アルゴリズム 54－6
（1996. 10. 17）

無向グラフ上の辺分離問題を解く簡単な$\tilde{O}(mn)$時間アルゴリズム

永持 仁，　中村 伸也，　茨木 俊秀
京都大学 工学研究科 数理工学教室
〒606-01 京都市左京区吉田本町

摘要 偶数次数の指定された点 s を持つ k-辺連結無向多重グラフ $G=(V,E)$ が与えられたとき，s に接続する2辺 $(s,u),(s,v)$ を1辺 (u,v) に置き換える操作を反復し（グラフの k-辺連結性を保ちながら）s を孤立点化させる問題を考える。最近、$n=|V|$, m を G において辺の張られている点対の数としたとき、この問題を解く $O(n(m+n\log n)\log n)$ 時間のアルゴリズムが開発されたが [H. Nagamochi and T. Ibaraki, Deterministic $\tilde{O}(nm)$ time edge-splitting in undirected graphs, Proceedings 28th ACM Symposium on Theory of Computing, 1996, pp. 64-73]、このアルゴリズムはやや複雑である。本報告では、同じ $O(n(m+n\log n)\log n)$ の計算時間を持つより簡単なアルゴリズムを提案する。

A Simple $\tilde{O}(nm)$ Time Edge-Splitting Algorithm in Undirected Graphs

Hiroshi NAGAMOCHI, Shinya NAKAMURA and Toshihide IBARAKI

Department of Applied Mathematics and Physics
Graduate School of Engineering
Kyoto University
Kyoto, Japan 606-01

Abstract This paper presents a deterministic $O(n(m+n\log n)\log n)=\tilde{O}(nm)$ time algorithm for splitting off all edges incident to a vertex s of even degree in a multigraph G, where n is the number of vertices in G and and m is the number of vertex pairs between which G has an edge. The algorithm is much simpler than the previous $O(n(m+n\log n)\log n)$ time edge-splitting algorithm due to [H. Nagamochi and T. Ibaraki, Deterministic $\tilde{O}(nm)$ time edge-splitting in undirected graphs, Proceedings 28th ACM Symposium on Theory of Computing, 1996, pp. 64-73].

1 Introduction

Let $G = (V, E)$ stand for an undirected multigraph with a set V of *vertices* and a set E of *edges*, where an edge with end vertices u and v is denoted by (u, v). For two disjoint subsets $X, Y \subset V$, we denote by $E_G(X, Y)$ the set of edges, one of whose end vertices is in X and the other is in Y, and by $c_G(X, Y)$ the number of edges in $E_G(X, Y)$. In particular, $c_G(u, v) = |E_G(u, v)|$. Throughout the paper, the set of edges $E_G(u, v)$ may alternatively be represented by a single *link* (u, v) with multiplicity $c_G(u, v)$. In this way, we also represent a multigraph $G = (V, E)$ by an edge-weighted simple graph $N = (V, L_G, c_G)$ with a set V of vertices and a set L_G of links weighted by $c_G : L_G \to Z^+$, where Z^+ is the set of non-negative integers. We denote $n = |V|$, $e = |E|$ and $m = |L_G|$. A *cut* is defined as a subset X of V with $\emptyset \neq X \neq V$, and the *size* of cut X is defined by $c_G(X, V - X)$, which may also be written as $c_G(X)$. If $X = \{x\}$, $c_G(x)$ denotes the degree of vertex x. For a subset $X \subseteq V$, define its *inner-connectivity* by $\lambda_G(X) = \min\{c_G(X') \mid \emptyset \neq X' \subset X\}$. In particular, $\lambda_G(V)$ (i.e., the size of a minimum cut in G) is called the *edge-connectivity* of G.

Let $s \in V$ be a *designated vertex* in V. A cut X is called *s-proper* if $\emptyset \neq X \subset V - s$. $\lambda_G(V - s)$ (i.e., the size of a minimum s-proper cut) is called the *s-based-connectivity* of G. Hence $\lambda_G(V) = \min\{\lambda_G(V - s), c_G(s)\}$. Given neighbors u, v of s (possibly $u = v$) and a non-negative *integer* $\delta \leq \min\{c_G(s, u), c_G(s, v)\}$ in $G = (V, E)$, we denote by $G/(u, v; \delta)$ the graph obtained from G by *splitting* δ pairs of edges (s, u) and (s, v) (i.e., delete δ edges from $E_G(s, u)$ and $E_G(s, v)$, respectively, and add δ edges to $E_G(u, v)$). A set of splitting operations, each of which splits a pair of edges incident to s is called a *splitting at s*. A splitting is *feasible* (with respect to k) if $\lambda_{G'}(V - s) \geq k$ holds for the resulting graph G', and is *complete* if G' does not have any edge incident to s. Lovász [6] showed an important property.

Theorem 1 [6, 3] *Let $G = (V, E)$ be a multigraph with a designated vertex $s \in V$ with even $c_G(s)$, and k be an integer with $2 \leq k \leq \lambda_G(V - s)$. Then for each $u \in \Gamma_G(s)$ there is a vertex $v \in \Gamma_G(s)$ such that splitting one pair of edges (u, s) and (s, v) is feasible.* □

For a vertex s with even degree in a multigraph G and an integer k with $2 \leq k \leq \lambda_G(V - s)$, there always exists a pair of edges (s, u) and (s, v) whose splitting is feasible (with respect to k). By repeatedly applying this property, we see that, for such s and k, there always exists a complete feasible splitting. Since a complete feasible splitting reduces the number of vertices in a graph while preserving its s-based-connectivity, it is widely used as a powerful tool in inductive proofs of various edge-connectivity problems.

Recently, a deterministic $O(n(m + n \log n) \log n)$ time algorithm for finding a complete feasible splitting is proposed in [8]. Reference [8] first designed an $O(n(m + n \log n) \log n)$ time algorithm for finding a complete feasible splitting in an Eulerian graph, and then modified it for general graphs. Although the algorithm for Eulerian graphs is fairly simple, the algorithm modified for general graphs is considerably complicated.

This paper first shows that the same algorithm designed for Eulerian graphs in [8] (with a slight modification) works for finding a complete feasible splitting in a general graph with respect to an even integer k. Then based on this, we present a new simple $O(n(m+n \log n) \log n)$ time algorithm for general graphs in case of odd k.

2 Preliminaries

For a multigraph $G = (V, E)$, its vertex set V and edge set E may be denoted by $V[G]$ and $E[G]$, respectively. For a subset $X \subseteq V$ in G, $G[X]$ denotes the subgraph induced by X. For a vertex $v \in V$, a vertex $u \neq v$ adjacent to v by an edge is called a *neighbor* of v in G. Let $\Gamma_G(v) = \{w \in V \mid (v, w) \in E\}$ denote the set of *neighbors* of v in G. $\Gamma_G(v)$ always satisfies $|\Gamma_G(v)| = O(n)$. We say that a cut X *separates* two disjoint subsets Y and Y' of V if $Y \subseteq X$ and $Y' \subseteq V - X$ (or $Y \subseteq V - X$ and $Y' \subseteq X$) hold. In particular, a cut X separates x and y if $x \in X$ and $y \in V - X$ (or $y \in X$ and $x \in V - X$). We say that a cut X *divides* a subset Z of V if $Z \cap X \neq \emptyset \neq Z \cap (V - X)$ holds. A cut X *crosses* another cut Y if none of subsets $X \cap Y$, $X - Y$, $Y - X$ and $V - (X \cup Y)$ is empty.

The *local edge-connectivity* $\lambda_G(x, y)$ for two vertices $x, y \in V$ is defined to be the minimum size of a cut in G that separates x and y. An ordering $v_1, v_2, \ldots, v_n$ of all vertices in V is called *legal* in G if it satisfies $c_G(\{v_1, v_2, \ldots, v_i\}, v_{i+1}) \geq c_G(\{v_1, v_2, \ldots, v_i\}, v_j)$, $1 \leq i < j \leq n$.

Lemma 1 [7, 9, 11] *Let $G = (V, E)$ be a multigraph.*

(i) *A legal ordering $v_1, v_2, \ldots, v_n$ of vertices in G can be found in $O(m+n \log n)$ time (if a weight function $c_G : L_G \to R^+$ is given as input) or in $O(e+n)$ time (if a set E is given as input).*

(ii) *The following property holds for the last two vertices v_{n-1} and v_n, $\lambda_G(v_{n-1}, v_n) = c_G(v_n)$.* □

The next property is already observed in [8].

Lemma 2 [8] *Given a multigraph $G = (V, E)$ with a designated vertex $s \in V$ with even $c_G(s) > 0$, let $\Gamma_G(s) = \{x_1, x_2, \ldots, x_p\}$, and let $x_{max} \in \Gamma_G(s)$ satisfy $c_G(s, x_{max}) = \max\{c_G(s, x_i) \mid i = 1, 2, \ldots, p\}$. Then deleting $\max\{0, c_G(s, x_{max}) - \sum_{i \neq max} c_G(s, x_i)\}$ edges from $E_G(s, x_{max})$ never decreases the s-based-connectivity of G, and the result-*

ing graph G' has even $c_{G'}(s)$ and satisfies $p \geq 2$ and

$$\max_{1\leq i\leq p} c_{G'}(s,x_i) \leq \frac{1}{2}\sum_{1\leq i\leq p} c_{G'}(s,x_i). \qquad (1)$$

□

Throughout this paper, we assume without loss of generality that a given multigraph $G = (V,E)$ satisfies (1).

Finally we introduce the converse of an edge-splitting operation. Given a graph $G = (V,E)$, a designated vertex $s \in V$, two adjacent vertices $u, v \in V - s$ (possibly $u = v$) and a non-negative integer $\delta \leq c_G(u,v)$, we construct the graph $G^{\#} = (V, E^{\#})$ from G by deleting δ edges in $E_G(u,v)$, and adding new δ edges to $E_G(s,u)$ and $E_G(s,v)$, respectively. In this case, we say that $G^{\#}$ is obtained from G by *hooking up* δ edges (u,v) at s.

3 Splitting for Even k

We first consider how to find a compete feasible splitting with respect to an even integer k.

3.1 (k,s)-semi-critical collections and $\mathcal{X}$-astride splitting

For a multigraph $G = (V,E)$ and $s \in V$, a family $\mathcal{X} = \{X_1, X_2, \ldots, X_p\}$ of disjoint subsets $X_i \subset V - s$ is called a *collection* in $V - s$. A collection $\mathcal{X}$ may be empty. If $\sum_{i=1}^{p} c_G(s,X_i) = c_G(s)$ holds, then $\mathcal{X}$ is called *covering* (i.e., either every neighbor of s is contained in some subset $X_i \in \mathcal{X}$ if $\Gamma_G(s) \neq \emptyset$, or $\mathcal{X} = \emptyset$ if $\Gamma_G(s) = \emptyset$). An s-proper cut X is called *(k,s)-semi-critical* in G if it satisfies

$$c_G(s,X) > 0,\ k \leq c_G(X) \leq k+1 \ \text{ and } \ \lambda_G(X) \geq k.$$

A collection $\mathcal{X}$ in $V - s$ is called *(k,s)-semi-critical* in G either if $\mathcal{X} = \emptyset$ or if all $X_i \in \mathcal{X}$ are (k,s)-semi-critical. For a collection $\mathcal{X} = \{X_1, X_2, \ldots, X_p\}$ in G, splitting a pair of edges (s,u) and (s,v) for neighbors u, v of s is called *$\mathcal{X}$-astride* if there is no X_i with $u, v \in X_i$.

Lemma 3 *For a multigraph $G = (V,E)$, a designated vertex $s \in V$ with a positive even integer $c_G(s)$ and a positive integer $k \leq \lambda_G(V - s)$, let $\mathcal{X} = \{X_1, X_2, \ldots, X_p\}$ be a (k,s)-semi-critical covering collection in G. Then*

(i) *$p \geq 2$ and*

$$\max_{1\leq i\leq p} c_G(s,X_i) \leq \frac{1}{2}\sum_{1\leq i\leq p} c_G(s,X_i). \qquad (2)$$

$$c_G(s,X_1) = c_G(s,X_2) \text{ for } p = |\mathcal{X}| = 2.$$

(ii) *Any feasible splitting of a pair of edges (s,u) and (s,v) is $\mathcal{X}$-astride.*

(iii) *Any subset $X_i \in \mathcal{X}$ satisfies $\lambda_{G'}(X_i) = \lambda_G(X_i)(\geq k)$ and $c_{G'}(X_i) = c_G(X_i)(= k)$ after an $\mathcal{X}$-astride splitting.*

Proof: Omitted. □

Note that (ii) of this lemma says that a complete feasible splitting at s is a complete $\mathcal{X}$-astride splitting at s, though being $\mathcal{X}$-astride does not mean feasibility in general.

3.2 Finding a complete $\mathcal{X}$-astride splitting

Given a (k,s)-semi-critical covering collection $\mathcal{X}$ in a multigraph $G' = (V,E')$ with a designated vertex $s \in V$, it is known in [8] that a complete $\mathcal{X}$-astride splitting at s can be found by an $O(nm)$ time algorithm, called C-SPLIT. The description of algorithm C-SPLIT is omitted due to space limitation. The idea is that we basically repeat choosing a pair of edges $(s,u) \in E_{G'}(s,X_i)$ and $(s,v) \in E_{G'}(s,X_j)$ for distinct $X_i, X_j \in \mathcal{X}$ and split these edges, where a pair of edges is chosen so that condition (2) of Lemma 3 is maintained after splitting the pair of edges (since $|\mathcal{X}| \neq 1$ holds as long as (2) holds).

Lemma 4 [8] *Given a multigraph $G' = (V,E')$, a designated vertex $s \in V$ with even $c_{G'}(s)$, and a covering collection $\mathcal{X}$ which satisfies (1), a multigraph $\hat{G}$ obtained by a complete $\mathcal{X}$-astride splitting at s is found in $O(nm)$ time (if weight function $c_{G'}$ is given as input) or in $O(n(e+n))$ time (if set E' is given as input), where $e = |E'|$, and creates $O(|\Gamma_{G'}(s)|)$ new links.* □

3.3 Hooking up the edges generated by infeasible splittings

Given a multigraph $G' = (V,E')$ with a designated vertex $s \in V$ and an positive integer $k \leq \lambda_{G'}(V - s)$, suppose that graph $\hat{G} = (V,\hat{E})$ is obtained from G' by a complete splitting at s, which is not necessarily feasible with respect to k. Let $B \subseteq \hat{E}$ be the set of edges created by the complete splitting. Then it is known in [8] that the following algorithm finds a minimal subset $B' \subseteq B$ such that hooking up the edges in B' recovers the s-based-connectivity of $\hat{G}$ up to k ("minimal" means that no proper subset $B'' \subset B'$ can do this).

Algorithm HOOK-UP

Input: A multigraph $\hat{G} = (V,\hat{E})$, a designated vertex $s \in V$ with $c_{\hat{G}}(s) = 0$, $B \subseteq \hat{E}$, and an positive integer k. (It is assumed that the graph G' obtained from $\hat{G}$ by hooking up all the edges in B satisfies $\lambda_{G'}(V - s) \geq k$.)

Output: A minimal subset $B' \subseteq B$ such that the multigraph $G^\# = (V, E^\#)$ obtained from $\hat{G}$ by hooking up the edges in B' satisfies $\lambda_{G^\#}(V-s) \geq k$, and a (k,s)-semi-critical covering collection $\mathcal{Y}$ in $G^\#$.

```
1 begin
2   H := G# := Ĝ; Y := ∅; B' := ∅;
3   while |V[H]| ≥ 4 do
4     Find two vertices v, w ∈ V[H] − s with
      λ_H(v, w) ≥ k;
5     Contract v and w into a single vertex x*
    and let H be the resulting graph;
6     if c_H(x*) < k then
7       Let X* ⊆ V − s be the set of all vertices
        contracted so far into x*;
8       Choose a set A of ⌈½(k − c_G#(X*))⌉
        edges arbitrarily from B ∩ E[G#[X*]];
9       B := B − A; B' := B' ∪ A;
10      Let G# denote the graph obtained by
        hooking up these edges in A at s in G#;
11      Let H denote the graph obtained by
        adding new 2⌈½(k − c_G#(X*))⌉ edges
        between s and x* in H;
12      Y := Y ∪ {X*}, after discarding from Y
        all X' ∈ Y such that X' ⊂ X*
13    end { if }
14  end; { while }
15  Output B' and Y
16 end. { HOOK-UP }
```

Lemma 5 [8] *Let $\hat{G} = (V, \hat{E})$, $s \in V$, $B \subseteq \hat{E}$ and k be the input of algorithm* HOOK-UP, *and let $n = |V|$, $m = |L_{\hat{G}}|$, $e = |\hat{E}|$. Then* HOOK-UP *runs in $O(n(m+n\log n))$ time (if weight function $c_{\hat{G}}$ is given as input) or in $O(n(e+n))$ time (if set $\hat{E}$ is given as input). Let B' and $\mathcal{Y}$ be the subset of B and the collection obtained by* HOOK-UP, *and let $G^\#$ denote the graph obtained from $\hat{G}$ by hooking up all edges in B'. Then:*

(i) $\lambda_{G^\#}(V-s) \geq k$.

(ii) *$\mathcal{Y}$ is a (k,s)-semi-critical covering collection in $G^\#$.*

(iii) *B' is minimal in the sense that, for any proper subset $B'' \subset B'$, graph G'' obtained from $\hat{G}$ by hooking up the edges in B'' satisfies $\lambda_{G''}(V-s) < k$.* □

3.4 Initial (k,s)-semi-critical covering collection

Given a graph $G = (V,E)$ with a designated vertex $s \in V$ and an even integer $k \leq \lambda_G(V-s)$, a (k,s)-semi-critical covering collection can be found as follows. As observed in Lemma 2, we can assume that G and s satisfy condition (1). Therefore, by letting $\mathcal{X} = \{\{x_1\}, \{x_2\}, \ldots, \{x_p\}\}$ for $\Gamma_G(s) = \{x_1, \ldots, x_p\}$, we can apply C-SPLIT to find an initial complete splitting at s, since such $\mathcal{X}$ satisfies condition (2). Let $\hat{G}$ and B denote the resulting graph and the set of edges created by these splittings, respectively. Clearly $\Gamma_{\hat{G}}(s) = \emptyset$ since $c_G(s)$ is even.

3.5 Entire algorithm for even k

We are now ready to describe the entire algorithm for finding a complete feasible splitting at s with respect to an even integer k in a multigraph.

Algorithm EVEN-SPLIT

Input: A multigraph $G = (V,E)$ with a designated vertex s, which satisfies (1), and a positive even integer $k \leq \lambda_G(V-s)$.

Output: A multigraph $G' = (V,E')$ obtained from G by a complete feasible splitting (with respect to k) at s.

```
1 begin
2   Let Ĝ = (V, Ê) be a graph obtained from G
    by an initial complete splitting, and B ⊆ Ê
    be the set of edges created by this complete
    splitting;
3   Apply HOOK-UP to Ĝ and B to obtain
    G# = (V, E#) that satisfies λ_G#(V − s) ≥ k,
    by hooking up some edges in B, and a (k, s)-
    semi-critical covering collection Y in G#;
4   G' := G#; X := Y;
5   while X ≠ ∅ do
6     Apply C-SPLIT to G' and X to obtain the
      graph Ĝ = (V, Ê) and the set B ⊆ E'
      of created edges, resulting from a complete
      X-astride splitting in G';
7     Apply HOOK-UP to Ĝ and B to obtain
      G# = (V, E#) that satisfies λ_G#(V − s) ≥ k,
      by hooking up some edges in B, and a (k, s)-
      semi-critical covering collection Y in the G#;
8     G' := G#; X := Y
9   end; { while }
10  Output G'
11 end. { EVEN-SPLIT }
```

Before proving the correctness of this algorithm, we introduce some more definitions. Two cuts X and Y are called *s-neighboring* in G if $X \cap Y$ contains a neighbor $u \in \Gamma_G(s)$ of s. Furthermore, two cuts X and Y which are s-neighboring each other are called *s-crossing* if they satisfy $X - Y \neq \emptyset$ and $Y - X \neq \emptyset$ (i.e., X and Y are crossing each other).

Lemma 6 *Let $\mathcal{X}, \mathcal{Y}$ be the two collections $\mathcal{X}$ in line 6 and $\mathcal{Y}$ obtained in line 7 in each iteration of the while-loop in EVEN-SPLIT.*

(i) *For each $Y \in \mathcal{Y}$, $\mathcal{X}$ contains at least two cuts $X, X' \in \mathcal{X}$ which are s-neighboring to Y.*

(ii) *Let two cuts $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ be s-crossing each other. Then $c_{G^\#}(X) = c_{G^\#}(Y) = k+1$, $c_{G^\#}(X-Y) = c_{G^\#}(Y-X) = k$, and $c_{G^\#}(X \cap Y, V-(X \cup Y)) = 1$.*

(iii) *Let $Y \in \mathcal{Y}$ s-cross each of two cuts $X_1, X_2 \in \mathcal{X}$. If k is an even integer, then at most one of X_1 and X_2 can s-cross some other cut in $\mathcal{Y} - \{Y\}$.*

Proof: (i) For each $Y \in \mathcal{Y}$, $\hat{G}[Y]$ contains an edge $(u, v) \in B$ that is hooked up in $G^{\#}$. Since any edge $(u, v) \in B$ is created by an $\mathcal{X}$-astride splitting of edges (s, u) and (s, v), there are two cuts $X, X' \in \mathcal{X}$ such that $u \in X$ and $v \in X'$. Since both u and v are neighbor of s in $G^{\#}$, each of X and X' is s-neighboring to Y.

(ii) Since X and Y are crossing each other, they satisfy

$$\begin{aligned} & c_{G^{\#}}(X) + c_{G^{\#}}(Y) \\ &= c_{G^{\#}}(X - Y) + c_{G^{\#}}(Y - X) \\ &\quad + 2c_{G^{\#}}(X \cap Y, V - (X \cup Y)). \end{aligned} \tag{3}$$

We see that

$$c_{G^{\#}}(X) \leq k + 1, \quad c_{G^{\#}}(Y) \leq k + 1, \tag{4}$$

$$c_{G^{\#}}(X - Y) \geq k, \quad c_{G^{\#}}(Y - X) \geq k, \tag{5}$$

$$c_{G^{\#}}(X \cap Y, V - (X \cup Y)) \geq 1 \tag{6}$$

hold, because $\mathcal{X}$ and $\mathcal{Y}$ is (k, s)-semi-critical covertting, $\lambda_{G^{\#}}(V) \geq k$ holds, and X and Y are s-crossing each other. From (3), inequalites (4), (5), and (6) hold only by equation.

(iii) Assuming that both X_1 and X_2 s-cross some other cuts $Y_1, Y_2 \in \mathcal{Y} - \{Y\}$, respectively (possibly $Y_1 = Y_2$). By applying Lemma 6 (ii) to X_1 and Y, we have $c_{G^{\#}}(Y - X_1) = c_{G^{\#}}(X_1 - Y) = k$. From this, we see $Y - X_1 - X_2 = \emptyset$, because otherwise (if $Y - X_1 - X_2 \neq \emptyset$) $Y - X_1$ would s-cross X_2 and $c_{G^{\#}}(Y - X_1) = k + 1$ would hold by applying Lemma 6 (ii) to $Y - X_1$ and X_2. From $Y - X_1 - X_2 = \emptyset$, $c_{G^{\#}}(Y \cap X_2) = c_{G^{\#}}(Y - X_1) = k$ holds. Similarly for $Y \cap X_1$, we obtain $c_{G^{\#}}(Y \cap X_1) = k$. By applying the above argument to three cuts X_1, Y and Y_1, where X_1 s-crosses each of Y and Y_1, we also see $X_1 - Y - Y_1 = \emptyset$ and $c_{G^{\#}}(Y_1 \cap X_1) = c_{G^{\#}}(X_1 - Y) = k$ holds.

Let $c_{G^{\#}}(X_1 \cap Y, X_2 \cap Y) = a$ and $c_{G^{\#}}(X_1 \cap Y, X \cap Y_1) = b$. Then

$$\begin{aligned} & c_{G^{\#}}(X_1 \cap Y) \\ &= c_{G^{\#}}(X_1 \cap Y, Y - X_1) + c_{G^{\#}}(X_1 \cap Y, X_1 - Y) \\ &\quad + c_{G^{\#}}(X_1 \cap Y, V - (X_1 \cup Y)) \\ &= a + b + 1 = k, \end{aligned} \tag{7}$$

where $c_{G^{\#}}(X_1 \cap Y, V - (X_1 \cup Y)) = 1$ from Lemma 6 (ii). From $X_1 - Y - Y_1 = \emptyset$, we have $c_{G^{\#}}(X_1) = a + 1 + 1 + c_{G^{\#}}(X_1 \cap Y_1, Y_1 - X_1) = k + 1$ by Lemma 6 (ii), from which $c_{G^{\#}}(X_1 \cap Y_1, Y_1 - X_1) = k - a - 1$. Then

$$c_{G^{\#}}(X_1 \cap Y_1) = (k - a - 1) + b + 1 = k \tag{8}$$

Therefore, $a = b$ from (8), and this implies $2a = k + 1$ from (7). This however contradicts the evenness of k, proving the lemma. □

The next lemma is easily derived from the previous lemma.

Lemma 7 *The two collections $\mathcal{X}$ in line 6 and $\mathcal{Y}$ obtained in line 7 in each iteration of the while-loop in* EVEN-SPLIT *satisfy $|\mathcal{Y}| \leq \frac{2}{3}|\mathcal{X}|$.*

Proof: By Lemma 6(i), each $Y \in \mathcal{Y}$ has two cuts in $\mathcal{X}$ which are s-neighboring to it. Let $X_Y, X'_Y \in \mathcal{X}$ denote such two cuts for a $Y \in \mathcal{Y}$. Since at most one of $X_Y, X'_Y \in \mathcal{X}$ can s-cross some other cut in $\mathcal{Y} - \{Y\}$ by Lemma 6(iii), we have $|\mathcal{X}| \geq \frac{3}{2}|\mathcal{Y}|$. □

By this lemma, EVEN-SPLIT terminates after at most $\lceil \log_{1.5} |\mathcal{X}_0| \rceil = O(\log |\Gamma_G(s)|)$ iterations of the while-loop, where $\mathcal{X}_0$ is the initial collection $\mathcal{X}$ in line 4. Since $\mathcal{X} = \emptyset$ in the last iteration of the while-loop means that $G^{\#} = \hat{G}$ holds in the same iteration, the output G' satisfies $\lambda_{G'}(V - s) \geq k$ and $c_{G'}(s) = 0$. This shows the correctness of EVEN-SPLIT.

Now let us consider the running time of EVEN-SPLIT. Assume that G is given by an edge weight function c_G for links, where $n = |V|$ and $m = |L_G|$. Since the number of new links (including self-loop type) created by an initial complete splitting in line 2 is $O(|\Gamma_G(s)|) = O(n)$, and lines 2-3 can be carried out in $O(n(m + n \log n))$ time by Lemma 5. In each iteration of the while-loop, $O(|\Gamma_G(s)|)$ new links are created by C-SPLIT in line 6 (see Lemma 4) and the while-loops are iterated $O(\log |\Gamma_G(s)|)$ times as discussed above, the number of links in each of these $\hat{G}$ and $G^{\#}$ constructed during the whole execution of EVEN-SPLIT is $O(m + |\Gamma_G(s)| \log |\Gamma_G(s)|)$. Hence each while-loop can be performed in $O(n'(m' + n' \log n'))$ time by Lemma 5, where $n' = n$ and $m' = m + |\Gamma_G(s)| \log |\Gamma_G(s)| = O(m + n \log |\Gamma_G(s)|)$. Therefore, the entire running time becomes $O(n(m + n \log n) \log |\Gamma_G(s)|)$.

When G is given by an edge set E, the number of edges in $\hat{G}$ or $G^{\#}$ is always at most $|E|$. Then by Lemma 5 and the above discussion on the number of iterations, the entire running time in this case becomes $O(n(e + n) \log |\Gamma_G(s)|)$. Consequently, we have the next theorem.

Theorem 2 *A complete feasible splitting in a multigraph $G = (V, E)$ with a designated vertex $s \in V$ and a positive even integer $k \leq \lambda_G(V - s)$ can be found in $O(n(m + n \log n) \log |\Gamma_G(s)|)$ time (if weight function c_G is given as input) or in $O(n(e + n) \log |\Gamma_G(s)|)$ time (if set E is given as input), where $n = |V|$, $e = |E|$ and $m = |L_G|$.* □

4 Splitting for Odd k

4.1 Sketch of Algorithm

We first outline our algorithm to find a complete feasible splitting at s for a given multigraph $G' = (V, E', c_{G'})$ with a designated vertex $s \in V$, and an odd positive integer $k \leq \lambda_{G'}(V - s)$. Let k' be $k - 1$. Our algorithm consists of the following two phases.

Phase 1: Let $k' = k-1$. We divide the set of edges incident to s $E_G(s)$ into two set $E_{k'}$ and $E_G(s) - E_{k'}$ such that $E_{k'}$ is minimal subject to the property that the resulting multigraph $G_{k'} = (V, (E - E_G(s)) \cup E_{k'})$ satisfies $\lambda_{G_{k'}}(V - s) \geq k'$. If $|E_{k'}|$ is odd, let $E_{k'} := E_{k'} \cup \{\tilde{e}\}$ by choosing an arbitrarily edge $\tilde{e}$ from $E_G(s) - E_{k'}$. Since $k'(= k-1)$ is even, apply EVEN-SPLIT to $G_{k'} = (V, (E - E_G(s)) \cup E_{k'})$, s and k' to obtain $G^* = (V - s, E_G(s) \cup B)$ from $G_{k'}$ by a complete feasible splitting with respect to k' at s, where B is the set of edges created by this splitting.

Phase 2: We split the edges in $E_G(s) - E_{k'}$ to find a complete feasible splitting with respect to k at s. For this purpose, we first construct a compact representation of all minimum cut in G^*. Based on a structural information from this representation, we find a feasible splitting with respect to k at s.

4.2 Finding a minimal set of edges E_k

Given a multigraph $G = (V, E)$ with a designated vertex $s \in V$ and an odd positive integer $k \leq \lambda_G(V - s)$, let $G_s = (V, E - E_G(s))$ be a graph obtained from G by deleting all edges in $E_G(s)$. We can easily modify algorithm HOOK-UP to obtain an algorithm, called MINIMAL that computes a minimal subset $E_k \subseteq E$ such that $\lambda_{G_k}(V - s) \geq k$, where G_k denotes the graph obtained from G_s by putting back the edges in E_k ("minimal" means that no proper subset $E' \subset E_k$ can do this). Although the description of algorithm MINIMAL is omitted due to space limitation, we can obtain the next.

Lemma 8 *Let $G = (V, E)$ be a multigraph with a designated vertex s, which satisfies (1), and a positive integer k with $k \leq \lambda_G(V - s)$, where $n = |V|$ and $m = |L_G|$. Then there is an $O(n(m + n \log n))$ time algorithm that computes a multigraph $G_k = (V, (E - E_G(s)) \cup E_k)$ such that $\lambda_{G_k}(V - s) \geq k$, $E_k \subseteq E_G(s)$, and E_k is minimal subject to these properties.* □

4.3 Structure in $G^*_{k'}$

Given a multigraph $G = (V, E)$ with a designated vertex $s \in V$ and an odd positive integer $k \leq \lambda_G(V - s)$, let $G_s = (V, E - E_G(s))$ be a graph obtained from G by deleting all edges in $E_G(s)$, and k' be $k-1$. We partition $E_G(s)$ into two sets E_A and E_B as follows. Let $E_{k'} \subseteq E$ be a minimal subset such that $\lambda_{G_{k'}}(V - s) \geq k'$, where $G_{k'}$ denotes the graph obtained from G_s by putting back the edges in $E_{k'}$. If $|E_{k'}|$ is even, then $E_A = E_{k'}$. If $|E_{k'}|$ is odd, then choose an edge e^* arbitrarily from $E_G(s) - E_{k'}$, and let E_A be $E_{k'} \cup \{e^*\}$ and $G_{k'} = (V, (E - E_G(s)) \cup E_A)$ be the graph from G_s obtained by putting back the edges in E_A. Then let $E_B = E_G(s) - E_A$, $G^*_{k'} = (V, E')$ denote the graph obtained from $G_{k'}$ by a complete feasible splitting with respect to k' at s, and $G_k = (V, E' \cup E_B)$ denote the graph obtained from $G^*_{k'}$ by putting back the edges in $E_G(s) - E_{k'}$. We find a feasible splitting of edges at s in G_k based on a structural information of a set of minimum cuts in $G^*_{k'}$. For this, we review a *cactus representation* [2], a compact representation of all minimum cuts in an undirected graph. An undirected graph is called a *cactus* if any two cycles (if any) have at most one common vertex. An node w in cactus graph is called a *leaf* if w has degree exactly two. Given an undirected graph $G = (V, E)$, we map it to an edge-weighted cactus $\mathcal{R} = (\mathcal{V}, \mathcal{E})$ by a mapping $\varphi : V \to W$. For notational convenience, we assume that cactus $\mathcal{R}$ in a cactus representation $(\mathcal{R}, \varphi)$ has no bridge. All the edges in $\mathcal{E}$ are weighted by $\lambda_G(V)/2$, i.e., $c_{\mathcal{R}}(e) = \lambda_G(V)/2$, for all $e \in \mathcal{E}$.

Let $\mathcal{C}(G) = \{Z \mid \emptyset \neq Z \subset V,\ d_G(Z) = \lambda_G(V)\}$ and $\mathcal{C}(\mathcal{R}) = \{S \mid \emptyset \neq S \subset \mathcal{V},\ d_{\mathcal{R}}(S) = \lambda_{\mathcal{R}}(\mathcal{V})\}$ (i.e., the sets of all minimum cuts of G and $\mathcal{R}$, respectively). In the use of mapping φ, we use the term "vertex" to denote an element in V, and the term "node" to denote an element in $\mathcal{V}$. Define

$$\varphi(Z) \equiv \{\varphi(v) \in \mathcal{V} \mid v \in Z\} \quad \text{for } Z \subseteq V, \text{ and}$$
$$\varphi^{-1}(S) \equiv \{v \in V \mid \varphi(v) \in S\} \quad \text{for } S \subseteq \mathcal{V}.$$

For $S \subseteq \mathcal{V}$, we use the notation $\tilde{S} = \mathcal{V} - S$. A pair $(\mathcal{R}, \varphi)$ of an edge-weighted cactus and a mapping φ is a cactus representation for $\mathcal{C}(G)$ if it satisfies (i) and (ii) below.

(i) For an arbitrary cut $S \in \mathcal{C}(\mathcal{R})$, $Z \in \mathcal{C}(G)$ holds, where $Z = \varphi^{-1}(S)$ (and hence $\bar{Z} = \varphi^{-1}(\tilde{S})$).

(ii) Conversely, for any cut $Z \in \mathcal{C}(G)$, there exists a cut $S \in \mathcal{C}(\mathcal{R})$ such that $S \supseteq \varphi(Z)$ and $\tilde{S} \supseteq \varphi(\bar{Z})$.

It is known in [2] that there always exists such a cactus representation with size $|\mathcal{V}| = O(|\mathcal{E}|) = O(|V|)$. When $(\mathcal{R}, \varphi)$ is a cactus representation for $\mathcal{C} \subseteq \mathcal{C}(G)$, we say that the cut $Z \in \mathcal{C}(G)$ and the cut $S \in \mathcal{C}(\mathcal{R})$ correspond to each other if $\varphi(Z) \subseteq S$ and $\varphi(\bar{Z}) \subseteq \tilde{S}$. A node $x \in \mathcal{V}$ with $\varphi^{-1}(x) = \emptyset$ is called an *empty node*. Note that if $\mathcal{R}$ has an empty node, a minimum cut in $\mathcal{C}(G)$ may correspond to more than one minimum cut in $\mathcal{C}(\mathcal{R})$, while any minimum cut in $\mathcal{C}(\mathcal{R})$ always corresponds to exactly one minimum cut in $\mathcal{C}(G)$. It is known that a cactus representation $(\mathcal{R}, \varphi)$ with size $|\mathcal{V}| = O(|\mathcal{E}|) = O(|V|)$ can be constructed in $O(nm \log(n^2/m))$ time [5] and $O(nm \log m)$ time [9].

The following lemma show the underlying structure $G^*_{k'}$.

Lemma 9 *Let $G^*_{k'}$ be the graph obtained from $G_{k'}$ by a complete feasible splitting with respect to k' at designated vertex s, B be a set of split edges in $G^*_{k'}$ obtained from $G_{k'}$ by a complete feasible splitting with respect to k' at s and $(\mathcal{R}, \varphi)$ be a cactus representation for minimum cuts of $G^*_{k'}$. Then, for all edges (v, v') in B, if v is mapped to the leaf in $\mathcal{R}$, v' is mapped to other vertex in $\mathcal{R}$ by φ, i.e., $\varphi(v) \neq \varphi(v')$.*

Proof: For an edge $e' = (v, v') \in B$ assume v is mapped to a leaf w in $\mathcal{R}$. From the definition of leaf, $\lambda_{G^*_{k'}}(\varphi^{-1}(w)) = k' + 1$. Then, even if the edge (v, v') is deleted from $G^*_{k'}$, the connectivity of the resulting graph $G^*_{k'-e'}$ does not decrease, i.e., $\lambda_{G^*_{k'-e'}}(V - s) \geq k'$. From this, the s-based-connectivity of the graph $G_{k'-\{(s,v),(s,v')\}}$, which is obtained from $G_{k'}$ by deleting the edges (s, v) and (s, v'), is still at least k'. This however contradicts the minimality of $E_{k'}$ obtained after line 2 in ODD-SPLIT, because at most one edge $\tilde{e}$ is added to $E_{k'}$ in line 4 in ODD-SPLIT. □

Lemma 10 *Let $G^*_{k'}$ be the graph obtained from $G_{k'}$ by a complete feasible splitting with respect to k' at designated vertex s and $(\mathcal{R}, \varphi)$ be a cactus representation for minimum cuts of $G^*_{k'}$. For each leaf u in $\mathcal{R}$, there exists at least one edge $(v, s) \in E_G(s) - E_A$ where v is mapped to u by φ.*

Proof: Let G, $G_{k'}$ and $G^*_{k'}$ be respectively the input graph, the graph obtained respectively in line 7 and line 6. From lemma 9, $c_{G^*_{k'}}(U) = c_{G_{k'}}(U) = k'$ hold for each leaf u in $\mathcal{R}$ and $U = \varphi^{-1}(u)$. And $c_G(U) = k = k' + 1$ since $\lambda_G(V - s) \geq k$ hold in G. Then, for any leaf u in $\mathcal{R}$, there must exist at least one edge $(v, s) \in E_G(s) - E_A$ for some $v \in \varphi^{-1}(u)$. □

4.4 Entire algorithm

Before describing the entire algorithm, we define a depth-first-search (DFS) traversal in a cactus which is employed in [10] to solve the *Edge Augmentation Problem*, which asks to find a minimum set of edges to add to a graph so that its edge-connectivity increase by one. To explain easily, we assume different simple cycles in a cactus are differently colored, so that all cycle-edges in the same cycle are assigned the same color. Since every cycle-edges is contained in a unique cycle, such coloring is possible. Our special DFS traversal starts at an arbitrary node and obeys the following rule: if a node w is visited for the first time via a cycle-edge that is colored by, say, red, then traverse all other edges incident to w *before* traversing the other red edge incident to w.

We are now ready to describe the entire algorithm for finding a complete feasible splitting at s with respect to an odd edge-connectivity in an arbitrarily multigraph.

Algorithm ODD-SPLIT
Input: A multigraph $G = (V, E)$ with a designated vertex s with even degree, which satisfies (1), and a positive odd integer k with $k \leq \lambda_G(V - s)$.
Output: A multigraph $G' = (V, E')$ obtained from G by a complete feasible splitting (with respect to k) at s.

```
1  begin
2    k' := k − 1
3    Apply MINIMAL to G = (V, E), s and k' to
       obtain G_{k'} = (V, (E − E_G(s)) ∪ E_{k'})
       such that λ_{G_{k'}}(V − s) ≥ k', E_{k'} ⊆ E_G(s), and
       E_{k'} is minimal subject to these properties;
4    if |E_{k'}| is odd then
5      Choose an edge e* arbitrarily from
         E_G(s) − E_{k'};
6      E_{k'} := E_{k'} ∪ {e*}
7    else E_{k'} := E_{k'};
8    end {if}
9    Let G_s = (V, E − E_G(s)) be a graph obtained
       from G by deleting all edges in E_G(s);
10   Let G_{k'} denote the graph from G_s obtained
       by putting back the edges in E_A;
11   Apply EVEN-SPLIT to G_{k'}, s and k' to
       obtain G*_{k'} = (V − s, (E − E_G(s)) ∪ B)
       from G_{k'} by a complete feasible splitting with
       respect to k' at s, where B is the set of
       edges created by this splitting;
12   Construct a cactus representation (R, φ) for
       minimum cuts of G*_{k'};
13   Let {u_1, u_2, ···, u_l} be the set of leaves in R;
14   L := ∅;
15   for each leaf u_i of R do
16     Choose an edge e arbitrarily from
         E_G(s, U_i) − E_A, where U_i = φ^{-1}(u_i);
17     L := L ∪ {e}
18   end; {for}
19   if l is odd then
20     Choose an edge ẽ arbitrarily from
         E_G(s) − K − L;
21     L := L ∪ {ẽ}
22   end; {if}
23   Traverse R in a DSF manner, and let
       e_1 = (s, v_1), ···, e_{|L|} = (s, v_{|L|}) be the edges
       in L, where v_1, ···, v_{|L|} appear in this order
       along the DFS traversal (where v_i = v_{i−1} or
       v_i = v_{i+1} is possible for ẽ = e_i = (s, v_i));
24   Form the pairs {(e_i, e_{i+|L|/2}) | 1≤i≤|L|/2}
       and split all edges in L according to this pairing;
25   Let L' denote the set of edges created by this
       splitting, and G' denote the graph obtained
       from G* by adding the set of edges in L';
26   Split all edges in E_G(s, U) − E_A − L arbitrarily;
27   Let G' denote the graph obtained from G' by
       adding the set of edges created this splitting;
28   Output G'
29 end. { ODD-SPLIT }
```

First, we show the the correctness of Algorithm ODD-SPLIT. From Lemma 10, the edge set L in line 16 in ODD-SPLIT is possible to be chosen. Then we only to have to show that adding the set of edges L' created by the splitting executed in line 25 to the graph G^* obtained after line 11 increases the edge-connectivity of the resulting graph by one. To prove this, we need next lemmas.

Lemma 11 *Let L be a set of edges in G obtained after line 17 in ODD-SPLIT and be assigned numbers as in ODD-SPLIT. Then, for any minimum cut X in G, all edges in L the end nodes of which are mapped to a vertex in X have consecutive numbers (assuming that the last number is followed by the first number).*

Proof: Suppose that a cut X is obtained by the removal of two edges (from the same cycle), each of weight $k'/2$. Let $w_1, \cdots, w_p$ be the vertices in this cycle, and suppose, without loss of generality, that w_1 is the first vertex of this cycle to be visited in the DFS traversal. Then $w_1, \cdots, w_p$ must be visited exactly in this order since any w_i can be reached from w_1 only via the cycle. Also, our DFS traversal rule implies that for every w_i, the subgraph that is attached to w_i by edges not from this cycle is traversed before w_{i+1}. Hence, since a cut X separates the cycle into two consecutive parts, the edges the end nodes of which are mapped to a vertex in X must have consecutive members. □

Next lemma ensures the feasibility with respect to k of $G' = (V, E')$ obtained from G by a complete splitting at s by Algorithm ODD-SPLIT.

Lemma 12 *Let $L = \{(s, v_1), \cdots, (s, v_{|L|})\}$ be a set of edges in G obtained after line 17 in ODD-SPLIT and be assigned numbers as in ODD-SPLIT and $L' = \{e_1 = (v_1, v_{1+|L|/2}), \cdots, e_{|L|} = (v_{|L|/2}, v_{|L|})\}$ be the set of edges created by splitting according to the pairs $\{(e_i, e_{i+|L|/2}) : 1 \leq i \leq |L|/2\}$. Then, the connectivity of the graph obtained from G^* by adding the set of edges in L' increases at least one from that of G^*.*

Proof: We show that at least one new edge connects Z and $W - Z$ for every minimum cut Z in $\mathcal{R}$ (and thus connects X and $V - X$ for every minimum cut X in G^*). Let X be a minimum cut in G^*. Without loss of generality, assume that $|\{(s, v_i) \in L \mid v_i \in X\}| \geq |\{(s, v_i) \in L \mid v_i \in V - X\}|$ and that the end node v_i of e_i is mapped to a vertex in X. Then, from Lemma 11, the end node of edge $e_{i+|L|/2}$ must be mapped to a vertex in $V - X$, thus the new edge created by splitting the pair of e_i and $e_{i+|L|/2}$ connects X and $V - X$. □

Next, we consider the running time of ODD-SPLIT. Assume that G is given by an edge weight function c_G for links, where $n = |V|$ and $m = |L_G|$. From Lemma 8, line 3 can be carried out in $O(n(m + n \log n))$ time. From Theorem 2, line 9 can be carried out in $O(n(m + n \log n) \log |\Gamma_G(s)|)$. Constructing a cactus representation $(\mathcal{R}, \varphi)$ for min-cuts of G^* in line 10 can be carried out in $O(nm' \log n^2/m')$ or $O(nm' \log n)$ time, where $m' + O(m + |\Gamma_G(s)| \log |\Gamma_G(s)|)$ is the number of links in the graph $G^*_{k'}$. Traversing $\mathcal{R}$ in a DFS manner in line 18 can be carried out in linear time of the size of $\mathcal{R}$. The other executions in ODD-SPLIT can be carried out in linear time of the size of G. Therefore, the entire running time becomes $O(nm \log n + n^2 \log n \log |\Gamma_G(s)|)$.

Theorem 3 *A complete feasible splitting in a multigraph $G = (V, E)$ with a designated vertex $s \in V$ and a positive odd integer $k \leq \lambda_G(V - s)$ can be found in $O(nm \log n + n^2 \log n \log |\Gamma_G(s)|)$ time (if weight function c_G is given as input) or in $O(n(e + n) \log |\Gamma_G(s)|)$ time (if set E is given as input), where $n = |V|$, $e = |E|$ and $m = |L_G|$.* □

References

[1] G.-R. Cai and Y.-G. Sun, *The minimum augmentation of any graph to k-edge-connected graph*, Networks, 19, 1989, pp. 151-172.

[2] E.A. Dinits, A.V. Karzanov and M.V. Lomonosov, *On the structure of a family of minimal weighted cuts in a graph*, Studies in Discrete Optimization (in Russian), A.A. Fridman (Ed.), Nauka, Moscow, 1976, pp. 290-360.

[3] A. Frank, *Augmenting graphs to meet edge-connectivity requirements*, SIAM J. Disc, Math., 5, 1992, pp. 25-53.

[4] A. Frank, T. Ibaraki and H. Nagamochi, *On sparse subgraphs preserving connectivity properties*, J. Graph Theory, 17, 1993, pp. 275-281.

[5] H.N. Gabow, *Applications of a poset representation to edge connectivity and graph rigidity*, Proc. 32nd FOCS, 1991, pp.812-821.

[6] L. Lovász, Combinatorial Problems and Exercises, North-Holland 1979.

[7] 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, pp 583-596.

[8] H. Nagamochi and T. Ibaraki, *Deterministic $\tilde{O}(nm)$ time edge-splitting in undirected graphs*, Proc. 28th STOC, 1996, pp. 64-73.

[9] H. Nagamochi, T. Ishii and T. Ibaraki, *A simple and constructive proof of a minimum cut algorithm*, Tech. Rep. #96001, Kyoto Univ., Dept. of Appl. Math. and Physics, 1996.

[10] D. Naor, D. Gusfield and C. Martel, *A fast algorithm for optimally increasing the edge connectivity*, Proc. 31st FOCS, 1990, pp. 698–707.

[11] M. Stoer and F. Wagner, *A simple min cut algorithm*, Lecture Notes in Computer Science 855, Springer-Verlag, 2nd ESA, 1994, pp. 141–147.

[12] T. Watanabe and A. Nakamura, *Edge-connectivity augmentation problems*, J. Comp. System Sci., 35, 1987, pp.96-144.