An $O(\log^2 k)$-Approximation Algorithm for the $k$-Vertex Connected Spanning Subgraph Problem

Jittat Fakcharoenphol  
Department of Computer Engineering  
Kasetsart University  
Bangkok, 10900, Thailand  
jittat@gmail.com

Bundit Laekhanukit  
Computer Engineering Division  
Kasetsart University, Kamphaeng Saen Campus  
Nakhonpathom, 73140, Thailand  
lbundit@gmail.com

ABSTRACT

We present an $O(\log n \log k)$-approximation algorithm for the problem of finding a $k$-vertex connected spanning subgraph of minimum cost, where $n$ is the number of vertices in the input graph, and $k$ is the connectivity requirement. Our algorithm works for both directed and undirected graphs. The best known approximation guarantees for these problems are $O(\ln k \cdot \min (\sqrt{k}, \frac{\ln k}{\ln \ln k}))$ by Kortsarz and Nutov, and $O(\ln k)$ in the case of undirected graphs where $n \geq 6k^2$ by Cheriyan, Vempala, and Vetta. Our algorithm is the first that has a polylogarithmic guarantee for all values of $k$.

Combining our algorithm with the algorithm of Kortsarz and Nutov in case of small $k$, e.g., $k < n/2$, we have an $O(\log^2 k)$-approximation algorithm.

As in previous work, we use the Frank-Tardos algorithm for finding $k$-outconnected subgraphs as a subroutine. However, with a structural lemma that we proved, we are able to show that we need only partial solutions returned by the Frank-Tardos algorithm; thus, we can avoid paying the whole cost of the optimal solution every time the algorithm is applied.

Categories and Subject Descriptors: F.2 [Analysis of Algorithms and Problem Complexity]: General

General Terms: Algorithms, Theory

Keywords: Graph connectivity, approximation algorithms

1. INTRODUCTION

Let $G = (V, E)$ denote the input graph with a nonnegative cost $c_e$ on each edge $e \in E$. A $k$-separator is a set $S \subseteq V$ such that $G \setminus S$ is disconnected and $|S| = k$. Graph $G$ is $k$-vertex connected or $k$-connected if it has at least $k + 1$ vertices and there is no $k$-separator in $G$, i.e., $G \setminus X$ is connected, for all $X \subseteq V$ with $|X| < k$. The connectivity of $G$, denoted by $\kappa(G)$, is the maximum integer $\ell$ such that $G$ is $\ell$-connected.

In the minimum-cost $k$-vertex connected subgraph problem, we are given a graph (or a digraph) $G = (V, E)$ with nonnegative cost $c_e$ on each edge, and want to find a $k$-vertex connected subgraph of $G$ with minimum cost. This problem is one of the important network design problems, and a lot of work has been done to study this problem, in general settings [19, 20, 14, 15, 2] and in more restricted settings [14, 12, 1].

This problem is NP-hard for undirected graph with $k = 2$ and for directed graph with $k = 1$. The best known approximation guarantees for this problems are $O(\ln k \cdot \min (\sqrt{k}, \frac{\ln k}{\ln \ln k}))$ by Kortsarz and Nutov [15] and $O(\ln k)$ in case of undirected graphs where $n \geq 6k^2$ by Cheriyan, Vempala, and Vetta [2]. For metric costs, the best approximation guarantees are $2 + (k - 1)/n$ for undirected graphs and $2 + k/n$ for directed graphs [14]. For uniform cost, a $(1 + 1/k)$-approximation algorithm has been devised [1] for both directed and undirected graphs. In case of Euclidean graphs, Czumaj and Linas [3] presented a PTAS algorithms which approximation ratio $1 + \epsilon$ the optimum. They give both randomize and derandomized version of the algorithms.

The general approach for this problem is to start with an empty subgraph and iteratively increase its connectivity. There are various techniques for increasing vertex connectivity. One related notion of connectivity is outconnectivity (to be defined later). The cost of the minimum-cost $k$-vertex outconnected subgraph is a lowerbound on the optimum value of this problem. Therefore, the Frank-Tardos algorithm [5] for finding $k$-vertex outconnected digraph has been used, starting with Khuller and Raghavachari [12], as a subroutine for finding sets of edges for connectivity augmentation.

The recent line of research starting from Cheriyan, Vempala, and Vetta [2] and Kortsarz and Nutov [15] tries to minimize the number of times the Frank-Tardos algorithm is applied, as that would give a better performance guarantee. In an important work, Cheriyan et. al. observe that if an undirected graph is not an $\ell$-critical graph, it is enough to compute the Frank-Tardos algorithm from $\ell$ roots to increase the connectivity. That leads to their $O(\ln k)$-approximation algorithm in the case that $n > 6k^2$, because of results on critical graphs of Mader [17] that states that all 3-critical graph has at most $6k^2$ vertices. Note that their algorithm computes the Frank-Tardos algorithm from 3 roots.

Kortsarz and Nutov [15] further explore this approach. They consider a set of $\ell$-fragments which are, intuitively and informally, sets of vertices whose neighboring set is $\ell$; these fragments are those that need to be covered in order to ensure that the connectivity of the graph is at least $\ell + 1$. With a set-cover type analysis, they obtain the bound of $O(\frac{\ln n}{\ln \ell})$ on the number of starting vertices for computing the Frank-Tardos algorithm that ensures that all relevant $\ell$-fragments are covered. From this bound, they obtain an $O(\frac{\ln n}{\ln \ell} \ln^2 k)$-approximation algorithm. They also obtain another primal-dual based $O(\sqrt{\ln k})$-approximation algorithm using an LP-relaxation from Ravi and Williamson [19, 20].
In this paper, we do not try to minimize the number of calls to the Frank-Tardos algorithm. But we make sure that the cost for each call is small. Our notion of progress is the number of cores, which is a set of minimal $\ell$-fragments. We try to decrease the number of cores in the subgraph with the Frank-Tardos algorithm. Instead of paying the cost of optimal solution, we show that we can use only partial solution from the Frank-Tardos algorithm; thus, paying only $O(1/n)$ of the usual cost\(^1\) when there are $t$ cores in the graph. With the standard analysis, we can increase the connectivity by paying at most $O(\ln n)$ times the standard lowerbound. This charging scheme is not new. As one of the referees points out to us, it has been used in Goemans, Goldberg, Plotkin, Shmoys, Tardos, and Williamson’s result for the edge connectivity case [8].

As a side product, our procedure can be used to approximate a minimum-cost augmenting set (a set of edges that after added increases the vertex connectivity of the graph) within a factor of $O(\ln n)$ of the optimal. Its proof also gives an upperbound of $O(1/\ell \ln n \cdot \text{opt}_\ell)$ on the cost of augmenting a $\ell$-connected subgraph to $(\ell + 1)$-connected subgraph.

Through out the paper, we let $n = |V|$ and $m = |E|$. We also assume that $G$ is $k$-vertex connected.

1.1 Organization

In Section 2, we give formal definitions and describe basic lemmas. Section 3 presents the algorithm and its analysis. The main ingredient, procedure $\text{PARTIALAUGMENT}$, is presented and analyzed in Section 4. The procedure uses a subroutine based on max-flow computation to find a certain union of fragments, which is described in Section 5.

1.2 Other related work

The problem of finding vertex-connectivity of graphs has been studied extensively. The fastest-known deterministic algorithm for computing vertex-connectivity is due to Gabow [6]. The algorithm runs in time $O((n + m\sqrt{k/2 + kn^{-1/4}}))m)$. A randomized algorithm of Henzinger, Rao, and Gabow [9] runs in time $O(mn)$ and finds the vertex connectivity with probability at least $1/2$.

While we focus with the vertex-connectivity, many researchers have worked on an edge version. The best known approximation ratio for the minimum-cost $k$-edge connected spanning subgraph problem is 2, due to Khuller and Vishkin [13]. In case when $k = 1$ the problem is a Steiner tree problem where Robins and Zelikovsky [21] present an $1.55$-approximation algorithm. For a uniform cost, Cherian and Thurimella [1] present a $1 + 1/k$ approximation algorithm for both directed and undirected version. Gabow, Goemans, Tardos and Williamson [7] extend the result to work in the case with multiple edges. We refer the reader to a comprehensive survey by Kortsarz and Nutov [16].

2. DEFINITIONS AND BASIC PROPERTIES

For a graph $G$, let $N_G(X)$ denote the neighbors of $X \subseteq V$. We say that $X$ is an $\ell$-fragment of $G$ if $|N_G(X)| = \ell$ and $V \setminus (N_G(X) \cup X) \neq \emptyset$. If $G$ is clear from the context, the subscript $G$ will be omitted.

Later on, we assume that $G$ is $\ell$-connected; thus, for all fragments, we mean $\ell$-fragments. A complementary fragment of $X$, denoted by $X^c$, is $V \setminus (N_G(X) \cup X)$. A fragment $X$ is called small or proper fragment if $|X| < |X|$. An $\ell$-core $C$ of $G$ is an inclusion-wise minimal small $\ell$-fragment. For $\ell$-connected graph $G$, we denote the set of all $\ell$-cores in $G$ by $S(G)$, and let $t(G)$ denote the size of $S(G)$.

For an $\ell$-core $C$, let $A_C$ be a union of all $\ell$-fragments that contains only one core $C$. For an $\ell$-connected graph $G$, let $\kappa(G) = \{A_C : C \in S(G)\}$.

We now list a few basic properties, most of which are proved in Jordan [11] and Kortsarz and Nutov [15].

**Proposition 1** ([11]). Let $X$ and $Y$ be intersecting $\ell$-fragments in an $\ell$-connected (directed or undirected) graph $G$ on $n$ vertices. If $n - |X \cup Y| \geq 1$ then $X \cap Y$ is an $\ell$-fragment, and if a strict inequality holds, then also $X \cup Y$ is an $\ell$-fragment. In particular, the intersection of two small $\ell$-fragments is also a small $\ell$-fragment.

This proposition implies that there is no two $\ell$-cores intersect; thus, the set of all $\ell$-cores is pairwise disjoint. By the definition of $\ell$-core together with Proposition 1, we have the following proposition.

**Proposition 2.** Any small $\ell$-fragment $F$ in an $\ell$-connected graph $G$ contains at least one $\ell$-core.

The following proposition, which is a consequence of the above propositions, was proved by Kortsarz and Nutov [15].

**Proposition 3.** ([15]). For an $\ell$-connected graph $G$, the sets in $\kappa(G)$ are pairwise disjoint.

3. THE ALGORITHM

In this section we describe and analyze the performance of our algorithm.

The algorithm starts with an empty subgraph $G_0$ of $G$. It then proceeds in $k$ rounds. In round $t$, starting from round 0, it augments $G_t$ resulting in $G_{t+1}$ whose vertex-connectivity is $\ell + 1$ with a procedure described in Subsection 4.

At the heart of our augmenting procedure lies the subroutine $\text{PARTIALAUGMENT}$. A set of edges $F$ is an augmenting set of $H$ iff $\kappa(H \cup F) > \kappa(H)$. We define a partially augmenting set to be a set of edges $A$ such that $\kappa(H \cup A) < \kappa(H)$. In other words, for an $\ell$-connected graph $H$, $A$ reduces the number of $\ell$-core in $H$ by at least one.

Let $\text{opt}_\ell = \text{opt}_\ell(G)$ be the cost of the optimal $k$-vertex connected subgraph of $G$. We prove in Section 4 the following lemma.

**Lemma 1.** Given an $\ell$-connected subgraph $H$ of $G$ with $t = \kappa(H)$ $\ell$-cores, $\text{PARTIALAUGMENT}$ finds a partially augmenting set for $H$ of cost at most

$$O\left(\frac{1}{t} \cdot \frac{1}{k - \ell}\right) \cdot \text{opt}_\ell.$$  

Using $\text{PARTIALAUGMENT}$, our augmenting procedure is straightforward; we repeatedly call $\text{PARTIALAUGMENT}$ until the resulting subgraph is $(\ell + 1)$-connected.

Given Lemma 1, we have the following theorem, which uses essentially the same idea as Theorem 3.6 in [8].

**Theorem 1.** There is an $O(\log n \log k)$-approximation algorithm for the minimum-cost $k$-vertex connected subgraph problem, which runs in polynomial time.
Proof. First, we show that for each round \( \ell \), the augmenting cost is at \( O(n^{k-\ell}) \cdot \text{opt}_{k} \). Let \( G_{\ell} = H_{0}, H_{1}, \ldots, H_{p} = G_{\ell+1} \) denote a sequence of subgraphs produced by \( \text{PartialAugment} \). From Lemma 1, the cost of augmenting \( H_{\ell} \) to \( H_{\ell+1} \) at most

\[
O\left(\frac{1}{n(H_{\ell-1})} \cdot \frac{1}{k-\ell}\right) \cdot \text{opt}_{k}.
\]

Summing up, we have the augmenting cost at round \( \ell \) to be at most

\[
O\left(\sum_{i=0}^{\ell-1} \left(\frac{1}{n} \cdot \frac{1}{k-\ell}\right) \cdot \right) \cdot \text{opt}_{k} \leq \left(\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{1}\right) \cdot O\left(\frac{1}{k-\ell}\right) \cdot \text{opt}_{k} = O(n \ln n) \cdot \text{opt}_{k}.
\]

since the number of \( \ell \)-cores decreases at each step, and the number of \( \ell \)-cores in \( H_{0} \) is at most \( n \).

Hence, the total cost is

\[
O\left(\sum_{\ell=0}^{k-1} \left(\frac{n}{k-\ell}\right) \cdot \text{opt}_{k}\right) = O(\log n \log k) \cdot \text{opt}_{k}.
\]

We note that this algorithm can be combined with the algorithm by Kortsarz and Nutov [15], when \( k = o(n) \), so that we have the final guarantee of \( O(\log^{2} k) \) for all values of \( n \) and \( k \).

Regarding the cost of an augmenting set, the proof above shows that one can augment an \( \ell \)-connected subgraph to be an \( (\ell + 1) \)-connected subgraph by adding edges of cost \( O(\frac{1}{\ell} \log n) \cdot \text{opt}_{k} \).

4. PARTIALLY AUGMENTING A GRAPH

In this section, we describe procedure \( \text{PartialAugment} \) that, given an \( \ell \)-connected subgraph \( H \) of \( G \), finds a partially augmenting set of small cost.

The procedure uses the fact that for each \( \ell \)-core \( C \) in \( H \), there exists a subroutine that finds a union \( A_{C} \) of all small fragments that contain only one core \( C \). To improve the presentation flow of this section, we describe the subroutine separately in Section 5.

Later in this section we describe the algorithm. Subsection 4.1 proves the correctness of \( \text{PartialAugment} \). We show its performance guarantee in Subsection 4.2.

The procedure \( \text{PartialAugment} \) finds, for each \( \ell \)-core \( C \), a partial augmenting set \( F_{C} \). It then returns the set \( F_{C} \), for \( C \in S(H) \), of minimum cost. To find \( F_{C} \), it uses the Frank-Tardos algorithm [5] to find subgraph which is \( \ell + 1 \)-outconnected from some root vertex \( r \) in \( C \).

A directed graph \( \hat{G} = (V, \hat{E}) \) is \( k \)-outconnected from root vertex \( r \) if it contains \( k \) internally disjoint paths from \( r \) to any other vertices; a graph is \( k \)-inconnected to root vertex \( r \) if its reverse graph is \( k \)-outconnected from root vertex \( r \). Since these two notions are the same in undirected graph, we will use only \( k \)-outconnectivity when considering undirected graphs.

The Frank-Tardos algorithm, given a directed graph \( H \), connectivity requirement \( k \), and a root vertex \( r \), outputs a \( k \)-outconnected subgraph rooted at \( r \) of minimum cost.

We first consider the undirected case. To find \( F_{C} \) for an \( \ell \)-core \( C \in S(H) \), the algorithm constructs a directed graph \( \hat{G}_{C} \) from \( G \) by having two opposite arcs for each edge in \( G \), and assigning zero cost to all arcs corresponding to edges in \( H \) and all edges which have no end points in \( A_{C} \) to zero.

4.1 Correctness of \( \text{PartialAugment} \)

In this section, we prove the correctness of \( \text{PartialAugment} \), i.e., we show that the set \( F_{C} \) that it returns is indeed a partial augmenting set. More generally, we show that any set \( F_{C} \), for any \( \ell \)-core \( C \), computed by \( \text{PartialAugment} \) is a partial augmenting set.

We focus on the undirected case. The proofs for directed case are similar.

The following lemma uses various facts we listed in Section 2.

**Lemma 2.** Let \( H \) be an \( \ell \)-connected graph with \( t \) \( \ell \)-cores. Let \( C \) be an \( \ell \)-core in \( H \). The number of \( \ell \)-cores in \( H \cup F_{C} \) is at most \( t - 1 \).

**Proof.** Let \( H' = H \cup F_{C} \). Let \( C = S(H) \) and \( C' = S(H') \).

Note that every small \( \ell \)-fragment \( X \) in \( H' \) is also a small \( \ell \)-fragment in \( H \), since \( |N_{H}(X)| \leq |N_{H'}(X)| \leq \ell \). This means that any \( \ell \)-core in \( H' \) is an \( \ell \)-fragment in \( H \).

For the sake of contradiction, assume that \( |C'| \geq t \).

We first show that \( |C'| \) cannot be greater than \( t \). Since any \( \ell \)-core in \( H' \) is an \( \ell \)-fragment in \( H \), and any \( \ell \)-fragment contains at least one \( \ell \)-core (from Proposition 2), we have that for any \( \ell \)-core \( X' \in C' \), \( X \) contains some \( \ell \)-core \( X \in C \).
By the Pigeon Hole Principle, if $|C'| > t$, there are $X', Y' \subset C'$ that contain the same core $X \in C$. By proposition 1 all $t$-core are non-intersecting; therefore, we have a contradiction, since $X' \cap Y' \supset X \neq \emptyset$.

Consider the case that $|C'| = t$. Using the previous argument, we have that exactly one core $D \subset C'$ must contain $C$ and no other $t$-cores in $C$.

Since the Frank-Tardos algorithm outputs a set $I_C$ such that $N_{I_C}(X) \geq \ell + 1$ for any set $X$ such that $|X| < n - \ell - 1$. Therefore, $N_{I_C}(D) \geq \ell + 1$.

Clearly, $D$ is contained in $A_C$. Therefore, every edge in $I_C$ with at least one end vertices in $D$ is in also $F_C$, i.e., $N_{I_C}(D) = N_{I_C}(D) \geq \ell + 1$. This contradicts the assumption that $D$ is an $t$-core in $H'$, and the lemma follows.

This lemma also implies that the number of cores in $H \cup I_C$ is at most $t - 1$ as well, since $I_C \supset F_C$.

### 4.2 The cost for PartialAUGMENT

Let $OPT_3$ denote the optimal solution to the $k$-vertex connected subgraph, and $opt_k$ denote its cost. We compare the cost of our solution to the cost of optimal solution of the following linear program for the minimum $k$-connected spanning subgraph that has been introduced in [4] and used in [2, 15].

\[
\begin{align*}
\min & \sum_{e \in E(T)} c_e x_e \\
\text{s.t.} & \sum_{e \in \delta(S,T)} x_e \geq k - (n - |S \cup T|) \quad \forall \emptyset \neq S, T \subset V, S \cap T = \emptyset \\
& x_e \geq 0 \quad \forall e \in E,
\end{align*}
\]

where $\delta(S, T) = \{ (u, v) \in E : u \in S, v \in T \}$. We call this linear program $LP(k)$, and let $Z$ denote its optimal cost. Since this is a relaxation, we have that $Z \leq opt_k$.

Our algorithm, given an $t$-connected subgraph $H = (V, E_h)$, constructs an instance $I(\ell + 1)$ of $LP(\ell + 1)$ by first assigning all zero cost to all edges in $H$. Given the optimal solution $OPT = (V, E_h)$, one can construct a solution $\tilde{x}$ of $I$ by assigning

\[
\tilde{x} = \begin{cases} 
1 & \text{if } e \in E_h, \\
1/(k - \ell) & \text{if } e \in E_h \setminus E_h, \\
0 & \text{otherwise},
\end{cases}
\]

It is well-known (see, e.g., [2]) that $\tilde{x}$ is feasible, and has cost at most $opt_k/(k - \ell)$. We include the proof in the Appendix for completeness.

**Lemma 3.** There is a feasible solution to $I(\ell + 1)$ with cost at most $\frac{1}{k - \ell} opt_k$.

Given a directed graph $\hat{G} = (V, \hat{E})$, the following linear program $\hat{LP}$ is the relaxation of the $k$-outconnected spanning subgraph program.

\[
\begin{align*}
\min & \sum_{a \in A} c_a x_a \\
\text{s.t.} & \sum_{a \in \delta(S,T)} x_a \geq k - (n - |S \cup T|) \quad \forall \emptyset \neq S, T \subset V, S \cap T = \emptyset, r \in S \\
& x_a \geq 0 \quad \forall a \in \hat{a},
\end{align*}
\]

This linear program has an integer optimal solution (see [5]), and the Frank-Tardos algorithm finds it in polynomial time. We refer to this linear program as $\hat{LP}(\hat{G}, k, r)$.

The linear programs $LP(k)$ and $\hat{LP}(\hat{G}, k, r)$ are related. As to be shown later in Lemma 4, a feasible solution $\tilde{x}$ to $LP(k)$ is also feasible to $\hat{LP}(\hat{G}, k, r)$. This allows us to bound the cost of our algorithm to $opt_k$.

Procedure PartialAUGMENT computes the partial augmentation from many roots. Recall that for each core $C$, the algorithm first constructs a graph $G_C$ by setting all cost of edges in $H$ and edges with no endpoints in $A_C$ to zero. It then compute $I_C$ using the Frank-Tardos algorithm. Let $F_C$ denote the set of augmenting edges computed by PartialAUGMENT.

We shall bound the cost of $F_C$ to the partial cost of $OPT_3$. Let $B_C$ be a set of edges whose at least one end is contained in $A_C$. Let $O_C$ denote the set of edges in $OPT_3 \cap B_C$, i.e., $O_C = \{(u, v) \in OPT_3 : \{u, v\} \cap A_C \neq \emptyset \}$.

We first consider the case for undirected graph.

**Lemma 4.** $x$ is feasible for $\hat{LP}(G_C, \ell + 1, v)$, where $v$ is any vertex in an $\ell$-core $C$; therefore, the cost of $F_C$ is at most

\[
2 \cdot \frac{1}{k - \ell} \sum_{e \in O_C} c_e.
\]

**Proof.** First, we show that $x$ is feasible for $\hat{LP}(G_C, \ell + 1, v)$. Let $S$ be any set $S \subseteq H$ such that $v \in S$ and $S^* \neq \emptyset$. Since $x$ is feasible to $LP(\ell + 1)$, we then have $\sum_{e \in \delta(S, T)} x_e \geq k - (n - |S \cup T|)$. Thus $x$ is also feasible to $\hat{LP}(G_C, \ell + 1, v)$.

Now, we will show that $F_C$ has cost as claimed. Let $u$ be a root vertex used in Step 3 for computing $I_C$. Since $x$ is feasible for $\hat{LP}(G_C, \ell + 1, u)$, and the Frank-Tardos algorithm finds the optimal solution for that linear program, we have that the cost of $I_C$ is at most the cost of $x$ in $\hat{G}_C$. Because we assign zero cost to all edges not in $B_C$ and we duplicate edges, the cost of $x$ is at most $2 \cdot \frac{1}{k - \ell} \sum_{e \in O_C} c_e$.

To see that the bound also applies to $F_C$, we note that the cost of all edges in $F_C$ remains the same as in graph $\hat{G}_C$.

An analogous lemma for directed graph can be proved in the same way. We omit it from this version. With Lemma 4, we have the following bound on the total cost of all partial augmenting sets.

**Corollary 1.** For undirected case

\[
\sum_{e \in B(H)} cost(F_C) \leq 4 \cdot \frac{1}{k - \ell} opt_k.
\]

For directed case,

\[
\sum_{e \in B(H)} cost(F_C) \leq 2 \cdot \frac{1}{k - \ell} opt_k.
\]

**Proof.** The corollary follows since all sets in $h(H)$ are pairwise disjoint as asserted in Proposition 3. Therefore, for the undirected case, an edge $e \in OPT_3$ can contribute at most twice in the right-hand-side sum, and at most once in the directed case.

With this corollary, the main lemma that guarantees the performance of PartialAUGMENT is proved.

**Lemma 1.** Given an $t$-connected subgraph $H$ of $G$ with $t = \ell(H)$ $t$-cores, PartialAUGMENT finds a partially augmenting set for $H$ of cost at most

\[
O\left(\frac{1}{\ell - \ell} \cdot opt_k\right)
\]
Proof. By Corollary 1, there must be at least one $\ell$-core in $H$ such that cost of $F_C$ is less than $\frac{1}{\ell} \cdot O(\frac{1}{\ell} \cdot opt_H)$. The lemma follows because PARTIAL_AUGMENT returns $F_C$ with minimum cost. \hfill \square

Note that if we replace the fractional solution $x$ with the minimum augmenting set, we can have the cost of the optimal augmenting set as the lowerbound instead of $\frac{1}{\ell} \cdot opt_H$. A proof similar to the main theorem shows that one can use PARTIAL_AUGMENT repeatedly to find an augmenting set of cost at most $O(\log n)$ times the optimal.

5. FINDING A UNION OF ALL SMALL $\ell$-FRAGMENTS CONTAINING ONLY ONE CORE C

In this section we describe a subroutine that, given an $\ell$-connected graph $H$ and an $\ell$-core $C$ of $H$, produces a set $A_C$ of all small $\ell$-fragments which contain core $C$ as a unique core.

The subroutine, instead of generating all small $\ell$-fragments which share the same $\ell$-core $C$, determines whether a given vertex $v \in V - C$ is in $A_C$ using a decision procedure described shortly.

Note that given graph $H = (V,E)$ and a pair of vertices $u$ and $v$, one can find the minimal subset $X$ containing $u$ such that $N(X)$ is the minimum vertex cut separating $u$ and $v$ using one max-flow computation (on the vertex capacitated network induced by $H$).

Therefore, given an $\ell$-connected graph $H = (V,E)$ and a vertex $v \in V$, we can find the inclusion-wise minimal $\ell$-fragment that contains $v$ by computing at most $n$ max-flow from $u$ to all other vertices. In [15], it is shown that this algorithm can be implemented to run in time $O(mn)$.

To determine if a given vertex $v$ is in $A_C$, one has to find an $\ell$-fragment $X$ that contains $C$ and $v$. Let $e$ be an arbitrary vertex in $C$. Note that if we add an edge $e_v$ from $e$ to $v$, the resulting graph would contain no $\ell$-fragments $X'$ which contains $C$ and $v \notin X \cup N(X')$, because $e_v$ would violate the fact that $N(X')$ is a separator. However, this does not rule out the possibility that $v$ itself lies in $N(X')$; therefore, the algorithm that finds the minimal $\ell$-fragment above might find $X'$ instead of the required $X$.

Our approach for finding $X$ is to modify the graph $H$ further to get $H'$ so that $v$ does not lie inside any $\ell$-separator separating a small $\ell$-fragment containing $C$. To do this, we first add an edge $e_v$ from $e$ to $v$ to $H'$, if there is no such edge already in $H$. Let's denote a set of neighbors of $v$ in $H$ by $N_H(v)$. We also connect every pair of vertices in $N_H(v)$ in $H'$.

The decision procedure can be described as follows. First, it finds all $\ell$-cores in $H$, denoted by $S(H)$. It then produces $H'$, and uses the max-flow computation to find the minimal $\ell$-fragment $X$ containing $e$. The procedure accepts $v$ if $X$ exists, contains no other cores in $S(H)$, and is small.

Lemma 5. The decision procedure accepts $v$ iff $v$ is in $A_C$.

Proof. ($\Rightarrow$) Note that an $\ell$-fragment in $H'$ is also an $\ell$-fragment in $H$. If the decision procedure accepts $v$, then there exists a small $\ell$-fragment $X$ in $H$ such that $X$ contains $v$ and also contains $C$ as a unique core. This is a certificate for the fact that $v$ is in $A_C$.

($\Leftarrow$) If $v$ is in $A_C$, then there exists a minimal small $\ell$-fragment $X$ in $H$ such that $X$ contains both $C$ and $v$ and contains no other $\ell$-cores distinct from $C$.

First, we show that $X$ remains a small $\ell$-fragment in $H'$. Clearly, both end vertices of $e_v$ are inside $X$ since $X$ contains both $v$ and $C \geq c$. Since $v$ is in $X$, each vertex $u \in N_H(v)$ is either in $N_H(X)$ or $X$. We have that any edges added to $H$ to form $H'$ do not cross an $\ell$-separator $N_H(X)$. Therefore, $N_H(X)$ remains an $\ell$-separator in $H'$; and $X$ is one of the candidates for the max-flow procedure.

Assume that the procedure finds another small fragment $Y \neq X$. Since $Y$ is minimal, we can assume that $Y \subset X$, for otherwise we can replace $Y$ with $Y \cap X$ (because of Proposition 1 and both $X$ and $Y$ are small). Because of edge $e_v$, only two cases are possible: either (i) $Y$ contains both $v$ or (ii) $v$ lies in $N_H(Y)$, where $N_H(Y)$ is a set of neighbors of $Y$ in $H'$. Case (i) is ruled out because of the minimality of $X$.

We now consider case (ii). Since $v$ lies in $N_H(Y)$ and there are edges connecting every pair of $N_H(v)$ in $H'$, $N_H(v)$ cannot lie in both $Y$ and $Y'$, because that would violate the fact that $N_H(Y)$ is a separator. (See Figure 2.)

We proceed to rule out the case that $N_H(v) \subseteq N_H(Y) \cup Y'$ and $N_H(v) \cap Y' \neq \emptyset$. Note that if $e \notin H$, by removing that edge in $H'$, we have that $v$ cannot be reached from any vertex in $Y$ and $N_H(Y) - \{v\}$ is also a separator of size $\ell - 1$, resulting in a contradiction. On the other hand, if $e \in H$, we have that $e \in N_H(v)$, and there exists an edge from $e \in Y$ to some vertices in $Y'$, also leading to a contradiction.

Thus, we have that $N_H(v) \subseteq Y \cup N_H(Y)$. Now consider the set $Y' = Y \cup \{v\}$. Clearly, $|N_H(Y')| = |N_H(Y) \cup \{v\}| = \ell - 1$. Since $Y$ is an $\ell$-fragment in $H'$, $Y' = Y''$ is non-empty; thus, $N_H(Y')$ is an $(\ell - 1)$-separator, contradicting the fact that $H$ is $\ell$-connected. Therefore, case (ii) is also impossible.

Since both cases (i) and (ii) are impossible, we have a contradiction, and the if-part follows. \hfill \square

6. ACKNOWLEDGEMENT

We thank anonymous referees and Zeav Nutov for useful comments and discussions.

APPENDIX

A. A PROOF OF LEMMA 3

In this section we present a proof of Lemma 3.

Proof of Lemma 3. First, we show that $x$ satisfies the constraints of $I(\ell + 1)$. Let $S$ and $T$ be arbitrary non-empty vertex sets such that $S$ and $T$ are disjoint. Consider the constraint,

$$\sum_{e \in E(S,T)} x_e \geq (\ell + 1) - (n - |S \cup T|)$$

Figure 2. $N_H(v)$ cannot cross $N_H(Y)$.
Note that $\delta_E(S, T) = \delta_E(S, T) \cup \delta_E(E \setminus S, T)$. For simplicity, let $q$ denote the number of edges in $\delta_E(S, T)$, and $r$ denote the number of edges in $\delta_E(E \setminus S, T)$. Since $H$ is $\ell$-connected, we have $r \geq \ell - (n - |S \cup T|)$. If $r > \ell - (n - |S \cup T|)$, then the constraint is satisfied. So let consider the case that $r = \ell - (n - |S \cup T|)$. Since $OPT_k$ is $k$-connected, $\delta_E(S, T) \geq k - (n - |S \cup T|)$. We have $q \geq k - (n - |S \cup T|) - r = k - \ell$. Recall that, $x_e = 1$ for every edge $e \in H$ and $x_e = 1/(k - \ell)$ for every edge $e \in E_A \setminus E_h$. We have

$$\sum_{e \in \delta_E(S, T)} x_e = r + q/(k - \ell) \geq (\ell + 1) - (n - |S \cup T|)$$

Thus, $x$ is a feasible solution for $R(\ell + 1)$.

We now consider the cost of $x$. Since cost of all edges in $H$ are zero, cost of $x$ is at most $\frac{1}{\ell} OPT_k$, and the lemma follows.

References