d_G(X)$ is called *the local edge-connectivity* $\lambda_G(u, v)$ between u and v. The minimum cut size among all cuts in G is called the *edge-connectivity* of G, and is denoted by $\lambda(G)$; The edge connectivity of a graph consists of a single vertex is set to be $+\infty$. Note that $\lambda(G) = \min_{u,v \in V} \lambda_G(u, v)$. A cut X with $d_G(X) = \lambda(G)$ is called a *minimum cut* in G. For a real $k \in \Re_+$, a graph G is called k*-edge-connected* if $\lambda(G) \geq k$. Given a subset $S \subseteq V$ and a vertex $v \in V - S$, we define the edge-connectivity between them as follows. Let $\lambda_G(S, v)$ denote the minimum size of an edge cut $C \subseteq E$ that separates v from S (i.e., v and S belong to different components in $G - C$). Notice that $\lambda_G(S, v)$ is equal to $\lambda_{G/S}(s, v)$ in the graph G/S obtained by contracting S into a single vertex s.

Vertex-connectivity For a connected graph $G = (V, E)$, a subset $Z \subset V$ is called a *vertex-cut* if $G - Z$ has at least two connected components. The size of a vertex-cut Z is defined by $|Z|$. The maximum number of internally vertex-disjoint paths from u to v is called *the local vertex-connectivity* between u and v, and is denoted by $\kappa_G(u, v)$. If u and v are not adjacent, then $\kappa_G(u, v)$ is equal to the minimum size of vertex-cut Z separating u and v (i.e., u and v belong to different components in $G - Z$). The minimum size of vertex-cuts in G is called the *vertex-connectivity*, denoted by $\kappa(G)$. Thus, $\kappa(G) = \min\{\kappa_G(u, v) \mid u, v \in V\}$. A graph G is called *k-vertex-connected* if $|V| \geq k+1$ and $\kappa(G) \geq k$ (i.e., there is no vertex-cut S with size at most $k-1$). Given a subset $S \subseteq V$ and a vertex $v \in V-S$, we define the vertex-connectivity between them in the following two ways. Let $\kappa_G(S,v)$ denote the minimum size of a vertex cut $Z \subseteq V-S-v$ that separates S and v, and $\hat{\kappa}_G(S,v)$ denote the maximum number of vertex-disjoint paths between S and v such that no two paths meet at the same vertex in S. Hence $\hat{\kappa}_G(S,v) \geq k$ means that v remains connected to at least one vertex in S after deleting any $k-1$ vertices in $V-v$. Also observe that $\hat{\kappa}_G(S,v) \leq \kappa_G(S,v)$ and $\hat{\kappa}_G(S,v) \leq |S|$.

3. Maximum Flows and (s,t)-Minimum Cuts

To define flows, let $G=(V,E)$ stand for a digraph with a set V of vertices and a set E of edges, where each edge $e \in E$ is weighted by a nonnegative real $c_G(e)$. For two disjoint subsets $X, Y \subseteq V$, $E(X,Y)$ in a digraph G denotes the set of edges e such that the tail and head of e are contained in X and Y, respectively. Let $s,t \in V$ be two designated vertices, which we call the source and sink of G, respectively. A subset $X \subset V$ such that $s \in X$ and $t \in V-X$ is called an (s,t)-cut, and its weight, denoted by $d_G^+(X)$, is defined by $\sum_{e\in E(X,V-X)} c_G(e)$.

A function $f : E \to \Re_+$ is called a *flow* (or (s,t)-flow) of G if it satisfies the following two types of constraints:
Flow Conservation Law:

$$\sum_{e\in E(v,V-v)} f(e) - \sum_{e\in E(V-v,v)} f(e) \begin{cases} =0 & \text{if } v \in V-\{s,t\}, \\ \geq 0 & \text{if } v=s, \\ \leq 0 & \text{if } v=t. \end{cases} \tag{3.1}$$

Capacity Constraint:

$$f(e) \leq c_G(e) \text{ for all edges } e \in E.$$

The *flow value* $v(f)$ of f is defined by

$$\sum_{e\in E(s,V-s)} f(e) - \sum_{e\in E(V-s,s)} f(e) \quad \Big(= - \sum_{e\in E(t,V-t)} f(e) + \sum_{e\in E(V-t,t)} f(e)\Big).$$

A flow f that maximizes $v(f)$ is called a *maximum flow* of G. It is a simple matter to see that the flow value $v(f)$ of an (s,t)-flow f cannot exceed the weight $d_G^+(X)$ of any (s,t)-cut X. The next theorem provides many efficient algorithms for solving connectivity problems.

Theorem 3.1 [6, 7] *For an edge-weighted graph G with a source s and a sink t,*
$\max\{v(f) \mid (s,t)\text{-flows } f\} = \min\{d_G^+(X) \mid (s,t)\text{-cuts } X\}$. ∎

A maximum (s,t)-flow in a digraph with n vertices and m edges can be computed efficiently. For example, A. Goldberg and R. E. Tarjan [15] have given an $O(nm\log(n^2/m))$ time algorithm. It is not difficult to see that a minimum (s,t)-cut X can be identified from any maximum (s,t)-flow f. Moreover it is known that all minimum (s,t)-cuts can be represented by a directed acyclic graph (DAG) with $O(n+m)$ size [42].

A minimum (u,v)-cut for two specified vertices u and v in an edge-weighted undirected graph G can be obtained $O(nm\log(n^2/m))$ time by applying the maximum flow algorithm to the digraph G' obtained from G by replacing each edge with two oppositely oriented edges with the same edge weight.

In an unweighted undirected graph G, Theorem 3.1 implies the well-known theorem of K. Menger, i.e., $\lambda_G(u,v)$ is equal to the maximum number of edge disjoint paths between u and v in G, and $\kappa_G(u,v)$ is equal to the maximum number of internally vertex disjoint paths between u and v in G.

4. Maximum Adjacency (MA) Ordering

For any pair of vertices $s, t \in V$ in a graph G, we can compute the local edge-connectivity $\lambda_G(s,t)$ by using the conventional maximum flow algorithm (e.g., [1, 43]). However, the local edge-connectivity $\lambda_G(u,v)$ for some pair $u, v \in V$ (which is specified by the algorithm) can be computed by a significantly simpler method. An ordering $\sigma = (v_1, v_2, \ldots, v_n)$ of vertices in G is called *a maximum adjacency ordering* (MA ordering, for short) if it satisfies

$$d_G(\{v_1, v_2, \ldots, v_i\}, v_{i+1}) \geq d_G(\{v_1, v_2, \ldots, v_i\}, v_j), \quad 1 \leq i < j \leq n.$$

For example, the vertices $v_1, v_2, \ldots, v_{19}$ in the graph G in Figure 1 are numbered as an MA ordering starting from v_1.

4.1. MA ordering algorithm

An MA ordering can be found by starting with an arbitrary vertex v_1 and by choosing the $(i+1)$-th vertex v_{i+1} as a vertex $u \in V - \{v_1, \ldots, v_i\}$ that has the largest sum of the weights of edges between u and the first i chosen vertices $\{v_1, \ldots, v_i\}$. By using the data structure of Fibonacci heap [11], an MA ordering starting from an arbitrarily chosen vertex v_1 can be obtained in $O(n+e)$ or in $O(m + n \log n)$ time [30]. The following property of an MA ordering is the starting point of the rest of development.

Theorem 4.1 *For a graph $G = (V,E)$, let v_{n-1} and v_n be the last two vertices in an MA ordering. Then:*
(i) [9, 12, 30, 34, 44] $\lambda_G(v_{n-1}, v_n) = d_G(v_n)$.
(ii) [10, 30] $\kappa_G(v_{n-1}, v_n) = d_G(v_n)$ *if G is simple and unweighted.* ∎

For example, $\lambda_G(v_{18}, v_{19}) = d_G(v_{18}) = 6$ for the last two vertices in an MA ordering $\sigma = (v_1, v_2, \ldots, v_{19})$ of the graph G in Figure 1. Theorem 4.1(i) tells that we can identify the local edge-connectivity between some two vertices u and v in nearly linear time. Unlike a maximum flow algorithm in the previous section, we cannot specify the pair of vertices u and v, which is part of the output. However, we can choose a vertex s so that it is not output in a pair of vertices (by starting an MA ordering from s). Also, an output pair u and v has a special type of a minimum (u,v)-cut that separates a single vertex v from the rest of vertices. These properties have been used effectively to design faster algorithms for several problems than those designed based on a maximum flow algorithm.

4.2. Constructing flows

We show a hierarchical structure of MA orderings, based on which a maximum flow with flow value $d_G(v_n)$ between the last two vertices v_{n-1} and v_n can be computed efficiently.

For a given MA ordering $\sigma = (v_1, v_2, \ldots, v_n)$ and a real $\delta \in [0, d_G(v_n)]$, we show a way of splitting the graph $G = (V,E)$ into two edge-weighted graphs $A = (V,E)$ and $B = (V,E)$ such that $c_G(v,w) = c_A(v,w) + c_B(v,w)$ for all pairs $v, w \in V$. The edge weights of A and B are determined as follows. For each $i = 2, 3, \ldots, n$, let h_i be the maximum index with

$$h_i < i,\ (v_{h_i}, v_i) \in E \text{ and } d_G(\{v_1, v_2, \ldots, v_{h_i}\}, v_i) \leq \delta,$$

where we let $h_i = 0$ if $E_G(\{v_1, v_2, \ldots, v_{i-1}\}, v_i) = \emptyset$. The edge weight c_A of A is defined by

$$c_A(v_h, v_i) = \begin{cases} c_G(v_h, v_i), & \text{if } i < h \leq h_i \\ \delta - d_G(\{v_1, v_2, \ldots, v_{h_i}\}, v_i), & \text{if } h = h_i + 1 \\ 0, & \text{if } h > h_i + 1. \end{cases}$$

Then the edge weight c_B of B is given by $c_B(e) = c_G(e) - c_A(e)$ for all edges $e \in E$. We call the resulting graphs A and B the *δ-skeleton* and the *δ-skin* of G (with respect to σ), respectively.

Lemma 4.1 [34] *Let $\sigma = (v_1, v_2, \ldots, v_n)$ be an MA ordering of a graph G. Then for any real $\delta \in [0, d_G(v_n)]$, the δ-skin B of G satisfies $\lambda_B(v_{n-1}, v_n) = d_G(v_n) - \delta$ $(= d_B(v_n))$.* ∎

The lemma will be the basis of an $O(nm+n^2 \log n)$ time algorithm for computing extreme sets of a graph in section 6. Based on Lemma 4.1, one can also construct a maximum flow between the last two vertices v_{n-1} and v_n of an MA ordering σ by repeatedly eliminating an acyclic δ-skin B for some $\delta > 0$ from the graph (note that B contains a unique path between v_{n-1} and v_n, which will be part of a maximum (v_{n-1}, v_n)-flow). By using a dynamic tree, this can be implemented to run in $O(m \log n)$ time [34]. Designing a slightly different algorihtm from this, S. R. Arikati and K. Mehlhorn have shown the next result.

Lemma 4.2 [3] *A maximum flow between the last two vertices of an MA ordering can be computed in $O(m)$ time and space.* ∎

The algorithm will be used to design an algorithm for constructing a cactus representaion in section 7.

4.3. Sparsification

An MA ordering is also used to find a sparse spanning subgraph of a given graph G while preserving the vertex- and edge-connectivities of G [10, 29].

Let $\sigma = (v_1, v_2, \ldots, v_n)$ be an MA ordering of a multigraph G. For each $i = 2, \ldots, n$, we denote the edges between $\{v_1, \ldots, v_{i-1}\}$ and v_i (i.e., those in $E_G(\{v_1, \ldots, v_{i-1}\}, v_i)$) by $e_{i,1} = (v_{j_1}, v_i), e_{i,2} = (v_{j_2}, v_i), \ldots, e_{i,p} = (v_{j_p}, v_i)$ so that $1 \le j_1 \le j_2 \le \cdots \le j_p$ holds. By defining

$$F_k = \{e_{2,k}, e_{3,k}, \ldots, e_{n,k}\},\ k = 1, 2, \ldots, |E| \qquad (4.1)$$

(some of $e_{i,k}$ may be void), we have a partition $(F_1, \ldots, F_{|E|})$ of E. Then it is not difficult to see from the definition of an MA ordering that (V, F_i) is a maximal spanning forest in $G - (F_1 \cup F_2 \cup \cdots \cup F_{i-1})$. For example, Figure 2 shows such spanning forests $F_1, \ldots, F_6$ for the MA ordering $\sigma = (v_1, v_2, \ldots, v_{19})$ of the graph G in Figure 1, where we regard G as a multigraph such that the number of lines between two vertices represents the number of edge between them.

Theorem 4.2 *For an unweighted multigraph $G = (V, E)$, let $F_1, F_2, \ldots, F_{|E|}$ be the partition of E obtained from an MA ordering by (4.1), where $F_i = F_{i+1} = \cdots = F_{|E|} = \emptyset$ possibly holds for some i. Let $G_k = (V, F_1 \cup F_2 \cup \cdots \cup F_k)$, for $k = 1, 2, \ldots, |E|$. Then each G_k has at most $k(|V| - 1)$ edges and satisfies*

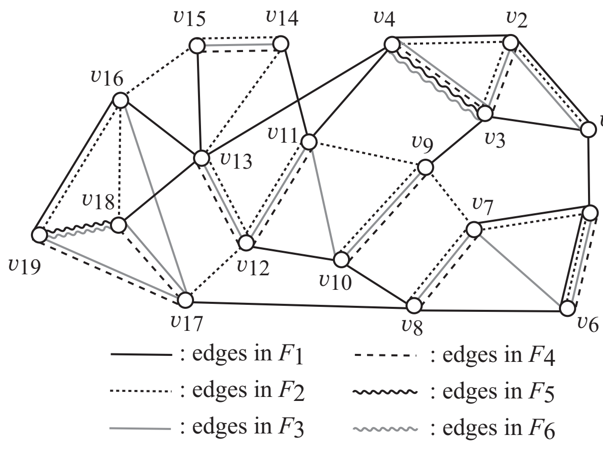
(i) *$\lambda_{G_k}(u, v) \ge \min\{\lambda_G(u, v), k\}$ for all $u, v \in V$,*

(ii) *$\kappa_{G_k}(u, v) \ge \min\{\kappa_G(u, v), k\}$ for all $u, v \in V$ if G is simple.* ∎

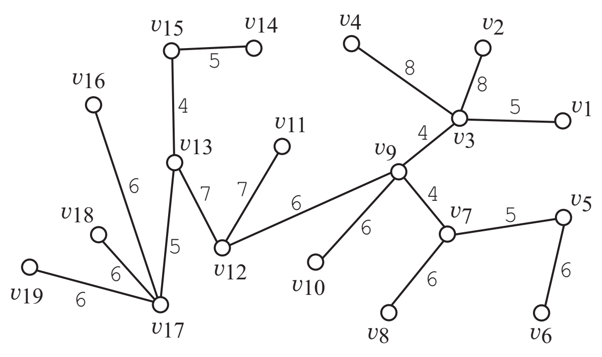
Since the above decomposition of G into forests $F_1, \ldots, F_{|E|}$ can be found in linear time [10, 29], such a sparse spanning subgraph G_k is widely used as a fast preprocessing for sparsifying a given graph G, in order to reduce the time complexity of many graph connectivity algorithms (see [13, 14, 17, 24, 27] for its applications).

5. Network Structures

In this section, we review four types of data structures, Gomory-Hu trees, maximal components, extreme sets and cactus representations, which all represent certain structural information of a given graph. The first two structures can be constructed by applying a maximum flow algorithm $O(n)$ times, while we will see that the third and last ones can be obtained by computing an MA ordering $O(n)$ times.



(a)



(b)

Figure 2: (a) A decomposition of the edge set of the multigraph G in Figure 1 into spanning forests $F_1, \ldots, F_6$, (b) A Gomory-Hu tree T for the graph G in Figure 1

5.1. Gomory-Hu trees

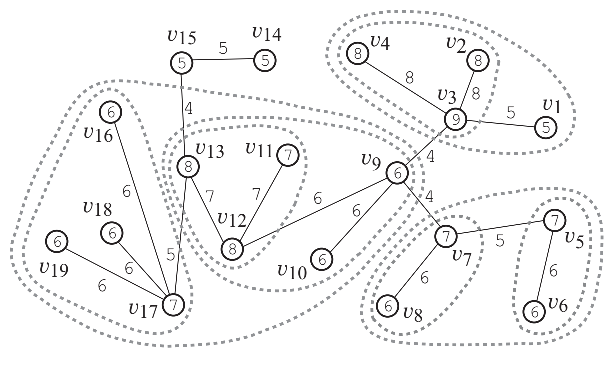
For an edge-weighted graph $G = (V, E)$, an edge-weighted tree $T = (V, F)$ on V (not necessarily a subgraph of G) is called a *flow equivalent tree* of G if

$$\lambda_T(u, v) = \lambda_G(u, v) \text{ for every pair of } u, v \in V.$$

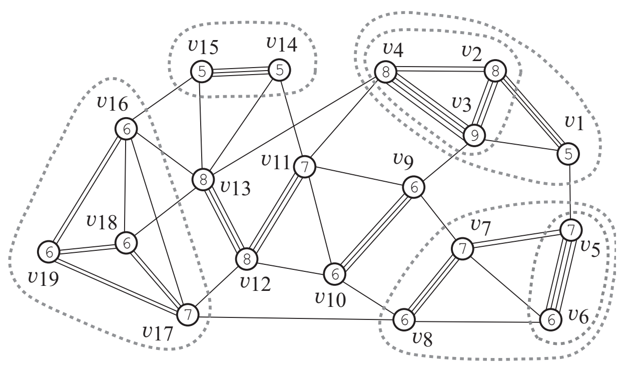
The condition implies that the minimum edge weight in the unique path between u and v in T is equal to $\lambda_G(u, v)$. A flow equivalent tree T is called a *Gomory-Hu tree* of G if, for each edge $e = (u, v)$ in T,

$$\text{the cut } \{X, V - X\} \text{ generated by } e \text{ in } T \text{ satisfies } d_G(X) = c_T(u, v).$$

Figure 2(b) shows a Gomory-Hu tree for the graph $G = (V, E)$ in Figure 1. R. E. Gomory and T. C. Hu [16] have shown that a Gomory-Hu tree can be constructed by applying a maximum flow algorithm $O(n)$ times.



(a)



(b)

Figure 3: (a) The set $\mathcal{Y}(G)$ of maximal components for the graph G in Figure 1, (b) The set $\mathcal{X}(G)$ of extreme sets for the graph G in Figure 1

5.2. Maximal components

Let $G = (V, E)$ be an edge-weighted graph. For a given $\ell \in \Re_+$, an *ℓ-edge-connected component* of G is defined to be a subset X of V such that (i) $\lambda_G(u, u') \geq \ell$ for any $u, u' \in X$ and (ii) for any $u \in X$ and $v \in V-X$, $\lambda_G(u, v) < \ell$ (i.e., X is inclusion-wise maximal subject to (i)). Observe that all ℓ-edge-connected components give rise to a partition of V.

An ℓ-edge-connected component $X \subseteq V$ is called *maximal* (with respect to ℓ) if $\ell = \min_{u,v \in X} \lambda_G(u, v)$. A set $X \subseteq V$ is simply called a *maximal component* if it is a maximal ℓ-edge-connected component for some ℓ (note that the set of maximal ℓ-edge-components may not be a partition of V). Let $\mathcal{Y}(G)$ denote the set of all maximal components. We see that $\mathcal{Y}(G)$ is a laminar family. For any two maximal components $X, Y \in \mathcal{Y}(G)$, $X \subset Y$ can hold only when X is an ℓ-edge-connected component and Y is an h-edge-connected component for some ℓ, h with $\ell > h$. Figure 3(a) shows the set $\mathcal{Y}(G)$ of maximal components of the graph $G = (V, E)$ in Figure 1, where $\mathcal{Y}(G)$ is depicted on the Gomory-Hu tree of the graph and the number inside the circle for each vertex v_i indicates the cut size $d_G(v_i)$.

We easily construct the family $\mathcal{Y}(G)$ of maximal components from any flow equivalent tree $T = (V, F)$ of G. Let $\lambda_1 < \lambda_2 < \cdots < \lambda_p$ be the distinct values in the edge weights in T, which are also the distinct values ℓ such that there is a maximal ℓ-edge-connected component of G. By definition, for any λ_i, each connected component T' in $T - \{e \in F \mid c_T(e) < \lambda_i\}$ corresponds to a λ_i-edge-connected component X of G (i.e., $V(T') = X$). Such an X is a maximal λ_j-edge-connected component for the $\lambda_j = \min_{u,v \in X} \lambda_G(u, v) (\geq \lambda_i)$. From this observation, we can construct the set $\mathcal{Y}_i$ of maximal λ_i-edge-components as follows. By letting $F_i = \{e \in F \mid c_T(e) = \lambda_i\}$, $i \in \{1, 2, \ldots, p\}$, and $\mathcal{C}$ be the set of graphs each of which consists of a single vertex $v \in V$, we repeat the next procedure in the order of $i = p, p-1, \ldots, 1$. Join connected components in $\mathcal{C}$ via edges in F_i, let $\mathcal{Y}_i := \{V(T') \mid$ newly created components T' in the i-th iteration$\}$ and $\mathcal{C}$ be the set of all current components. Sorting the weights of edges in T takes $O(n \log n)$ time, and the total time for joining components is $O(n \log n)$ (since each join can be executed in $O(\log n)$ time with an appropriate data structure for union-find operations).

Conversely, it is not difficult to see that a flow equivalent tree of a graph G can be computed from the family $\mathcal{Y}(G)$ of maximal components. However, in general, a Gomory-Hu tree cannot be constructed only from $\mathcal{Y}(G)$ without knowing G (for example, if G is a tree with unit edge weights, then $\mathcal{Y}(G)$ consists of singlton sets $\{v\}$, $v \in V$).

5.3. Extreme sets

A nonempty proper subset X of V is called an *extreme set* of an edge-weighted graph G if $d_G(X) < d_G(Y)$ for all nonempty proper subsets Y of X. We denote by $\mathcal{X}(G)$ the family of all extreme sets of G. Any singleton set $\{v\}$ with a vertex v is an extreme set, which we call *trivial*. By definition any subset $X \subseteq V$ contains an extreme set $X' (\subseteq X)$ such that $d_G(X') \leq d_G(X)$. Also note that there are at least two extreme sets X and Y such that $d_G(X) = d_G(Y) = \lambda(G)$ and $X \cap Y = \emptyset$, because, for any minimum cut $\{X, V-X\}$, each of X and $V-X$ contains an extreme set.

Lemma 5.1 *For any graph G, no two extreme sets in $\mathcal{X}(G)$ intersect each other (hence $\mathcal{X}(G)$ is laminar).*

Proof: Let X and Y be two subsets of V that intersect each other. Then $d_G(X) + d_G(Y) \geq d_G(X - Y) + d_G(Y - X)$ holds. Thus if X is an extreme set, then $d_G(X) < d_G(X - Y)$ holds and thereby $d_G(Y) > d_G(Y - X)$, implying that Y cannot be an extreme set. ∎

Figure 3(b) shows the extreme sets for the graph G in Figure 1, where $d_G(\{v_1, v_2, v_3, v_4\}) = d_G(\{v_5, v_6, v_7, v_8\}) = d_G(\{v_{14}, v_{15}\}) = \lambda(G)$ holds (trivial extreme sets are not enclosed by broken lines). D. Naor *et al.* observed the following characterization.

Lemma 5.2 [40] *Every extreme set $X \in \mathcal{X}(G)$ is a maximal ℓ-edge-connected component for $\ell = \min_{u,v\in X} \lambda_G(u, v)$.* ∎

Based on this, we can easily obtain the family $\mathcal{X}(G)$ of all extreme sets in G from the family of $\mathcal{Y}(G)$ of all maximal components in G. Note that a maximal component $Y \in \mathcal{Y}(G)$ is not an extreme set only if there is a nonempty and proper subset $Y' \subset Y$ such that $d_G(Y') \le d_G(Y)$, i.e., such an extreme set Y' exists (recall that any subset contains at least one extreme set). Therefore, a maximal component $Y \in \mathcal{Y}(G)$ is an extreme set if and only if $d_G(Y) < f_G(Y')$ for all maximal components with $Y' \subset Y$. Let $\mathcal{T}$ be a rooted tree that represents the laminar family $\mathcal{Y}(G)$. We can discard non-extreme sets from $\mathcal{Y}(G)$ in the time to traverse the tree $\mathcal{T}$. Thus, we can identify $\mathcal{X}(G) \subseteq \mathcal{Y}(G)$ in $O(n)$ time if $\mathcal{T}$ and $\{d_G(Y) \mid Y \in \mathcal{Y}(G)\}$ are available.

Extreme sets have the following usefull property.

Lemma 5.3 *For a graph $G = (V, E)$, a weight function $b : V \to \Re_+$ and a real $k \in \Re$, $d_G(Y) + \sum_{v\in Y} b(v) \ge k$ for all $Y \in 2^V - \{\emptyset, V\}$ if and only if $d_G(X) + \sum_{v\in X} b(v) \ge k$ for all $X \in \mathcal{X}(G)$.*

Proof: It suffices to show that for each set $Y \in 2^V - \{\emptyset, V\}$, there is an extreme set X such that $d_G(X) + \sum_{v\in X} b(v) \le d_G(Y) + \sum_{v\in Y} b(v)$. A set $Y \in 2^V - \{\emptyset, V\}$ contains an extreme set $X \subseteq Y$ with $d_G(X) \le d_G(Y)$, for which $d_G(X) + \sum_{v\in X} b(v) \le d_G(Y) + \sum_{v\in Y} b(v)$ holds by $\sum_{v\in Y-X} b(v) \ge 0$, as required. ∎

5.4. Cactus representations

We denote by $\mathcal{C}(G)$ the set of all minimum cuts in an edge-weighted graph G. For example, the graph G in Figure 4(a) has the minimum cut size 4, where the number of lines between two vertices represents weight of the edge between them.

A connected graph is called a *cactus* if each edge belongs to exactly one cycle, where a pair of multiple edges with the same end vertices is treated as a cycle of length 2. A graph consisting of a single vertex is called a *trivial* cactus. Thus, every pair of cycles, if any, in a cactus has at most one vertex in common.

For a given graph G, we introduce an unweighted cactus $\mathcal{R}$ and a mapping $\varphi : V(G) \to V(\mathcal{R})$. Throughout this paper, we shall use the term "vertex" to denote an element in $V(G)$, and the term "node" to denote an element in $V(\mathcal{R})$. A set $V(\mathcal{R})$ may contain a node x such that $V(G)$ contains no vertex v with $\varphi(v) = x$, and such a node x is called an *empty node.* Any non-trivial cactus $\mathcal{R}$ satisfies $\lambda(\mathcal{R}) = 2$. Let $\mathcal{C}(\mathcal{R})$ denote the set of all minimum cuts of $\mathcal{R}$. Thus, $\{S, V(\mathcal{R}) - S\} \in \mathcal{C}(\mathcal{R})$ holds if and only if $E_{\mathcal{R}}(S, V(\mathcal{R}) - S)$ is a set of two edges belonging to the same cycle in $\mathcal{R}$.

Definition 5.1 *For a given subset $\mathcal{C}' \subseteq \mathcal{C}(G)$ of minimum cuts, a pair $(\mathcal{R}, \varphi)$ of a cactus $\mathcal{R}$ and a mapping φ is called a cactus representation for $\mathcal{C}'$ if it satisfies the following* (i) *and* (ii).

(i) *For an arbitrary minimum cut $\{S, V(\mathcal{R}) - S\} \in \mathcal{C}(\mathcal{R})$, the cut $\{X, \overline{X}\}$ defined by $X = \{u \in V(G) \mid \varphi(u) \in S\}$ and $\overline{X} = \{u \in V \mid \varphi(u) \in V(\mathcal{R}) - S\}$ belong to $\mathcal{C}'$*

(ii) *Conversely, for any minimum cut $\{X, \overline{X}\} \in \mathcal{C}'$, there exists a minimum cut $\{S, V(\mathcal{R}) - S\} \in \mathcal{C}(\mathcal{R})$ such that $X = \{u \in V \mid \varphi(u) \in S\}$ and $\overline{X} = \{u \in V \mid \varphi(u) \in V(\mathcal{R}) - S\}$.* ∎

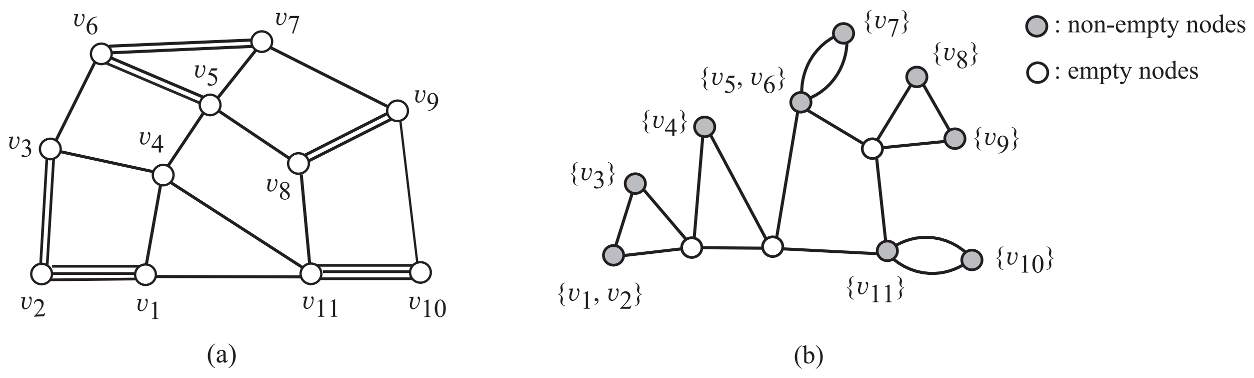


Figure 4: Illustration for (a) an edge-weighted graph G and (b) a cactus representation $\mathcal{R}$ for $\mathcal{C}(G)$ of the graph G

E. A. Dinits *et al.* [5] have proven that every graph G admits a cactus representation for $\mathcal{C}(G)$ such that the number of empty nodes is $O(n)$. For example, Figure 4(b) shows a cactus representation for all minimum cuts of the graph in Figure 4(a), where unshaded circles stand for empty nodes.

6. Computing Extreme Sets

In this section, we show an $O(mn + n^2 \log n)$ time algorithm [28] for finding all extreme sets of a given graph G without using a Gomory-Hu tree.

For a graph $G = (V, E)$ and a weight function $b : V \to \Re_+$, a star augmentation $G + b$ is called the *k-regular star augmentation* if $d_{G+b}(v) = \max\{k, d_G(v)\}$ (i.e., $b(v) = \max\{0, k - d_G(v)\}$) for all $v \in V$.

Lemma 6.1 *Let $X \in \mathcal{X}(G)$ be a non-trivial extreme set of a graph G, and v, w be two vertices. Then X does not separate v and w if $\lambda_{G+b}(v, w) \geq k$ holds in the k-regular star augmentation $G + b$ for some real k with $d_G(X) < k \leq \min_{v \in X} d_G(v)$.*

Proof: By $k \leq \min_{v \in X} d_G(v)$, $d_{G+b}(X) = d_G(X)$. Hence $|X \cap \{v, w\}| = 1$ would imply $\lambda_{G+b}(v, w) \leq d_{G+b}(X) = d_G(X) < k$, a contradiction to the assumption $\lambda_{G+b}(v, w) \geq k$. ∎

Lemma 6.2 [32] *For a graph $G = (V, E)$ and a real $K \geq 0$, let $G + b$ be the K-regular star augmentation, and $\sigma = (v_0 = s, v_1, v_2, \ldots, v_{n-1}, v_n)$ be an MA ordering σ starting with s in $G + b$. Then $\lambda_{G+b'}(v_{n-1}, v_n) \geq k$ holds in the k-regular star augmentation $G + b'$ for any k with $0 \leq k \leq K$.*

Proof: Let k be a real with $0 \leq k \leq K$, and $G + b'$ be the k-regular star augmentation of G. Let $\delta = K - k$, and B be the δ-skin of the K-regular star augmentation $G + b$. By construction of $G + b'$ and B, we see that $c_{G+b'}(v, w) \geq c_B(v, w)$ for all $v, w \in V \cup \{s\}$. In particular, $\lambda_{G+b'}(v_{n-1}, v_n) \geq \lambda_B(v_{n-1}, v_n)$. By applying Lemma 4.1 to $\delta = K - k$, we have $\lambda_B(v_{n-1}, v_n) = d_{G+b}(v_n) - \delta \geq K - \delta = k$. Therefore $\lambda_{G+b'}(v_{n-1}, v_n) \geq k$, as required. ∎

Lemma 6.3 *For a graph $G = (V, E)$ and a real $K \geq \max_{v \in V(G)} d_G(v)$, let $G + b$ be the K-regular star augmentation, and $\sigma = (v_0 = s, v_1, v_2, \ldots, v_{n-1}, v_n)$ be an MA ordering σ starting with s in $G + b$. Then no non-trivial extreme set X in G separates v_n and v_{n-1}.*

Proof: For each non-trivial extreme set X in G, there is a real k_X with $d_G(X) < k_X \leq \min_{v \in X} d_G(v)$. By $K \geq \max_{v \in V(G)} d_G(v)$, $k_X \leq K$ holds. By Lemma 6.2, $\lambda_{G+b'}(v_{n-1}, v_n) \geq$

k_X holds in the k_X-regular star augmentation $G+b'$ of G. Then by Lemma 6.1, X does not separate v_n and v_{n-1}. ∎

For $K = \max_{v\in V(G)} d_G(v)$ and the last two vertices v and w in an MA ordering starting with s in the K-regular star augmentation $G+b$, we see by Lemma 6.3 that the v and w can be contracted into a single vertex, say z, without losing any non-trivial extreme sets in G (since any non-trivial extreme set X does not separate $\{v,w\}$). Notice that for the resulting vertex z, the set of all vertices contracted into z may be an extreme set in the original graph G, and will be retained as a candidate of an extreme set of G before executing the same procedure of contracting a vertex pair in the resulting graph. After repeating the procedure until the graph has only two vertices, we can obtain the set of extreme sets of G by discarding all non-extreme sets from the set of candidates. The algorithm can be described as follows.

```
Algorithm **EXTREME**
Input: A graph $G = (V, E)$.
Output: The family $\mathcal{X}(G)$ of extreme sets of $G$.
1  $\mathcal{X} := \{\{v\} \mid v \in V\}$; $G' := G$;
2  while $|V(G')| \geq 3$ do
3      $K := \max_{v\in V(G')} d_{G'}(v)$;
4      Starting from $s$, find an MA ordering $\sigma$ in the $K$-regular star augmentation
       $G' + b$ of $G'$;
5      Contract the last two vertices $v, w \in V(G')$ in $\sigma$ into a single vertex $z$,
       and let $G'$ denote the resulting graph;
6      Let $X_z$ denote the set of vertices in $V$ that have been contracted into $z$ so far,
       and set $\mathcal{X} := \mathcal{X} \cup \{X_z\}$;
7  end; /* while */
8  $\mathcal{X} := \mathcal{X} - \{X \in \mathcal{X} \mid d_G(Y) \leq d_G(X),\ Y \subset X$ for some $Y \in \mathcal{X}\}$.
9  Output $\mathcal{X}(G) := \mathcal{X}$.
```

Theorem 6.1 *Algorithm EXTREME correctly finds the family of extreme sets of a given edge-weighted graph in $O(mn + n^2 \log n)$ time and $O(n + m)$ space.*

Proof: Since $\mathcal{X}$ in line 8 is a laminar family, $|\mathcal{X}| \leq 2n$ holds. The time and space bounds are immediate from the complexity of computing MA orderings. We show the correctness. As observed in the above, the set $\mathcal{X}$ after the while-loop contains all extreme sets of G. By definition, all sets discarded in line 8 cannot be extreme sets. For the correctness, it suffices to show that each set in the final set $\mathcal{X}$ is an extreme set of G. Assume that the final set $\mathcal{X}$ contains a non-extreme set X, for which there is an extreme set X' such that $X' \subset X$ and $d_G(X') \leq d_G(X)$. Since the $\mathcal{X}$ contains all extreme sets, X' is in the final $\mathcal{X}$. However, $X' \subset X$ and $d_G(X') \leq d_G(X)$ imply that X must have been discarded in line 8, a contradiction. Thus, the final $\mathcal{X}$ is $\mathcal{X}(G)$. ∎

7. Constructing Cactus Representations

In this section, we show that a cactus representation can be constructed in $O(mn+n^2 \log n)$ time by computing MA orderings $O(n)$ times.

7.1. (s,t)-cactus representations

We say that an edge $e = (s,t)$ in G is *critical* if $c_G(e) > 0$ and $\lambda_G(s,t) = \lambda(G)$.

Lemma 7.1 [25] *Let $e = (s, t)$ be a critical edge in a graph G. Then no two minimum cuts separating s and t cross each other. Hence, there is an ordered partition $(V_1, \ldots, V_r)$ of $V(G)$ such that the set of cuts of the form $\{V_1 \cup V_2 \cup \cdots \cup V_i, V_{i+1} \cup \cdots \cup V_r\}$ is equal to the set of all minimum cuts in $\mathcal{C}(G)$ that separate s and t.* ▮

Such an ordered partition in the lemma is called the (s, t) *minimum cut ordered partition* ((s, t)-MC-partition, for short).

Lemma 7.2 [25, 41] *Let (s, t) be a critical edge in a graph G. Then given an (s, t)-maximum flow, the (s, t)-MC-partition can be obtained in $O(n + m)$ time and space.* ▮

For example, the (s, t)-MC-partition $\pi_{(s,t)}$ with $(s, t) = (v_1, v_{11})$ of the graph G in Figure 4(a) is given by $(V_1 = \{v_1, v_2\}, V_2 = \{v_3\}, V_3 = \{v_4\}, V_4 = \{v_5, v_6, v_7\}, V_5 = \{v_8, v_9\}, V_6 = \{v_{10}, v_{11}\})$, as shown in Figure 5(a).

Let π be a partition $\{V_1, V_2, \ldots, V_r\}$ (or an ordered partition $(V_1, V_2, \ldots, V_r)$) of $V(G)$. We say that a cut $\{X, \overline{X}\}$ is *compatible* with π if

$$X = \cup_{i \in I} V_i \text{ for some } I \subset \{1, 2, \ldots, r\},$$

and that a cut $\{X, \overline{X}\}$ is *indivisible* with π if

$$X \subset V_i \text{ for some } i \in \{1, 2, \ldots, r\}.$$

Notice that any cut non-crossing with π is either compatible or indivisible with π. We denote by $\mathcal{C}_{comp}(\pi)$ (resp., $\mathcal{C}_{indv}(\pi)$) the set of all minimum cuts in $\mathcal{C}(G)$ that are compatible with π (resp., indivisible with π). H. Nagamochi and T. Kameda [36] have proven the following properties.

Lemma 7.3 [36] *Let (s, t) be a critical edge in a graph G, and $\pi_{(s,t)}$ be the (s, t)-MC-partition over $\mathcal{C}(G)$. Then any minimum cut $\{X, \overline{X}\} \in \mathcal{C}(G)$ is either compatible or indivisible with $\pi_{(s,t)}$ (i.e., $\mathcal{C}(G) = \mathcal{C}_{comp}(\pi_{(s,t)}) \cup \mathcal{C}_{indv}(\pi_{(s,t)})$).* ▮

Note that $\mathcal{C}_{comp}(\pi_{(s,t)})$ may contain a minimum cut that does not separate s and t.

Theorem 7.1 [36] *Let (s, t) be a critical edge in a graph G, and $\pi_{(s,t)}$ be the (s, t)-MC-partition. There exists a cactus representation $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$ for all minimum cuts in $\mathcal{C}_{comp}(\pi_{(s,t)})$, which we call an* (s, t)-cactus representation. *Moreover, given $\pi_{(s,t)}$, an (s, t)-cactus representation can be constructed in $O(n + m)$ time and space.* ▮

Figure 5(b) shows an (s, t)-cactus-representation with $(s, t) = (v_1, v_{11})$ of the graph G in Figure 4(a).

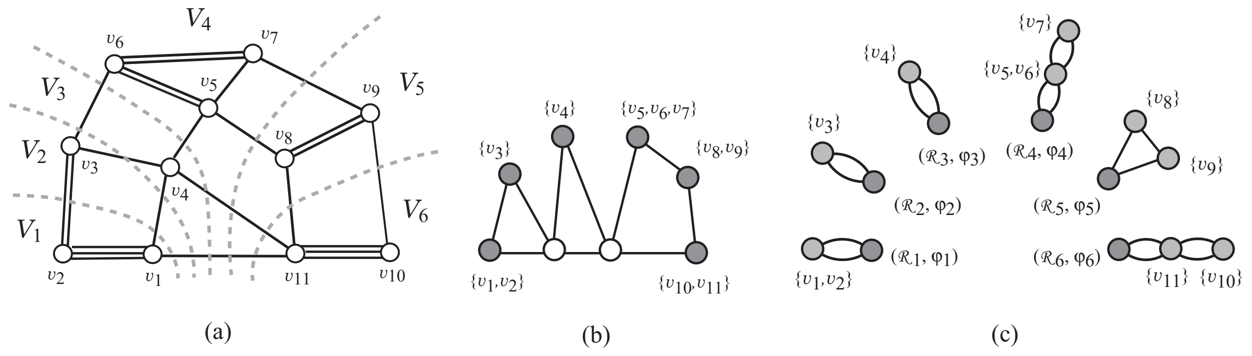
Lemma 7.4 [38] *In a graph G, an edge $e = (s, t)$ with $c_G(e) > 0$ satisfying the following* (i) *and* (ii) *can be found in $O(m + n \log n)$ time and $O(n + m)$ space.*

(i) *$\lambda_G(s, t)$ can be computed in $O(m + n \log n)$ time and $O(n + m)$ space.*

(ii) *If $\lambda_G(s, t) = \lambda(G)$, then an (s, t)-cactus representation can be constructed in $O(m + n \log n)$ time and $O(n + m)$ space.*

Proof: We first compute an MA ordering $\sigma = (v_1, v_2, \ldots, v_n)$ of G, and choose the vertex v_p with the largest index p such that v_p and v_n are joined by an edge with positive weight. Let $s = v_n$ and $t = v_p$. Note that $\sigma' = (v_1, v_2, \ldots, v_p, v_n)$ is an MA ordering in the graph $G' = G - \{v_{p+1}, v_{p+2}, \ldots, v_{n-1}\}$. Hence, by Theorem 4.1, $\lambda_G(s, t) \geq \lambda_{G'}(s, t) = d_{G'}(s, V(G') - s) = d_G(s, V(G) - s)$ holds. Since $\lambda_G(s, t) \leq d_G(s, V(G) - s)$, this shows (i).

Assume $\lambda_G(s, t) = \lambda(G)$. By Lemma 4.2, a maximum (s, t)-flow f can be found in $O(m)$ time. By Lemma 7.2 and Theorem 7.1, we can compute an (s, t)-cactus representation in linear time and space from the f. This proves (ii). ▮



V_1
V_2
V_3
V_4
V_5
V_6
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11}
(a)
$\{v_3\}$ $\{v_4\}$ $\{v_5, v_6, v_7\}$ $\{v_8, v_9\}$ $\{v_1, v_2\}$ $\{v_{10}, v_{11}\}$
(b)
$\{v_4\}$ $\{v_7\}$ $\{v_5, v_6\}$ $\{v_8\}$ $\{v_3\}$ $\{v_9\}$ $\{v_1, v_2\}$ $\{v_{11}\}$ $\{v_{10}\}$
$(\mathcal{R}_1, \varphi_1)$ $(\mathcal{R}_2, \varphi_2)$ $(\mathcal{R}_3, \varphi_3)$ $(\mathcal{R}_4, \varphi_4)$ $(\mathcal{R}_5, \varphi_5)$ $(\mathcal{R}_6, \varphi_6)$
(c)

Figure 5: Illustration for (a) (s,t)-MC-partition with $(s,t) = (v_1, v_{11})$, (b) (s,t)-cactus-representation with $(s,t) = (v_1, v_{11})$, and (c) cactus representations $(\mathcal{R}_i, \varphi_i)$ for $G_i = G/(V(G) - V_i)$

7.2. Cactus algorithm

We are ready to describe how to compute a cactus representation for all minimum cuts by a divide-and-conquer method. In what follows, we denote by G^* an input graph for which we want to construct a cactus representation. Let $\lambda = \lambda(G^*)$ for G^*. Based on Lemma 7.4, we construct a cactus representation for $\mathcal{C}(G^*)$ of G^* recursively as follows. We first choose an edge (s,t) in Lemma 7.4. If $\lambda_G(s,t) > \lambda$ (i.e., no minimum cut in G separates s and t), then we contract vertices s and t, and set $G := G/\{s,t\}$. If $\lambda_G(s,t) = \lambda$, then by Theorem 7.1 we can obtain an (s,t)-MC-partition $\pi_{(s,t)} = (V_1, \ldots, V_r)$ and an (s,t)-cactus representation $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$ in G. By Lemma 7.3, all minimum cut compatible with $\pi_{(s,t)}$ are represented by $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$, and each minimum cut $\{X, V(G) - X\}$ indivisible with $\pi_{(s,t)}$ satisfies $X \subset V_i$ for some $V_i \in \pi_{(s,t)}$. For each $i = 1, 2, \ldots, r$, we contract $V(G) - V_i$ into a single vertex $\overline{v}_i$ letting $G_i := G/(V(G) - V_i)$ be the resulting graph (note that $\lambda(G_i) \geq \lambda(G)$). Assume that, for each G_i, a cactus representation $(\mathcal{R}_i, \varphi_i)$ of G_i has been obtained in a recursive way, where we let $(\mathcal{R}_i, \varphi_i)$ be the trivial cactus if $\lambda(G_i) > \lambda(G)$. Then any minimum cut in G is represented in at least one of the representations $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$, $(\mathcal{R}_1, \varphi_1), \ldots, (\mathcal{R}_r, \varphi_r)$. Moreover, we can combine these representations into a single representation. For example, Figure 5(c) shows cactus representations $(\mathcal{R}_i, \varphi_i)$ for $G_i = G/(V(G) - V_i)$ obtained in the (s,t)-MC-partition in Figure 5(a). Observe that the cactus representation in Figure 4(b) can be obtained by attaching the cacti in Figure 5(c) to the (s,t)-cactus representation in Figure 5(b).

Let CACTUS(G', V^{old}) denote a recursive procedure for computing a cactus representation for a given graph G', where V^{old} is specified as a subset of $V(G')$ so as to detect minimum cuts that have already been found. The entire algorithm is given as follows.

```
Algorithm CONSTRUCT
Input: A graph $G^*$.
Output: A cactus representation $(\mathcal{R}, \varphi)$ for $\mathcal{C}(G^*)$.
  Compute $\lambda := \lambda(G^*)$;
  $V^{old} := \emptyset$;
  $(\mathcal{R}, \varphi) :=$CACTUS$(G^*, V^{old})$.
```

We now describe a procedure for CACTUS(G', V^{old}). If CACTUS(G', V^{old}) for a graph G' is invoked during execution of CACTUS(G'', V^{old}) for a graph G'', then we call graph G' a *child* of G'', and G'' the *parent* of G'. The parent-child relation between graphs G' and G''

induces a tree $\mathcal{T}$ rooted at the input graph G^*. The $\mathcal{T}$ represents the recursive computation during CONSTRUCT.

We remark that if $d_G(V_i, V(G) - V_i) = \lambda$ for a $V_i \in \pi_{(s,t)}$, then cut $\{V_i, \overline{v}_i\}$ remains to be a minimum cut in a child G_i even though the cut has already been detected in its parent G. The same minimum cut may appear in a descendent of G_i. A minimum cut is old in a graph G' (i.e., it has been detected in some ancestor of G') only if it separates a single vertex v from $V(G') - \{v\}$. Thus, we can check whether the current (s,t)-cactus representation $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$ contains a new minimum cut (i.e., a minimum cut that has not been detected in the ancestor) or not by marking the vertices $v \in V(G')$ "old" if v is some contracted vertex $\overline{v}_i$ in the ancestor. Note that any (s,t)-MC-partition $\pi_{(s,t)}$ with $|\pi_{(s,t)}| \geq 4$ represents a new minimum cut, since $\{V_1 \cup V_2, V_3 \cup \cdots \cup V_r\}$ is a minimum cut which does not separate a single vertex from the rest of vertices. If $|\pi_{(s,t)}| \in \{2,3\}$, then the (s,t)-cactus representation is shown to have at most three nodes [39]. Thus, with the set V^{old} of vertices marked "old", we can test whether the (s,t)-cactus representation $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$ contains a new minimum cut or not in $O(|V(\mathcal{R}_{(s,t)})|)$ time. Procedure CACTUS is then given as follows.

```
Procedure **CACTUS** $(G, V^{old})$
Input: A graph $G$ and a subset $V^{old} \subset V(G)$.
Output: A cactus representation $(\mathcal{R}, \varphi)$ for a set $\mathcal{C}'$ of minimum cuts such that
  $\mathcal{C}(G) - \{\{\overline{v}, V(G) - \overline{v}\} \mid \overline{v} \in V^{old}\} \subseteq \mathcal{C}' \subseteq \mathcal{C}(G)$.
1 if $|V(G)| = 1$ then return the trivial cactus $(\mathcal{R}, \varphi)$
2 else
3   Choose an edge $e = (s,t) \in E(G)$ with $c_G(e) > 0$;
4   if $\lambda_G(s,t) > \lambda$ or the $(s,t)$-cactus representation $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$ represents
      no minimum cut other than those $\{\overline{v}, V(G) - \overline{v}\}$, $\overline{v} \in V^{old}$
5   then
6     $G := G/\{s,t\}$;
7     $V^{old} := V^{old} - \{s,t\}$;
8     return CACTUS$(G, V^{old})$
9   else
10    for each $V_i$ in the $(s,t)$-MC-partition $\pi_{(s,t)} = (V_1, V_2, \ldots, V_r)$ do
11      Contract all vertices $V(G) - V_i$ into a single vertex $\overline{v}_i$;
12      $G_i := G/(V(G) - V_i)$;
13      $V_i^{old} := (V^{old} \cap V_i) \cup \{\overline{v}_i\}$;
14      $(\mathcal{R}_i, \varphi_i) :=$ CACTUS$(G_i, V_i^{old})$
15    end; /* for */
16    Combine cactus representations $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$, $(\mathcal{R}_1, \varphi_1)$, ..., $(\mathcal{R}_r, \varphi_r)$
      into a single cactus representation $(\mathcal{R}, \varphi)$;
17    return $(\mathcal{R}, \varphi)$
18  end /* if */
19 end. /* if */
```

We can show that this algorithm invoks a computation of an MA ordering $O(n)$ times.

Theorem 7.2 [38] *A cactus representation for all minimum cuts in an edge-weighted graph G can be constructed in $O(mn + n^2 \log n)$ time and $O(n + m)$ space.* ∎

7.3. Increasing edge-connectivity by one

As an application of cactus structures, we in this subsection consider the problem of augmenting a given unweighted multigraph $G = (V, E)$ to a multigraph $G + F = (V, E \cup F)$ such that $\lambda(G + F) = \lambda(G) + 1$ by adding a smallest set F of new edges.

A cut $X \subset V$ in a graph $G = (V, E)$ is called a *minimum cut* if $d_G(X) = \lambda(G)$. Furthermore, a minimum cut $Z \subset V$ is called a *minimal* minimum cut if no proper subset X of Z is a minimum cut. Let $\mathcal{M}(G)$ denote the set of all minimal minimum cuts in a graph G. We can identify the $\mathcal{M}(G)$ by computing a cactus representation $(\mathcal{R}, \varphi)$ of G. Each non-empty node $x \in V(\mathcal{R})$ with degree 2 corresponds to a minimal minimum cut $\{\varphi^{-1}(x), V - \varphi^{-1}(x)\} \in \mathcal{C}(G)$ and vice versa.

Lemma 7.5 *Let $G = (V, E)$ be a multigraph. Then:*

(i) *For any minimum cut $X \subset V$ in G, there is a minimal minimum cut Z in G such that $Z \subseteq X$.*

(ii) *For any minimal minimum cut Z in G, $\lambda_G(u, v) > \lambda(G)$ holds for all $u, v \in Z$.* ▮

Note that any two minimal minimum cuts are disjoint. From this, we have the following observation on the number of edges to be added to increase the edge-connectivity.

Lemma 7.6 *For a multigraph $G = (V, E)$ and a set F of new edges, if $\lambda(G+F) \geq \lambda(G)+1$ then $|F| \geq \lceil |\mathcal{M}(G)|/2 \rceil$.*

Proof: Consider any F such that $\lambda(G + F) \geq \lambda(G) + 1$. For each cut $X \in \mathcal{M}(G)$, F must contain an edge which is incident to a vertex in X, since otherwise $\lambda(G + F) \leq d_{G+F}(X) = d_G(X) = \lambda(G)$ would hold. Therefore, by the disjointness of cuts in $\mathcal{M}(G)$, the number of endvertices of edges in F is at least $|\mathcal{M}(G)|$, implying $|F| \geq \lceil |\mathcal{M}(G)|/2 \rceil$. ▮

Lemma 7.7 [40] *Let $\mathcal{M}' = \mathcal{M}(G)$ if $|\mathcal{M}(G)|$ is even, and $\mathcal{M}' = \mathcal{M}(G) \cup \{\{z^*\}\}$ if $|\mathcal{M}(G)|$ is odd, where z^* is a vertex arbitrarily chosen from V. Then:*

(i) *The subsets in $\mathcal{M}'$ have a cyclic ordering $(Z_1, \ldots, Z_\ell)$ (where the last Z_ℓ is followed by the first Z_1) such that, for any minimum cut X in G, all cuts $Z_i \subseteq X$ have consecutive indices i.*

(ii) *For a cyclic ordering $(Z_1, \ldots, Z_\ell)$ in* (i)*, let z_i be a vertex arbitrarily chosen from each $Z_i \in \mathcal{M}'$. Then adding to G new $\ell/2$ edges of unit weights, $(z_i, z_{i+\ell/2})$, $i = 1, \ldots, \ell/2$ increases the edge-connectivity up to $\lambda(G) + 1$.*

Proof: Let $(\mathcal{R}, \varphi)$ be a cactus representation for all minimum cuts in G. Since the degree of each node in cactus $\mathcal{R}$ is even, $\mathcal{R}$ has an Eulerian walk τ. Starting with a node x_0 in $\mathcal{R}$, we traverse τ during which we compute a desired cyclic ordering for subsets in $\mathcal{M}'$ as follows. Let $(Z_1, Z_2, \ldots, Z_\ell)$ be a cyclic ordering of subsets in $\mathcal{M}'$ that are labeled in such a way that the node $\varphi(z)$ with $z \in Z_i$ appears earlier than the node $\varphi(z')$ with $z \in Z_j$ for any $j > i$ during the traversal. For any minimum cut $\{X, V - X\}$ in G, there is a pair of arcs in $\mathcal{R}$ that generates a cut $\{U, V(\mathcal{R}) - U\}$ such that $\{\varphi(x) \mid x \in X\} \subseteq U$ and $\{\varphi(v) \mid x \in V - X\} \subseteq V(\mathcal{R}) - U$. All subsets $Z \in \mathcal{M}'$ with $Z \subseteq X$ map to nodes in U, and they must have consecutive numbers by the traversal of τ.

(ii) follows from (i) and Lemma 7.5. ▮

For example, consider a multigraph G in Figure 4(a), whose edge-connectivity is 4. The graph G has the set $\mathcal{M}(G)$ of minimal minimum cuts $Z_1 = \{v_1, v_2\}$, $Z_2 = \{v_3\}$, $Z_3 = \{v_4\}$, $Z_4 = \{v_7\}$, $Z_5 = \{v_8\}$, $Z_6 = \{v_9\}$ and $Z_7 = \{v_{10}\}$. Such a cyclic ordering of Lemma 7.7(i) in this example is given by $(Z_1, Z_2, \ldots, Z_7, Z_8 = \{v_{11}\})$, where v_{11} is chosen as z^*. By Lemma 7.7(ii), $F = \{(v_1, v_8), (v_3, v_9), (v_4, v_{10}), (v_7, v_{11})\}$ is an optimal solution to the graph G.

8. Edge-Connectivity Augmentation

The problem of increasing the edge- or vertex-connectivity of a given graph up to a specified target value by adding the smallest number of new edges is called a *connectivity augmentation problem*. The problem has been extensively studied recently (see [8, 33] for a survey). In this section, we consider the *edge-connectivity augmentation problem with a degree constraint*, which asks to augment a given unweighted multigraph $G = (V, E)$ to a k-edge-connected multigraph $G + F$ with $d_{G+F}(u) \leq \beta(u)$, $u \in V$ by adding a smallest set F of new edges, where $k \geq 2$ is a specified integer and $\beta : V \to \mathbf{Z}_+$ is a given function. We represent G and $G + F = (V, E \cup F)$ as simple graphs with edges weighted by integers by storing each set of multiple edges with the same endvertices as an integer-weighted edge. Let n and m be the number of vertices and edges in the edge-weighted graph G. In the following, we show that an algorithm due to A. A. Benczúr and D. R. Karger [4] to this problem can be implemented to run in $O(mn + n^2 \log n)$ time by using the algorithms for computing a family of extreme sets and a cactus representation in the previous sections.

For a given target $k \in \mathbf{Z}_+$, we define the *deficit* $\mathrm{dft}(X)$ of a subset $X \subseteq V$ by

$$\mathrm{dft}(X) = \max\{k - d_G(X), 0\}.$$

For a weight function $a : V \to \Re$, we denote $\sum_{v \in X} a(v)$ by $a(X)$ for all $X \subseteq V$. A. A. Benczúr and D. R. Karger's algorithm [4] consists of three phases, each of which we show in the following subsections.

8.1. Phase-1: optimal star augmentation

We consider a star augmentation $G + b$ of a given multigraph $G = (V, E)$ such that

$$d_{G+b}(X)(= d_G(X) + b(X)) \geq k \text{ for all } X \subset V, \tag{8.1}$$

and

$$d_{G+b}(v)(= d_G(v) + b(v)) \leq \beta(v) \text{ for all } v \in V. \tag{8.2}$$

In phase-1, we find a star augmentation $G + b$ that minimizes $b(V)(= d_{G+b}(s))$ subject to (8.1) and (8.2). For this, we compute the family $\mathcal{X}(G)$ of extreme sets of G, and define $\mathcal{X}_k(G) = \{X \in \mathcal{X}(G) \mid d_G(X) < k\}$. Then by Lemma 5.3, (8.1) is equivalent to the next.

$$d_{G+b}(X) \geq k \text{ for all } X \in \mathcal{X}_k(G). \tag{8.3}$$

Let $\mathcal{T}$ be the tree representation for $\mathcal{X}_k(G)$. Starting with all nodes in $\mathcal{T}$ *unscanned* and $b(v) := 0$ for all $v \in V$, we repeatedly choose a lowest unscanned node X from $\mathcal{T}$ such that $\mathrm{dft}(X)(= d_G(X)+b(X)) < k$ holds for the current b, and increase $b(v)$, $v \in X$ (arbitrarily) so that $d_G(X)+b(X) = k$ holds. Note that if we failed to attain $d_G(X)+b(X) = k$ by increasing $b(v)$, $v \in X$ as much as possible under the degree constraint, then $d_G(X) + \beta(X) < k$ holds for the X, indicating that the problem is infeasible.

Let b be the final weight function obtained by the procedure. Consider a subset $X \in \mathcal{X}_k(G)$ such that $d_G(X) + b(X) = k$. Let $\mathcal{M}$ be the set of inclusion-wise maximal subsets among such subsets. Since all sets in $\mathcal{M}$ are pairwise disjoint, we see that

$$b(V)(= d_{G+b}(s)) = \sum_{X \in \mathcal{M}} \mathrm{dft}(X)$$

holds and that at least $\lceil b(V)/2 \rceil$ new edges are need to be added to G to obtain a k-edge-connected graph. If $d_{G+b}(s)$ is odd, then we add an edge between s and V so that (8.2) remains valid, making $d_{G+b}(s)$ even; if such an edge cannot be chosen, then the problem is infeasible.

For example, we consider a target $k = 8$ and a degree function $\beta(u) = 9$, $u \in V$, in the graph G in Figure 1. Then $\mathcal{X}_8(G)$ is given as shown in Figure 6(a), where an extreme set X with $\mathrm{dft}(X) \geq 2$ (resp., $\mathrm{dft}(X) = 1$) is enclosed by a black broken line (resp., by a gray line), and the number on the line indicates the $\mathrm{dft}(X)$ of the extreme set X. A final weight function b is given by $b(v_1) = 3$, $b(v_2) = b(v_5) = 1$, $b(v_6) = 2$, $b(v_7) = 1$, $b(v_8) = b(v_9) = b(v_{10}) = 2$, $b(v_{11}) = 1$, $b(v_{14}) = b(v_{15}) = 3$, $b(v_{16}) = 2$, $b(v_{17}) = 1$ and $b(v_{18}) = b(v_{19}) = 2$ (where $b(V) = 28$), and $\mathcal{M} = \{\{v_1, v_2, v_3, v_4\}, \{v_5, v_6\}, \{v_7\}, \{v_8\}, \{v_9\}, \{v_{10}\}, \{v_{11}\}, \{v_{14}\}, \{v_{15}\}, \{v_{16}\}, \{v_{17}\}, \{v_{18}\}, \{v_{19}\}\}$.

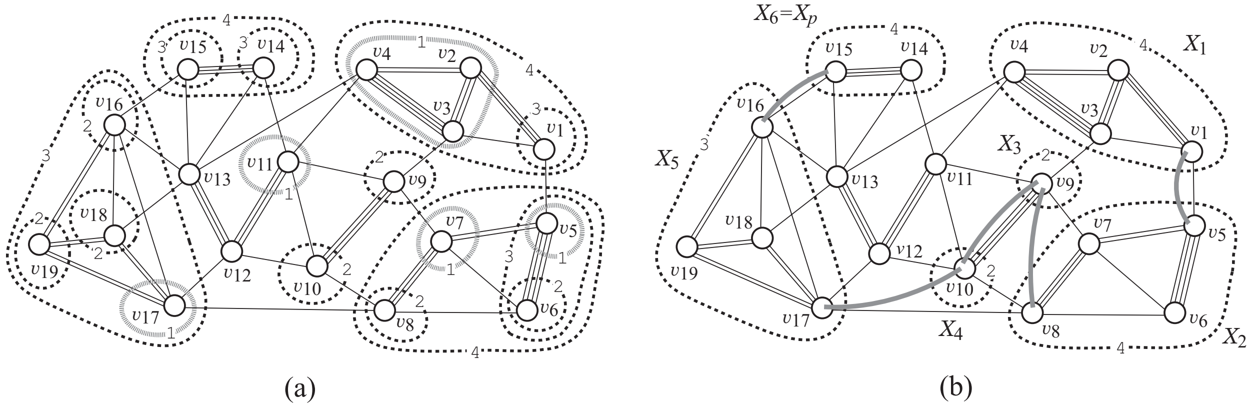


Figure 6: (a) The extreme sets in $\mathcal{X}_8(G)$ for the graph G in Figure 1 and target $k = 8$; (b) Maximal extreme sets $X_1, X_2, \ldots, X_p$ with $\mathrm{dft}(X_i) \geq 2$ in (a), and a path augmentation $G + E'$

8.2. Phase-2: augmentation up to $k - 1$

Our goal is to find a set F of new edges such that the augmented graph $G+F$ and the graph (V, F) respectively satisfy $\lambda(G + F) \geq k$ and $d_{(V,F)}(v) = b(v)$ for all $v \in V$. However, in phase-2, we only find a set F' of new edges such that $\lambda(G + F') = k - 1$ before we compute a set $F'' = F - F'$ of remaining new edges with $\lambda(G + F' \cup F'')$ in phase-3. In phase-2, we construct such an edge set F' by repeatedly choosing a new edge set E' with the following property, where we denote $d_{(V,E')}(X)$, $X \subseteq V$, by $d_{E'}(X)$ for convenience.

(i) $d_{E'}(v) \leq b(v)$ for all $v \in V$.
(ii) No edge in E' has both endvertices in any extreme set $X \in \mathcal{X}(G)$.
(iii) $\mathcal{X}(G + E') \subseteq \mathcal{X}(G)$.

Claim 8.1 [4] *For any new edge set E' satisfying the above conditions* (i)-(iii)*, the augmented graph $G' = G + E'$ and the reduced weight b' with $b'(v) = b(v) - d_{E'}(v)$, $v \in V$, remain to satisfy* (8.1) *and* (8.2). ∎

To find a new edge set E' satisfying (i)-(iii), we consider the extreme sets X with $\mathrm{dft}(X) \geq 2$, i.e., the extreme sets in $\mathcal{X}_{k-1}(G)$. Now we denote all inclusion-wise maximal extreme sets in $\mathcal{X}_{k-1}(G)$ by $X_1, X_2, \ldots, X_{p-1}, X_p$ so that $d_G(X_1)$ and $d_G(X_p)$ satisfy

$$d_G(X_p) = d_G(X_1) \leq \min\{d_G(X_i) \mid i = 1, 2, \ldots, p\},$$

where $d_G(X_p) = d_G(X_1) = \lambda(G)$ holds as observed before Lemma 5.3. Choose a vertex $u_1 \in X_1$, a vertex $\overline{u}_p \in X_p$, vertices $u_i, \overline{u}_i \in X_i$ (possibly $u_i = \overline{u}_i$), $i = 2, 3, \ldots, p - 1$ such that $b(u_i), b(\overline{u}_i) \geq 1$ if $u_i \neq \overline{u}_i$ $(2 \leq i \leq p-1)$ and $b(u_i) \geq 2$ (or $b(\overline{u}_i) \geq 2$) otherwise. Let

$$E' = \{(u_i, \overline{u}_{i+1}) \mid u_i \in X_i,\ \overline{u}_{i+1} \in X_{i+1},\ 1 \leq i \leq p-1\}$$

Hence $d_{E'}(X_1) = d_{E'}(X_p) = 1$ and $d_{E'}(X_i) = 2$ for all $i = 2, 3, \ldots, p-1$. By $\mathrm{dft}(X_i) \geq 2$, we can choose such an E', which forms a collection of vertex-disjoint paths. We call the graph $G + E'$ augmented by such an E' a *chain augmentation*. For the above example of the graph G in Figure 1 with $k = 8$ and $\beta(u) = 9$, $u \in V$, Figure 6(b) shows the maximal extreme sets $X_1, X_2 \ldots, X_p$ with $\mathrm{dft}(X_i) \geq 2$ and a chain augmentation $G + E'$, where the edges in E' are depicted by thick gray lines.

Claim 8.2 [4] *A chain augmentation $G + E'$ satisfies the above conditions* (i)-(iii). ▮

Therefore, we can augment a given graph G to a $(k-1)$-edge-connected graph by repeating the procedure: find a chain augmentation $G + E'$ in G, and update $G := G + E'$ and $b(v) := b(v) - d_{E'}(v)$ for all $v \in V$. A naive implementation of this algorithm would take $O(m + k)$ iterations of the procedure. To obtain a faster implementation, we try to use the same E' as many times as possible. An edge set E' can be reused in $G + E'$ if the following three conditions hold:
(t1) the updated weight b' satisfies $b'(v) \geq d_{E'}(v)$ for all $v \in V$,
(t2) all X_i still satisfy $\mathrm{dft}(X_i) \geq 2$ in $G + E'$,
(t3) all X_i remain extreme in $G + E'$.
Therefore, we can augment t copies of an edge set E' in G if t is the minimum of the following t_1, t_2 and t_3:

- To meet (t1) for the current weight b, t should be at most

$$t_1 = \min\{\lfloor b(v)/d_{E'}(v) \rfloor \mid \text{ an edge in } E' \text{ is incident to } v\}.$$

- Since (t2) must hold after adding $(t-1)$ copies of E', t should satisfy that $\mathrm{dft}(X_i) - (t-1) \geq 2$ $(i \in \{1, p\})$ and $\mathrm{dft}(X_i) - 2(t-1) \geq 2$ $(i = 2, 3, \ldots, p-1)$. Let

$$t_2 = \min\Big\{\min_{i=1,p} \mathrm{dft}(X_i) - 1,\ \min_{i=2,\ldots,p-1} \lfloor \mathrm{dft}(X_i)/2 \rfloor\Big\}.$$

- From (t3), each X_i should satisfy $d_G(X_i) + (t-1) \cdot d_{E'}(X_i) < d_G(Y) + (t-1) \cdot d_{E'}(Y)$ for all subsets $Y \subset X$ (recall that $d_G(X_i) < d_G(Y)$). Let $t_3 = \min\{r_1, r_2, \ldots, r_p\}$, where

$$r_i = \min\left\{\left\lceil \frac{d_G(Y) - d_G(X_i)}{d_{E'}(X_i) - d_{E'}(Y)} \right\rceil \;\Big|\; Y \in \mathcal{X}(G),\ Y \subset X_i,\ d_{E'}(X_i) - d_{E'}(Y) \geq 1\right\}.$$

Let us analyze the run time of phase-2. For a graph G and a weight b before phase-2, let

$$V_b = \{v \in V \mid b(v) > 0\}, \quad n_b = |V_b|, \text{ and } n_k = |\mathcal{X}_k(G)|.$$

By (8.1), each extreme set $X \in \mathcal{X}_k(G)$ contains a vertex in V_b. Hence $n_k \leq 2n_b - 2$. We see that, during the algorithm, t_1 becomes tight (i.e., $t_1 = \min\{t_1, t_2, t_3\}$ holds) at most n_b times, t_2 at most n_k times, and t_3 at most n_k times. Therefore, we repeat the procedure of finding a chain augmentation $O(n_b)$ times. Since each iteration can be implemented to run in $O(m + n \log n)$ time, phase-2 takes $O(mn + n^2 \log n)$ time.

Now consider the number of pairs of vertices that are joined by new edges in F'. After each iteration, we try to construct the next chain augmentation using as many edges in the previous E' as possible. We can reuse an edge $(u, v) \in E$ such that $b(u), b(v) \geq 1$ and the two maximal extreme sets X_i and X_j containing u and v respectively remain to be maximal extreme sets with $\mathrm{dft}(X_i), \mathrm{dft}(X_j) \geq 2$. With this observation, we see that an edge joining a new pair of vertices is introduced to E' only when it joins two maximal extreme set X' and X'' such that

(a) at least one of X' and X'' is new,
(b) both X' and X'' are old, but not joined in the previous iteration, or
(c) both X' and X'' are old and joined in the previous iteration.

Note that (b) occurs when a maximal extreme X with $\mathrm{dft}(X) \geq 2$ disappeared in the previous iteration, and that (c) occurs when $b(u)$ or $b(v)$ for the edge (u, v) that joined X' and X'' became 0 in the previous iteration. Therefore the number of pairs of vertices joined by edges in F' is at most

$$2n_k + n_k + (n_b - n') \leq 7n_b - n' - 6,$$

where n' denotes the number of vertices u with $b(u) \geq 1$ (in fact $b(u) = 1$) after the final iteration.

8.3. Phase-3: augmentation on a cactus

After phase-2, the remaining task is to increase the edge connectivity of the current graph $G+F'$ by one by adding a set F'' of new edges such that $d_{F''}(v) \leq b(v)$, $v \in V$ for the current weight b. Recall that condition (8.1) remains valid. Hence, for each minimal minimum cut $Z \in \mathcal{M}(G + F')$ (see section 7.3), we can choose a vertex v_Z with $b(v_Z) \geq 1$. Note that $b(V)$ is even. If $|\mathcal{M}(G + F')|$ is odd, then we can find a vertex z^* with $b(z^*) \geq 1$ from $V - \{v_Z \mid Z \in \mathcal{M}(G + F')\}$ (or $z^* = v_Z$ if $b(v_Z) \geq 2$ for some Z). By Lemma 7.7, we can find a set F'' of $\lceil |\mathcal{M}(G + F')|/2 \rceil$ new edges such that $\lambda(G + F' \cup F'') \geq \lambda(G + F') + 1 = k$ and $d_{F''}(v) \leq b(v)$, $v \in V$. Therefore, we obtain a set $F = F' \cup F''$ of new edges such that $\lambda(G + F) \geq k$ and $|F| \leq \lfloor b(V)/2 \rfloor$. Note that phase-3 introduces at most $\lceil n'/2 \rceil$ new edges. Hence F consists of at most $7n_b - n' - 6 + \lceil n'/2 \rceil \leq 7n_b - 6$ weighted edges. Thus, by Theorems 6.1 and 7.2, the entire run time is $O(n(m + n_b) + n^2 \log n)$.

Theorem 8.1 *For a multigraph $G = (V, E)$ stored as an integer-weighted graph, a weight function $\beta : V \to \mathbf{Z}_+$, and an integer $k \geq 2$, the edge-connectivity augmentation problem with degree constraint can be solved in $O(mn + n^2 \log n)$ time and $O(n + m)$ space. Moreover the number of new weighted edges added to G can be bounded from above by $7n - 6$.* ▮

8.4. Splitting algorithm

Let $G = (V, E)$ be a multigraph which has a designated vertex $s^* \in V$ with an even degree. For two edges $e_1 = (s^*, u_1)$ and $e_2 = (s^*, u_2)$, we say that a multigraph G' is obtained from G by *splitting e_1 and e_2 at s^** if two edges e_1 and e_2 are replaced with a single edge (u_1, u_2), where possibly $u_1 = u_2$ and in this case the split edge (u_1, u_2) is a self-loop, which will be simply removed. It is known as Lovász's theorem [26] that all edges incident at s^* can be split while preserving the edge-connectivity of G in $V - s$ (i.e., the resulting graph G', where the isolated s^* is neglected, satisfies $\lambda(G') = \min_{u,v \in V - s^*} \lambda_G(u, v)$). Such a complete splitting plays an important role in solving many graph connectivity problems such as the orientation problem (see [31]). The previously fastest deterministic algorithm for finding a complete splitting runs in $O((mn + n^2 \log n) \log n)$ time [31, 37]. By Theorem 8.1, this can be improved by factor of $O(\log n)$.

Theorem 8.2 *For a multigraph G stored as an integer-weighted graph, and a designated vertex s^*, there is a complete splitting at s^* such that the number of pairs of vertices joined* *by the split edges is at most* $7|\Gamma_G(s^*)| - 6$ *edges. Moreover such a splitting can be found in* $O(mn + n^2 \log n)$ *time and* $O(n+m)$ *space.*

Proof: Let $k = \min_{u,v \in V(G)-s^*} \lambda_G(u,v)$, $G^* = G - s^*$, $n_b = |\Gamma_G(s^*)|$, and $\beta(v) = c_G(s^*, v)$, $v \in V(G) - s^*$. By applying Theorem 8.1 to the graph G^* and weight function β, we can find a set F of new edges such that $\lambda(G^* + F) \geq k$ and $d_F(v) \leq \beta(v)$ for all $v \in V(G^*)$. (Note that the augmentation problem with G^* and β is feasible since the weight function $b = \beta$ satisfies (8.1) and (8.2).) We can regard that F is obtained by splitting $d_F(v)$ multiple edges (s^*, v), $v \in V(G^*)$. For each vertex $v \in V(G^*)$, we split the rest of $\beta(v) - d_F(v)$ multiple edges (s^*, v) into self-loops (v, v), which will be removed. Therefore, the number of pairs of vertices joined by split edges is at most $7n_b - 6$ by Theorem 8.1. The time and space complexities follow from Theorem 8.1. ∎

9. Source Location Problem

Problems of selecting the best location of facilities in a given network so as to satisfy a certain requirment are called *location problems*. Recently, the location problems with requirements measured by edge-connectivity, vertex-connectivity, or flow-amount have been studied extensively. The *source location problem* which asks to find an optimal set of sources in a graph under connectivity and/or flow-amount requirements is defined as follows. Given an edge-weighted graph $G = (V, E)$, a cost function $w : V \to \Re_+$, and a demand function $r : V \to \Re_+$, we want to choose a subset $S \subseteq V$ so as to

$$\begin{array}{ll} \text{Minimize} & \sum\{w(v) \mid v \in S\} \\ \text{subject to} & \psi(S, v) \geq r(v) \text{ for all } v \in V - S, \end{array}$$

where $\psi(S, v)$ is a measurement based on the edge-connectivity or the edge-connectivity between S and a vertex v.

9.1. Edge-connectivity requirement

The source location problem with the edge-connectivity requirement $\psi(S, v) = \lambda_G(S, v)$ in undirected graphs was first treated by H. Tamura *et al.* [45, 46]. They gave polynomial time algorithms for a uniform cost $w : V \to \{1\}$. K. Arata *et al.* [2] have proven that the problem is weakly NP-hard for a general cost $w : V \to \Re_+$. The problem with $\psi(S, v) = \lambda_G(S, v)$ in digraphs is not completely settled; the complexity status for the case of a uniform cost w and a uniform demand $r : V \to \{k\}$ is unknown so far. S. Honami *et al.* [18] have given an $O(n^2 m)$ time algorithm for an unweighted digraph with $\psi(S, v) = \lambda_G(S, v)$, a uniform cost w and a uniform demand $r : V \to \{k\}$ ($k \leq 3$). H. Ito *et al.* [21] have shown that the problem with a uniform cost w and a uniform demand $r : V \to \{k\}$ in an unweighted digraph can be solved in polynomial time if k is fixed.

In the sequel, we consider the source location problem with $\psi(S, v) = \lambda_G(S, v)$ for a general cost function $w : V \to \Re_+$, and a uniform demand $r : V \to \{k\}$ in an edge-weighted graph $G = (V, E)$. We call a subset $S \subseteq V$ *k-feasible* if $\lambda_G(S, v) \geq k$ for all $v \in V - S$.

Theorem 9.1 *Given a family* $\mathcal{X}(G)$ *of extreme sets of an edge-weighted graph* $G = (V, E)$*, the source location problem with the edge-connectivity requirement for a cost function* $w : V \to \Re_+$*, and a uniform demand* $r : V \to \{k\}$ *can be solved in* $O(n)$ *time.*

Proof: Let $\mathcal{X}_k(G) = \{X \in \mathcal{X}(G) \mid d_G(X) < k\}$. In the tree representation $\mathcal{T}_k$ of $\mathcal{X}_k(G)$, consider the set of leaf nodes, and let $X_1, X_2, \ldots, X_p$ be the sets in $\mathcal{X}_k(G)$ that correspond to these leaf nodes. Since any two X_i and X_j are disjoint, we see that, for any k-feasible

$S \subseteq V$, $S \cap X_i \neq \emptyset$, $i = 1, 2, \ldots, p$. Let $S^* = \{v_1, v_2, \ldots, v_p\}$ by choosing a vertex v_i with the minimum weight $w(v_i)$ from each X_i. Note that S^* is k-feasible since, for any cut $Y \subset V$ with $d_G(Y) < k$, there is an extreme set $X_i \subseteq Y$ with $d_G(X_i) \leq d_G(Y)$ that corresponds to a leaf node in $\mathcal{T}_k$. Also we easily see that S^* attains the minimum cost because any k-feasible set must contain at least one vertex from each X_i. ∎

9.2. Vertex-connectivity requirement

For the vertex-connectivity requirement $\psi(S, v) = \kappa_G(S, v)$, H. Ito *et al.* [20] have proven that the source location problem with a uniform cost $w : V \to \{1\}$ in an undirected graph is NP-hard. Contrary to this, H. Nagamochi *et al.* [35] considered the source location problem with a measurement $\psi(S, v) = \hat{\kappa}^+_G(S, v)$ in a digraph with a general cost $w : V \to \Re_+$, and a uniform demand $r : V \to \{k\}$ ($k \in \mathbf{Z}_+$), and gave an $O(\min\{k, \sqrt{n}\}nm)$ time algorithm, where in a digraph $\hat{\kappa}^+_G(S, v)$ (resp., $\hat{\kappa}^-_G(S, v)$) is defined to be the maximum number of vertex-disjoint directed paths from S to v (resp., from v to S) such that no two paths meet at the same vertex in S. From this, they also gave an $O(\min\{k, \sqrt{n}\}kn^2)$ time algorithm for solving the source location problem with a measurement $\psi(S, v) = \hat{\kappa}_G(S, v)$ in an undirected graph with a general cost $w : V \to \Re_+$ and a uniform demand $r : V \to \{k\}$. They further prove that the next source location problem can be solved in $O(n^4 m)$ time: given a digraph $G = (V, E)$, a general cost $w : V \to \Re_+$, two integers k and ℓ, find a minimum cost subset $S \subseteq V$ such that $\hat{\kappa}^+_G(S, v) \geq k$ and $\hat{\kappa}^-_G(S, v) \geq \ell$ for all $v \in V - S$. For a non-uniform demand case in the problem with $\psi(S, v) = \hat{\kappa}_G(S, v)$, T. Ishii *et al.* [19] gave a linear time algorithm to a demand function $r : V \to \{0, 1, 2, 3\}$ and showed that the problem is NP-hard if there exists a vertex $v \in V$ with $r(v) \geq 4$.

H. Ito *et al.* [20] considered the source location problem with a measurement "$\kappa_G(S, v) \geq k$ and $\lambda_G(S, v) \geq \ell$ for all $v \in V - S$" and a uniform cost $w : V \to \{1\}$ in an unweighted graph $G = (V, E)$, and presented an $O(\min\{\ell mn, \ell^2 n^2, mn^3\})$ time algorithm for $k \leq 2$.

In what follows, we show the next result due to H. Nagamochi *et al.* [35].

Theorem 9.2 *Let $G = (V, E)$ be a simple unweighted graph with a cost function $w : V \to \Re_+$. A minimum cost subset S such that $\hat{\kappa}_G(S, v) \geq k$ for all $v \in V - S$ can be computed in $O(\min\{k, \sqrt{n}\}kn^2)$ time.* ∎

We call a subset $S \subseteq V$ *k-feasible* if $\hat{\kappa}_G(S, v) \geq k$ for all $v \in V - S$. Recall that $\hat{\kappa}_G(S, v) \leq |S|$ holds. Hence any k-feasible set S satisfies $|S| \geq k$ and $\{v \in V \mid |\Gamma_G(v)| < k\} \subseteq S$. We call a subset $X \subseteq V$ *dominating* in G if $V - X - \Gamma_G(X) = \emptyset$, and *non-dominating* otherwise. A star augmentation H obtained from G is called *s-basally k-connected* if, for the designated vertex $s \in V(H)$, it holds

$$|\Gamma_G(X)| + d_H(s, X) \geq k \qquad \text{for all nonempty, non-dominating sets } X \subset V(G) \text{ in } G = H - s. \tag{9.1}$$

Lemma 9.1 *For an s-basally k-connected graph H, let S be a subset of $V(G)$ such that $\Gamma_H(s) \subseteq S$ and $|S| \geq k$. Then S is k-feasible in G.*

Proof: Assume indirectly that $\hat{\kappa}_G(S, v) < k$ for some $v \in V(G) - S$ in $G = H - s$. That is, there is a set $C \subseteq V(G)$ of at most $k - 1$ vertices such that $G - C$ has no path from any vertex $u \in S - C$ to v, where $S - C \neq \emptyset$ by $|S| \geq k$. Let X be the set of vertices in $V(G) - C$ that have paths to v in $G - C$. Note that X is not dominating in G since $V(G) - X - \Gamma_G(X) \supseteq S - C \neq \emptyset$. By $\Gamma_H(s) \subseteq S$, $X \cap \Gamma_H(s) \subseteq X \cap S = \emptyset$. This implies that $d_H(s, X) = 0$ and $|\Gamma_G(X)| \leq k - 1$ hold for such X, contradicting (9.1). ∎

We now show how to construct an s-basally k-connected graph H. It has been shown [35] that condition (9.1) of the s-basal connectivity is equivalent to a pair of the next conditions:

$$\begin{aligned}
&|\Gamma_G(x)| + c_H(s,x) \geq k && \text{for all singleton sets } X = \{x\} \subset V(G), && (9.2)\\
&|\Gamma_G(X)| + |\Gamma_H(s) \cap X| \geq k && \text{for all non-dominating sets } X \subset V(G) \text{ in } G && (9.3)\\
& && \text{with } |\Gamma_G(X)| + |X| \geq k. &&
\end{aligned}$$

Lemma 9.2 *Given a graph G and a k-feasible set S in G, let H be the graph obtained from G by adding a new vertex s together with* $\max\{1, k - |\Gamma_G(v)|\}$ *edges from s to each vertex $v \in S$. Then H is s-basally k-connected.*

Proof: Assume indirectly that H is not s-basally k-connected. Thus, $G = H - s$ has a nonempty, non-dominating set $X \subseteq V(G)$ such that $k - 1 \geq |\Gamma_G(X)| + d_H(s, X) \geq |\Gamma_G(X)| + |X \cap S|$. If $X \subseteq S$, then it is easy to see that X satisfies (9.2) or (9.3) by the construction of H. Assume that there is a vertex $u \in X - S$. Then G has a set P of k disjoint paths from S to u such that no two paths share a vertex in S. Since each of such paths must contain at least one vertex from $\Gamma_G(X) \cup (X \cap S)$, we have $|P| \leq |\Gamma_G(X)| + |X \cap S| \leq k - 1$, a contradiction. ∎

We observe that the problem enjoys a matroidal property. Given a graph G, we define $V_0 = \{v \in V(G) \mid |\Gamma_G(v)| < k\}$, and construct the graph H_0 obtained from G by adding a new vertex s and $\max\{1, k - |\Gamma_G(v)|\}$ edges from s to each vertex $v \in V(G)$. By Lemma 9.2 with $S = V(G)$, H_0 is s-basally k-connected.

By setting $U = V(G) - V_0$ as a ground set, we define a set system $\mathcal{M} = (U, \mathcal{I})$ by putting

$$\mathcal{I} = \left\{ X \subseteq U \;\middle|\; |X| \leq |V(G)| - k \text{ and } H_0 - E_{H_0}(s, X) \text{ remains } s\text{-basally } k\text{-connected} \right\}$$

(note that $|E_{H_0}(s, X)| = |X|$ since $|\Gamma_G(v)| \geq k$ for $v \in X \subseteq U$). For any $X \in \mathcal{I}$, $S := \Gamma_{H_0 - E_{H_0}(s,X)}(s) (= V(G) - X)$ is k-feasible in G by Lemma 9.1, since $|\Gamma_{H_0 - E_{H_0}(s,X)}(s)| \geq k$ holds and $H_0 - E_{H_0}(s, X)$ is s-basally k-connected. Conversely, for a given k-feasible set S in G, let $X := V(G) - S$. Then $X = V(G) - S \subseteq U$ (by $V_0 \subseteq S$) and $|X| \leq |V(G)| - k$ (by $|S| \geq k$). By Lemma 9.2, $H_0 - E_{H_0}(s, X)$ is s-basally k-connected, and hence $X \in \mathcal{I}$. It has been shown that the next holds.

Lemma 9.3 [35] *$\mathcal{M} = (U, \mathcal{I})$ is a matroid.* ∎

Since $\mathcal{M} = (U, \mathcal{I})$ is a matroid, we can obtain a subset $X \in \mathcal{I}$ with the maximum cost $w(X)$ by a greedy algorithm. Starting from $H = H_0$, we scan edges (s, v), $v \in V(G) - V_0$ in the nonincreasing order of cost $w(v)$, where if $H - (s, v)$ remains s-basally k-connected then we remove edge (s, v) from the current graph H and set $H := H - (s, v)$. We repeat scanning edges until we obtain an s-basally k-connected graph H' such that $E_{H'}(s, V(G))$ becomes minimal subject to the s-basal k-connectivity or $|\Gamma_{H'}(s)| = k$ holds. Clearly, a maximum cost subset $X \in \mathcal{I}$ corresponds to a k-feasible set S with $|S| \geq k$ with the minimum cost $w(S) = w(V(G)) - w(X)$. Therefore, $S = \Gamma_{H'}(s)$ in the resulting graph H' is a minimum cost k-feasible set.

By using a fast algorithm for computing the vertex-connectivity [17], the above algorithm can be implemented to run in $O(\min\{k, \sqrt{n}\}nm)$ time. This complexity can be reduced to $O(m + \min\{k, \sqrt{n}\}kn^2)$ by using Theorem 4.2(ii). For this, we execute the flow computation on a sparse spanning subgraph of H with $O(kn)$ edges that preserves the local vertex-connectivity up to k. This establishes Theorem 9.2.

Acknowledgement

This research was partially supported by the Scientific Grant-in-Aid from Ministry of Education, Culture, Sports, Science and Technology of Japan. The author would like to thank anonymous referees for their helpful comments.

References

[1] R. K. Ahuja, T. L. Magnanti and J. B. Orlin: *Network Flows: Theory, Algorithms, and Applications* (Prentice-Hall, Englewood Cliffs, NJ, 1993).

[2] K. Arata, S. Iwata, K. Makino and S. Fujishige: Locating sources to meet flow demands in undirected networks. *Journal of Algorithms*, **42** (2002), 54–68.

[3] S. R. Arikati and K. Mehlhorn: A correctness certificate for the Stoer-Wagner min-cut algorithm. *Information Processing Letters*, **70** (1999), 251–254.

[4] A. A. Benczúr and D. R. Karger: Augmenting undirected edge connectivity in $\tilde{O}(n^2)$ time. *Journal of Algorithms*, **37** (2000), 2–36.

[5] E. A. Dinits, A. V. Karzanov and M. V. Lomonosov: On the structure of a family of minimal weighted cuts in a graph. A. A. Fridman (eds.): *Studies in Discrete Optimization* (in Russian) (Nauka, Moscow, 1976), 290–306.

[6] P. Elias, A. Feinstein and C. F. Shannon: A note on the maximum flow through a network. *IRE Transaction on Information Theory*, **IT-2** (1956), 117–119.

[7] L. R. Ford and D. R. Fulkerson: Maximal flow through a network. *Canadian Journal of Mathematics*, **8** (1956), 399–404.

[8] A. Frank: Connectivity augmentation problems in network design. J. R. Birge and K. G. Murty (eds): *Mathematical Programming: State of the Art* (The University of Michigan, Ann Arbor, MI, 1994), 34–63.

[9] A. Frank: On the edge-connectivity algorithm of Nagamochi and Ibaraki. Laboratoire Artemis, IMAG, Université J. Fourier, Grenoble, March (1994).

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

[11] M. L. Fredman and R. E. Tarjan: Fibonacci heaps and their uses in improved network optimization algorithms, *Journal of the ACM*, **34** (1987), 596-615.

[12] S. Fujishige: Another simple proof of the validity of Nagamochi and Ibaraki's min-cut algorithm and Queyranne's extension to symmetric submodular function minimization. *Journal of the Operations Research Society of Japan*, **41** (1998), 626–628.

[13] H. N. Gabow: A matroid approach to finding edge connectivity and packing arborescences. *Proceedings of the 23rd ACM Symposium on Theory of Computing*, New Orleans, Louisiana (1991), 112–122.

[14] A. V. Goldberg and S. Rao: Flows in undirected unit capacity network. *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science* (1997), 32–35.

[15] A. V. Goldberg and R. E. Tarjan: A new approach to the maximum flow problem. *Journal of the ACM*, **35** (1988), 921–940.

[16] R. E. Gomory and T. C. Hu: Multi-terminal network flows. *SIAM Journal of Applied Mathematics*, **9** (1961), 551-570.

[17] M. R. Henzinger, S. Rao and H. N. Gabow: Computing vertex connectivity: New bounds from old techniques. *Journal of Algorithms*, **34** (2000), 222–250.

[18] S. Honami, H. Ito, H. Uehara and M. Yokoyama: An algorithm for finding a node-subset having high connectivity from other nodes (in Japanese). *Information Processing Society of Japan Special Interest Group Notes*, **AL-66-99-8** (1999), 9–16.

[19] T. Ishii, H. Fujita and H. Nagamochi: Source location problem with local 3-vertex-connectivity requirements. *Proceedings of the 3rd Hungarian-Japanese Symposium on Discrete Mathematics and Its Applications*, Tokyo (2003), 368–377.

[20] H. Ito, M. Ito, Y. Itatsu, K. Nakai, H. Uehara, and M. Yokoyama: Source location problems considering vertex-connectivity and edge-connectivity simultaneously. *Networks*, **40** (2002), 63–70.

[21] H. Ito, K. Makino, K. Arata, K. Itatsu, and S. Fujishige: Source location problem with edge-connectivity requirements in digraphs. *Proceedings of the 2nd Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications*, Hungary (2001), 92–97.

[22] H. Ito, H. Uehara and M. Yokoyama: A faster and flexible algorithm for a location problem on undirected flow networks. *The Institute of Electronics, Information and Communication Engineers Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E83-A** (2000), 704–712.

[23] H. Ito and M. Yokoyama: Edge connectivity between nodes and node-subsets. *Networks*, **31** (1998), 157–163.

[24] D. R. Karger and M. S. Levine: Finding maximum flows in undirected graphs seems easier than bipartite matching. *Proceedings of the 30th ACM Symposium on Theory of Computing* (1998), 69–78.

[25] A. V. Karzanov and E. A. Timofeev: Efficient algorithm for finding all minimal edge cuts of a nonoriented graph. *Kibernetika*, **2** (1984), 8–12; translated in Cybernetics (1986), 156–162.

[26] L. Lovász: *Combinatorial Problems and Exercises* (North-Holland, 1979).

[27] D. W. Matula: A linear time $2+\varepsilon$ approximation algorithm for edge connectivity. *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms* (1993), 500–504.

[28] H. Nagamochi: Computing extreme sets in graphs and its applications. *Proceedings of the 3rd Hungarian-Japanese Symposium on Discrete Mathematics and Its Applications*, Tokyo (2003), 349–357.

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

[30] H. Nagamochi and T. Ibaraki: Computing edge-connectivity of multigraphs and capacitated graphs. *SIAM Journal of Discrete Mathematics*, **5** (1992), 54–66.

[31] H. Nagamochi and T. Ibaraki: Deterministic $\tilde{O}(nm)$ time edge-splitting in undirected graphs. *Journal of Combinatorial Optimization*, **1** (1997), 5–46.

[32] H. Nagamochi and T. Ibaraki: Augmenting edge-connectivity over the entire range in $\tilde{O}(nm)$ time. *Journal of Algorithms*, **30** (1999), 253–301.

[33] H. Nagamochi and T. Ibaraki: Graph connectivity and its augmentation: applications of MA orderings. *Discrete Applied Mathematics*, **123** (2002), 447–472.

[34] H. Nagamochi, T. Ishii and T. Ibaraki: A simple and constructive proof of a minimum cut algorithm. *The Institute of Electronics, Information and Communication Engineers Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E82-A** (1999), 2231–2236.

[35] H. Nagamochi, T. Ishii and H. Ito: Minimum cost source location problem with vertex-connectivity requirements in digraphs. *Information Processing Letters*, **80** (2001), 287–294.

[36] H. Nagamochi and T. Kameda: Constructing cactus representation for all minimum cuts in an undirected network. *Journal of the Operations Research Society of Japan*, **39** (1996), 135–158.

[37] H. Nagamochi, S. Nakamura and T. Ibaraki: A simplified $\tilde{O}(nm)$ time edge-splitting algorithm in undirected graphs. *Algorithmica*, **26** (2000), 56–67.

[38] H. Nagamochi, S. Nakamura and T. Ishii: Constructing a cactus for minimum cuts of a graph in $O(mn + n^2 \log n)$ time and $O(m)$ space. *The Institute of Electronics, Information and Communication Engineers Transactions on Information and Systems*, **E86-D**, (2003), 179–185.

[39] H. Nagamochi, Y. Nakao and T. Ibaraki: A fast algorithm for cactus representations of minimum cuts. *Journal of Japan Society for Industrial and Applied Mathematics*, **17** (2000), 245–264.

[40] D. Naor, D. Gusfield and C. Martel: A fast algorithm for optimally increasing the edge connectivity. *SIAM Journal of Computing*, **26** (1997), 1139–1165.

[41] D. Naor and V. V. Vazirani: Representing and enumerating edge connectivity cuts in *RNC*. F. Dehne, J.-R. Sack and N. Santoro (eds.): Proceedings of the 2nd Workshop, WADS'91, *Lecture Notes in Computer Science*, **519**, Springer Verlag (1991), 273–285.

[42] J. C. Picard and M. Queyranne: On the structure of all minimum cuts in a network and applications, *Mathematical Programming Study.* **13** (1980), 8–16.

[43] S. Raghavan and T. L. Magnanti: Network Connectivity. M. Dell'Amico, F. Maffioli and S. Martello (eds.): *Annotated Bibliographies in Combinatorial Optimization* (John Wiley & Sons, 1997), 335-354.

[44] M. Stoer and F. Wagner: A simple min cut algorithm. *Journal of the ACM*, **44** (1997), 585–591.

[45] H. Tamura, M. Sengoku, S. Shinoda, and T. Abe: Location problems on undirected flow networks. *The Institute of Electronics, Information and Communication Engineers Transactions*, **E73** (1990), 1989–1993.

[46] H. Tamura, H. Sugawara, M. Sengoku and S. Shinoda: Plural cover problem on undirected flow networks (in Japanese). *The Institute of Electronics, Information and Communication Engineers Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **J81-A** (1998), 863–869.

[47] T. Watanabe and A. Nakamura: Edge-connectivity augmentation problems. *Journal of Computer and System Sciences*, **35** (1987), 96–144.

Hiroshi Nagamochi
Department of Applied Mathematics and Physics,
Kyoto University,
Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501,
Japan.
E-mail: `nag@amp.i.kyoto-u.ac.jp`