Approximating Multiroot 3-Outconnected Subgraphs

Zeev Nutov*

Abstract

Consider the following problem: given an undirected graph with nonnegative edge costs, and requirements k_u for every node u, find a minimum-cost subgraph that contains $\max\{k_u, k_v\}$ internally disjoint paths between every pair of nodes u, v. For $k = \max k_u \geq 2$ this problem is NP-hard. The best known algorithm for it has approximation ratio $2(k-1)$. For a general instance of the problem, for no value of $k \geq 2$ a better approximation algorithm was known.

We consider the case of small requirements $k_u \in \{1, 2, 3\}$; these may arise in applications, as in practical networks the connectivity requirements are usually rather small. For this case we give an algorithm with approximation ratio $\frac{10}{3}$. This improves the best previously-known approximation ratio 4. Our algorithm also implies an improvement for arbitrary k. In the case in which we have an initial graph which is 2-connected, our algorithm achieves approximation ratio 2.

Key-wodrs: approximation algorithm, rooted 3-outconnected subgraph.

1 Introduction

A basic problem in network design is to find a minimum cost subgraph of a given graph (network) that satisfies given connectivity requirements (see [7, 4] for surveys). A fundamental problem in this area is the *survivable network design problem*: find a cheapest spanning subgraph so that for every pair of nodes $\{u, v\}$, there are k_{uv} internally disjoint paths[1] between u and v, where k_{uv} is a nonnegative integer (requirement) associated with the pair $\{u, v\}$. No efficient approximation algorithm for this problem is known.

An *α-approximation algorithm* for a minimization problem is a polynomial time algorithm that produces a solution of value no more than α times the value of an optimal solution; α is called the *approximation ratio* of the algorithm. A particular important case of the survivable network design problem is the problem of finding a k-connected spanning subgraph of minimal cost, that is, the case of *uniform* requirements when $k_{uv} = k$ for every node pair $\{u, v\}$. Ravi and Williamson [13] presented a $2H(k)$-approximation algorithm, where

*nutov@oumail.openu.ac.il, Open University of Israel, Klauzner 16 Str., Ramat-Aviv, Israel. Supported in part by NSERC research grant OGP0138432.

[1]Unless stated otherwise, "connectivity", "disjoint paths", and "cut" mean node connectivity, internally disjoint paths, and node cut, respectively.

$H(k) = 1 + \frac{1}{2} + \cdots + \frac{1}{k}$. However, the proof of the approximation ratio in [13] contains an error. The algorithm of [13] has k iterations; at iteration i the algorithm finds an edge set F_i such that $G_i = (V, F_1 \cup \cdots \cup F_i)$ is i-connected. At the end, G_k is output. There is an example [14] showing that the edge set F_k found at the last iteration has cost at least $k/2$ times the value of an optimal solution. On the other hand, it is possible to get a k-approximation algorithm, see [8]. A $\lceil (k+1)/2 \rceil$-approximation algorithms are known for $k \leq 7$; see [9] for $k = 2$, [1] for $k = 2, 3$, [3] for $k = 4, 5$, and [8] for $k = 6, 7$.

Particular cases of the survivable network design problem, where pairwise node requirements are defined by single node requirements, arise naturally in network design. For example, Stoer [12] discusses problems where pairwise requirements are of the form $k_{uv} = \min\{k_u, k_v\}$, where the single node requirements k_u are given. In this paper we consider the case in which the pairwise requirements are of the form $k_{uv} = \max\{k_u, k_v\}$, that is, the following problem:

The multiroot problem: Given an undirected graph with nonnegative edge costs, and requirements k_u for every node u, find a minimum-cost subgraph that contains $\max\{k_u, k_v\}$ internally disjoint paths between every pair of nodes u, v.

A graph is said to be *k-outconnected from a node r* if it contains k internally disjoint paths between r and any other node; such node r is usually referred to as the *root*. It is easy to see that a subgraph is a feasible solution to the multiroot problem if and only if it is k_u-outconnected from every node u. A graph is said to be *$(k_1, k_2, \ldots, k_q)$-outconnected from* $(r_1, r_2, \ldots, r_q)$ if it is simultaneously k_i-outconnected from each r_i, $i = 1, \ldots, q$. Given an instance of the multiroot problem, we use q to denote the number of nodes with positive single node requirements k_u, and $k = \max k_u$ is the maximum requirement. In this notation, the multiroot problem can be formulated as follows: given an undirected graph with nonnegative costs on the edges, a vector $\vec{R} = (r_1, \ldots, r_q)$ of q root nodes, and a vector $\vec{K} = (k_1, \ldots, k_q)$ of positive connectivity requirements, find a minimum-cost subgraph which is $\vec{K}$-outconnected from $\vec{R}$. We shall always assume without loss of generality that $k_1 \geq k_2 \geq \ldots \geq k_q$ (thus $k = k_1$). Observe that the minimum-cost k-connected subgraph problem is a special case of the multiroot problem when the number of the roots with requirement k is at least k (since any root with requirement k must be contained in every node cut smaller than k).

The one root problem (i.e., when $q = 1$) was considered long ago. Note that it is NP-hard, even when $k = 2$ and unit costs. To see this, note that a 2-outconnected spanning subgraph of a graph G has $\leq |V(G)|$ edges if and only if G has a Hamiltonian cycle. The general setting with more than one root was introduced in [11]. A possible application is described in [11]. We remark here that some other possible applications can fit this paradigm. For example, the root nodes ("suppliers") can be power stations, computer servers, transportation centers, etc., with possibly distinct abilities, and the goal is to achieve a certain level of supply reliability to the terminals ("demanders") according to the ability of the suppliers.

Let us say that a directed graph D is *$r \to k$-outconnected*, if in D there are k internally disjoint paths *from* r to any other node. For *directed* graphs, Frank and Tardos [5] showed that the problem of finding an optimal $r \to k$-outconnected subdigraph is solvable in polynomial time; a faster algorithm is due to Gabow [6]. This implies a 2-approximation algorithm

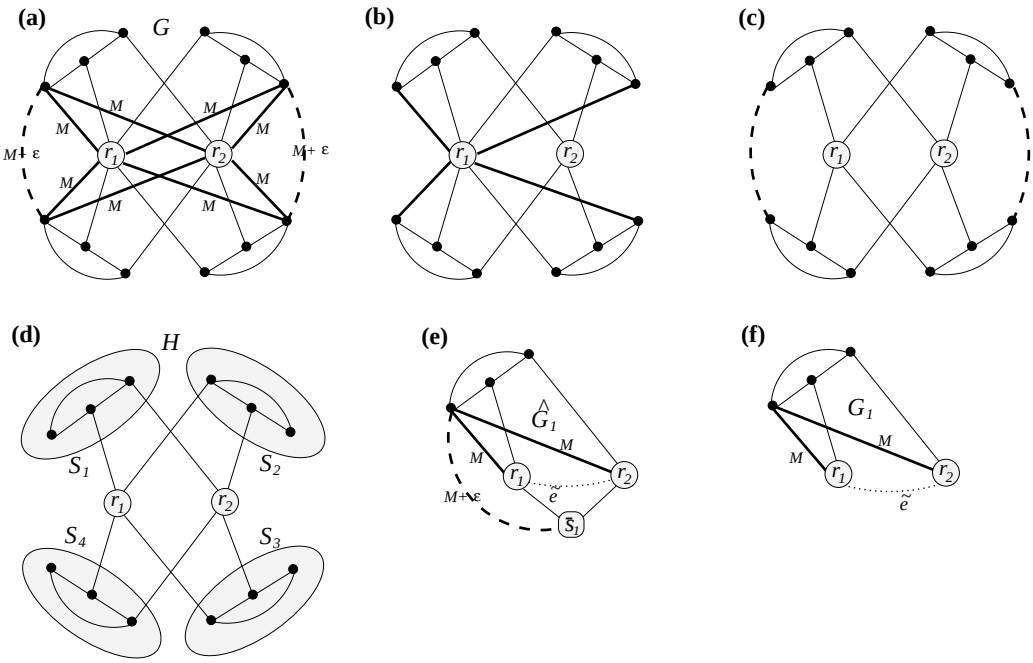

Figure 1: (a) A graph G. The costs are: thick edges: M, dashed edges: $M + \epsilon$, and all the other edges: 0; (b) a 3-outconnected from r_1 subgraph; (c) an optimal $(3, 3)$-outconnected from (r_1, r_2) subgraph; (d) an optimal 2-connected subgraph H of G and the sides of $\{r_1, r_2\}$; (e) $\{r_1, r_2\}$-component $\hat{G}_1$ with respect to H (the virtual edge is shown by dotted line); (f) split $\{r_1, r_2\}$-component G_1 with respect to H.

for the (undirected) one root problem, as follows. The *directed version* of a graph G is the digraph $D(G)$ obtained from G by replacing every undirected edge of G by the two anti-parallel directed edges with the same ends and of the same cost. To get a 2-approximation algorithm for the one root case, we compute in the directed version of the input graph an optimal $r \to k$-outconnected subdigraph, and output its underlying (undirected) simple graph. It is easy to see that the output subgraph is k-outconnected from r, and its cost is at most twice the value of an optimal k-outconnected (from r) subgraph, see [9].

For the multiroot problem, a $2q$-approximation algorithm follows by applying the above algorithm for each root and taking the union of the resulting q subgraphs. In [2], it was shown that we can always assume that $q \leq k$ as otherwise some requirements are redundant (so they can be set to zero), and if $q = k$ and there are no redundant requirements, then the problem is equivalent to that of finding a minimum-cost k-connected spanning subgraph. The approximation guarantee $2q$ of the above algorithm is tight for $k = 3$ and $q = k - 1$. To see this, consider the graph shown in Fig. 1(a), with two roots r_1, r_2 and $(k_1, k_2) = (3, 3)$; all the thick edges (dashed edges) have cost M ($M + \epsilon$) and all the other edges have cost 0. In the first step, the algorithm will find the subgraph of cost $4M$ shown in (b). Indeed, in the directed version of G, an optimal $r_1 \to 3$-outconnected subdigraph is the one obtained from the directed version of the graph in (b) by deleting the arcs entering r_1. Note that the subgraph in (b) is *not* 3-outconnected from r_2, since $G \setminus r_1$ is not 2-outconnected from r_2. At the second step, the algorithm will find a similar subgraph for r_2 (even if before the second

step the cost of all the edges found at the first step will be reduced to 0). The output graph will be the union of these two, and will have cost $8M$, while an optimal solution shown in (c) has cost $2M + 2\epsilon$. This example can be generalized for arbitrary $k \geq 3$, see [2].

In [2] are given a $\min\{2, \frac{k+2q-3}{k}\}$-approximation algorithm for unit costs, and a 3-approximation algorithm for metric costs. (*Remark:* The algorithm of [2] for metric costs computes a *k-connected* subgraph. However, for arbitrary costs, the input graph may contain a feasible solution to the problem, but not be k-connected, see Fig 1(a)). For metric costs, a $(2 + \frac{\lfloor (k-1)/2 \rfloor}{k})$-approximation algorithm for $k \leq 7$ is given in [8].

But, in the case of arbitrary costs, for no value of k an algorithm with a better approximation ratio than $2(k-1)$ is known. In this paper we give an algorithm for $k = 3$ with approximation ratio $\frac{10}{3}$.

This paper is organized as follows. Section 2 contains some preliminary results and definitions. In section 3 we give a 2-approximation algorithm for augmenting a 2-connected graph to be $(3,3)$-outconnected, and in Section 4 we show a $\frac{10}{3}$-approximation algorithm for the multiroot problem with $k = 3$.

A preliminary version of this paper is [10].

2 Definitions and preliminary results

Here are some notations and definitions used in the paper. Unless stated otherwise, the graphs in the paper are assumed to be connected, and simple (i.e., without loops and parallel edges); if parallel edges arise, only the cheapest is kept. For an arbitrary graph G, $V(G)$ denotes the node set of G, and $E(G)$ denotes the edge set of G. Let $G = (V, E)$ be a graph. For any set of edges and nodes $U = E' \cup V'$, we denote by $G \setminus U$ (respectively, $G \cup U$) the graph obtained from G by deleting U (respectively, by adding U), where deletion of a node implies also deletion of all the edges incident to it. For a nonnegative cost function c on the edges of G and a subgraph $G' = (V', E')$ of G we use the notation $c(G') = c(E') = \sum\{c(e) : e \in E'\}$. Similar notations are used for digraphs.

A subset $C \subset E \cup V$ is called a *separator* if $G \setminus C$ is disconnected. A particular case of a separator that contains nodes only is referred to as (*node*) *cut*. If $|C| = k$ then C is called a *k-separator* (*k-cut*). A *side* of a separator C is the node set of a connected component of $G \setminus C$. A graph G is *k-connected* if it has no l-separator with $l < k$. A separator C separates two nodes u and v if u and v belong to distinct sides of C. For $r \in V$ and $S \subseteq V \setminus r$ we say that G *is k-outconnected from r to S* if for every $v \in S$ there are k internally disjoint paths between r and v in G. Note that a graph is k-outconnected from r to S if and only if it has no l-separator with $l < k$ separating r and some $v \in S$ (by Menger's Theorem).

Throughout the paper, for an instance of a problem, we will denote by $\mathcal{G}$ the input graph; $n = |V(\mathcal{G})|$ denotes the number of nodes in $\mathcal{G}$, and $m = |E(\mathcal{G})|$ the number of edges in $\mathcal{G}$.

Note that to prove a $\frac{10}{3}$-approximation algorithm for the multiroot problem with $k = 3$ it is sufficient to consider the case $q = 2$ and $(k_1, k_2) = (3, 3)$. Indeed, for $q = 1$ (and k arbitrary), there is a 2-approximation algorithm. For $q = 3$, if all three requirements are non-redundant, then, by [2], the problem is equivalent to that of finding a minimum-cost

3-connected spanning subgraph; for the latter, a 2-approximation algorithm is given in [1]. Now, if $q = 2$, then there are three possible cases for (k_1, k_2): $(3,1)$, $(3,2)$, and $(3,3)$. In the first case k_2 is redundant, so this case is reduced to the case $q = 1$. For the second case a 3-approximation algorithm is given in [11]. Proposition 2.1 generalizes the latter. It also shows that improving approximation guarantees for small values of k can be used to obtain better approximation guarantees for arbitrary values of k.

Proposition 2.1 *Let $\vec{K} = (k_1, k_2, \ldots, k_q)$ be a vector of requirements of root nodes $\vec{R} = (r_1, r_2, \ldots, r_q)$, $k_1 \geq k_2 \geq \ldots \geq k_q$. Suppose that there exists an α_j-approximation algorithm for the multiroot problem with requirement vector $\vec{K}_j = (k_{j+1} - j, \ldots, k_q - j)$, $k_q \geq j + 1$. Then for the original problem there exists a $(2j + \alpha_j)$-approximation algorithm.*

Proof: Apply j times the 2-approximation algorithm for finding a k_i-outconnected subgraph from r_i, $i = 1, \ldots, j$. Let G_j be the union of the resulting j subgraphs. Clearly, G_j is $(k_1, \ldots, k_j)$-outconnected from $(r_1, \ldots, r_j)$, and has cost at most $2jopt$. Moreover, if C is a separator of G_j that separates r_i from some other node, $i \geq j$, and $|C| < k_i$, then $\{r_1, \ldots, r_j\} \subseteq C$. This implies that augmenting G_j to a subgraph which is $(k_{j+1}, \ldots, k_q)$-outconnected from $(r_{j+1}, \ldots, r_q)$ is equivalent to the problem of augmenting $G_j \setminus \{r_1, \ldots, r_j\}$ to a spanning subgraph of $G \setminus \{r_1, \ldots, r_j\}$ which is $\vec{K}_j$-outconnected from $\vec{R}_j = (r_{j+1}, \ldots, r_q)$. Assume that for the latter problem there exists an α_j-approximation algorithm, and let F_j denote its output augmenting edge set. Then $G_j \cup F_j$ is $\vec{K}$-outconnected from $\vec{R}$, and $c(G_j \cup F_j) \leq c(G_j) + c(F_j) = 2jopt + \alpha_j opt = (2j + \alpha_j)opt$. □

Here are some typical applications of Proposition 2.1. Suppose that for some value of j the requirement vector $\vec{K}_j$ has $q - j$ requirements of value $q - j$ each. In this case, the problem for $\vec{K}_j$ is reduced to the problem of finding a minimum cost $(q - j)$-connected spanning subgraph. Note that combining this with the 4-approximation algorithm of [8] for the minimum-cost 7-connected subgraph problem implies a $(2k - 10)$-approximation algorithm for the minimum-cost k-connected subgraph problem, $k \geq 8$. This approximation ratio is better than k for $k = 8, 9$.

Consider now the worst case when there are $k - 1$ requirements each of value k or $k - 1$. Let j be the number of requirements of value k. Then $\vec{K}_j$ has $k - j - 1$ requirements of value $k - j - 1$ each. The problem for $\vec{K}_j$ is the same as the problem of finding $(k - j - 1)$-connected spanning subgraph. If $j = k - 2$, then $k - j - 1 = 1$, $\alpha_j = 1$, and thus we have for this case a $(2k - 3)$-approximation algorithm. The 3-approximation algorithm for $(k_1, k_2) = (3, 2)$ of [11] is a particular case of this approach.

Our $\frac{10}{3}$-approximation algorithm for requirements $(3, 3)$ implies an improvement for any vector $\vec{K}$ with $k - 1$ requirements of value k each. Indeed, let $j = k - 3$. Then $\vec{K}_j = (3, 3)$, and thus we obtain an algorithm with approximation ratio $2(k - 3) + \frac{10}{3} = 2(k - \frac{4}{3})$, which is an improvement over the previously best known $2(k - 1)$-approximation.

The idea of our $\frac{10}{3}$-approximation algorithm for $(k_1, k_2) = (3, 3)$ is as follows. In the first phase, the algorithm uses the 2-approximation algorithm of [9] to find a 2-connected subgraph H of G; in Section 4 we show that $c(H) \leq (\frac{4}{3} + \frac{2}{3n})opt$, where opt denotes the value of an optimal solution to the problem. In the second phase, an additional set of edges

is found by our 2-approximation algorithm for augmenting a 2-connected subgraph to a $(3,3)$-outconnected from (r_1, r_2).

3 Augmenting a 2-connected graph

In this section we will show a 2-approximation algorithm for augmenting a 2-connected subgraph to be $(3,3)$-outconnected. It follows from Menger's Theorem that a graph G is $(3,3)$-outconnected from (r_1, r_2) if and only if either G is 3-connected, or $\{r_1, r_2\}$ is a *unique* 2-separator of G (recall that in this paper, a separator is a "mixed cut" that may contain edges, and that separators that do not contain edges are called cuts). So, for augmenting a 2-connected graph H to be $(3,3)$-outconnected from (r_1, r_2), it is sufficient (and necessary) to "destroy" all the 2-separators of H except of, maybe, $\{r_1, r_2\}$. The first consequence is, that if $\{r_1, r_2\}$ is not a cut of H, the problem is reduced to finding a min-cost 3-connected subgraph. For the latter, a 2-approximation algorithm is given in [1].

To shrink a subset S of $V(G)$ means to combine all nodes in S into a single node s, deleting all edges with both end-nodes in S, and for every edge with one end-node in S, to replace this end-node by s; an edge of a new graph is identified with the corresponding edge of G, and, for every inclusion maximal set of parallel edges, only the cheapest is kept.

Let $\{r_1, r_2\}$ be a cut of a 2-connected spanning subgraph H of G. For a proper subset S of $V(G) \setminus \{r_1, r_2\}$, let us denote $\bar{S} = V(G) \setminus (S \cup \{r_1, r_2\})$. Assume that $\{r_1, r_2\}$ has l sides in H, say $S_1, \ldots, S_l$. Decomposing G into components (with respect to H and $\{r_1, r_2\}$) means the following. Delete the edge (r_1, r_2) from G, if such exists, and add a new *virtual edge* $\tilde{e} = (r_1, r_2)$ of cost 0; this results in a graph $\tilde{G}$. Then the $\{r_1, r_2\}$-*components* of G (see Fig. 1(e)) are the graphs $\hat{G}_1, \ldots, \hat{G}_l$, where $\hat{G}_i = \hat{G}(S_i)$ is obtained from $\tilde{G}$ by shrinking $\bar{S}_i$ into a single *virtual node* $\bar{s}_i$. The *split* $\{r_1, r_2\}$-*components* of G (see Fig. 1(f)) are the graphs $G_1, \ldots, G_l$, where $G_i = G(S_i) = \hat{G}_i \setminus \bar{s}_i$. Clearly, if G is 2-connected, then its every (split) $\{r_1, r_2\}$-component is 2-connected.

The following facts are well known, or can be easily deduced from the well-known structure of 2-cuts. We omit a full proof, and instead refer the reader to [15] for details.

Lemma 3.1 *Let $\{r_1, r_2\}$ be a cut of a 2-connected graph G.*

(1) *If there is a 2-separator in G separating r_1 and r_2 then:*
   (i) *$\{r_1, r_2\}$ has exactly two sides in G;*
   (ii) *G has a 2-separator distinct from $\{r_1, r_2\}$ that* does not *separate r_1 and r_2.*

(2) *$\{r_1, r_2\}$ is a unique 2-separator of G if and only if every split $\{r_1, r_2\}$-component of G with respect to itself is 3-connected.*

Sketch of the proof: It is not hard to see that if S is a side of $\{r_1, r_2\}$, then there is a path between r_1 and r_2 in $G \setminus \bar{S}$. In particular, if $\{r_1, r_2\}$ has l sides, then there are at least l internally disjoint paths between r_1 and r_2. Thus, if there is a 2-separator that separates r_1 and r_2, then (by Menger's theorem) $l \leq 2$, and thus $l = 2$.

Let $\{a_1, a_2\}$ be a 2-separator that separates r_1 and r_2. Note that (r_1, r_2) is not an edge of G. For $i = 1, 2$, if a_i is an edge, let v_i be an (arbitrarily chosen) end-node of a_i distinct from r_1 and from r_2; otherwise (i.e., a_i is a node) let $v_i = a_i$. Note that $\{v_1, v_2\}$ is a 2-cut that separates r_1 and r_2 as well, and, has 2 sides, say T_1, T_2, where $r_1 \in T_1$ and $r_2 \in T_2$. Let S_1, S_2 be the sides of $\{r_1, r_2\}$, where $v_1 \in S_1$ and $v_2 \in S_2$. Now, if $T_i \cap S_j = \emptyset$ for all $i \neq j = 1, 2$, then G must be a cycle of length 4. In this case, r_1 and any edge incident to r_2 is a 2-separator that does not separate r_1 and r_2. Otherwise, assume without loss of generality that $T_1 \cap S_1$ is nonempty. It is not hard to see that $\{r_1, v_1\}$ separates $T_1 \cap S_1$ from $V \setminus (T_1 \cap S_1 \cup \{r_1, v_1\})$.

Both directions of part 2 can be proved by contradiction. If a split $\{r_1, r_2\}$-component of G is not 3-connected, then it has a 2-separator $P \neq \{r_1, r_2\}$. It is not hard to see that then P is also a 2-separator of G. If G has a 2-separator distinct from $\{r_1, r_2\}$, then, by part 1 of the lemma, G has a 2-separator $P \neq \{r_1, r_2\}$ that does not separate r_1 and r_2. It can be verified that P must be a separator of some split $\{r_1, r_2\}$-component of G. □

Our first key observation is:

Lemma 3.2 *Let H be a 2-connected spanning subgraph of a graph G, and let $\{r_1, r_2\}$ be a cut of H with sides $S_1, \ldots, S_l$. Let (G_i) $\hat{G}_i$, $i = 1, \ldots, l$, be the (split) $\{r_1, r_2\}$-components of G with respect to H. Then:*

(i) *If $l = 2$ then, G is $(3,3)$-outconnected from (r_1, r_2) if and only if the following holds:*

- *if G has an edge (u, v) with $u \in S_1$, $v \in S_2$, then G is 3-connected;*
- *if G has no such edge, then each one of G_1, G_2 is 3-connected, and $G \setminus (r_1, r_2)$ is $(3,3)$-outconnected from (r_1, r_2).*

(ii) *If $l \geq 3$ then, G is $(3,3)$-outconnected from (r_1, r_2) if and only if $G \setminus (r_1, r_2)$ is $(3,3)$-outconnected from (r_1, r_2), and for every $1 \leq i \leq l$ holds:*

- *if G has an edge (u, v) with $u \in \bar{S}_i$, $v \in S_i$, then $\hat{G}_i$ is 3-outconnected from $\bar{s}_i$;*
- *if G has no such edge, then G_i is 3-connected.*

Proof:

(i) Clearly, if G contains an edge (u, v) with $u \in S_1$, $v \in S_2$, then $\{r_1, r_2\}$ is not a cut of G. Thus, in this case, G is $(3,3)$-outconnected from (r_1, r_2) if and only if G is 3-connected.

Assume now that G has no such edge. Then $\{r_1, r_2\}$ is a cut of G, and the split $\{r_1, r_2\}$-components of G with respect to itself and with respect to H coincide. Thus, by Lemma 3.1(2), G is $(3,3)$-outconnected from (r_1, r_2) if and only if each of G_1, G_2 is 3-connected. It remains to show that if G is $(3,3)$-outconnected from (r_1, r_2), then so is $G \setminus (r_1, r_2)$. Assume, on the contrary, this is not so. Then, $G \setminus (r_1, r_2)$ has a 2-separator P separating r_1 and r_2. Then, by Lemma 3.1(1ii), $G \setminus (r_1, r_2)$ has a 2-separator $P' \neq \{r_1, r_2\}$ that does not separate r_1 and r_2. It is easy to see, that P' is also a 2-separator of G, a contradiction.

(ii) Note that if $G \setminus (r_1, r_2)$ is not $(3,3)$-outconnected from (r_1, r_2), then there is a 2-separator P that separates r_1 and r_2 in $G \setminus (r_1, r_2)$. Thus, P is also a 2-separator of $H \setminus (r_1, r_2)$. This contradicts Lemma 3.1(1i), since $H \setminus (r_1, r_2)$ is 2-connected, and $\{r_1, r_2\}$ has at least 3 sides in $H \setminus (r_1, r_2)$.

Let S_i be a side of $\{r_1, r_2\}$ in H, $1 \leq i \leq l$.

Claim 1: If there exists an edge $(u, v) \in G$, $u \in S_i$, $v \in \bar{S}_i$, then G is 3-outconnected from r_1 to S_i and from r_2 to S_i if and only if $\hat{G}_i$ is 3-outconnected from $\bar{s}_i$.

Proof: Note that in this case (a) $\{r_1, r_2\}$ is *not* a 2-separator of $\hat{G}_i$, and (b) $\hat{G}_i$ has no 2-separator separating any two from $r_1, r_2, \bar{s}_i$. We now prove both directions by contradiction.

Assume the contrary; that G is 3-outconnected from r_1 to S_i and from r_2 to S_i, but $\hat{G}_i$ is not 3-outconnected from $\bar{s}_i$. Let P be a 2-separator separating $\bar{S}_i$ from some $v \in S_i$ in $\hat{G}_i$. In particular, $\bar{s}_i \notin P$. Since by (a) $P \neq \{r_1, r_2\}$, one of r_1, r_2, say r_1, is not in P. Then, by (b), also P separates r_1 and v. It is not hard to see that P must also separate r_1 and v in G, which is a contradiction.

Assume now that $\hat{G}_i$ is 3-outconnected from $\bar{s}_i$, but G is not $(3,3)$-outconnected from (r_1, r_2). Then, by Lemma 3.1, there is a 2-separator P of G separating one of r_1, r_2, say r_1, from some $v \in S_i$, but P does not separate r_1 and r_2. Thus P is also a separator of $G \cup (r_1, r_2)$. Also, P does not contain an edge or a node that is not in $\hat{G}_i$, since $G_i = \hat{G}_i \setminus \bar{s}_i$ is 2-connected. So, P is a separator of $\hat{G}_i$, which is a contradiction. □

Claim 2: If an edge as in Claim 1 does not exist, then G is 3-outconnected from r_1 to S_i and from r_2 to S_i if and only if G_i is 3-connected.

Proof: Note that under the assumptions of the Claim, S_i is also a side of $\{r_1, r_2\}$ in G. In particular, the $\{r_1, r_2\}$-component G_i of G with respect to itself and with respect to H coincide. Thus, by Lemma 3.1, if G is $(3,3)$-outconnected from (r_1, r_2), then G_i must be 3-connected.

The proof of the other direction is similar to the proof of Claim 1. □

Now, observe that $\cup S_i = V \setminus \{r_1, r_2\}$. Thus, by Lemma 3.1(1i), in this case G is $(3,3)$-outconnected from (r_1, r_2) if and only if for every $i = 1, \ldots, l$ holds: G is 3-outconnected from r_1 to S_i, and from r_2 to S_i. Combining this with Claims 1 and 2 finishes the proof of part (ii) of the Lemma.

The proof of the Lemma is complete. □

The following algorithm makes a 2-connected graph H (3,3)-outconnected from (r_1, r_2). The problematic case is when $\{r_1, r_2\}$ is a cut of H, and has at least 3 sides. The idea is to use Lemma 3.2(ii), which essentially states that a subgraph is a solution to the problem if and only if it can be composed from certain 3-connected graphs and 3-outconnected graphs. These can be found in the (split) $\{r_1, r_2\}$-components within twice the value of an optimal using the algorithms of ([9]) [1]. But this does not guarantee the overall approximation ratio 2, if we compare the cost of each of the subgraphs found to the cost of the directed version of the corresponding "piece" of an optimal solution. Our second observation is that the arcs

of the directed version of any optimal solution can be *partitioned* in a similar way (without every part necessarily being a directed version of a subgraph of G), so that at least one of the 3-connected or 3-outconnected graphs found has cost at most twice the cost of its directed counterpart.

Algorithm for augmenting a 2-connected graph

Input: A graph $\mathcal{G}$ with cost function c on the edges, a 2-connected spanning subgraph H of $\mathcal{G}$ with $c(H) = 0$, and two nodes r_1, r_2 of $\mathcal{G}$;

Output: An edge set $F \subseteq E(\mathcal{G}) \setminus E(H)$ so that $H \cup F$ is $(3,3)$-out-connected from (r_1, r_2);

1. If $\{r_1, r_2\}$ is not a cut of H, then using the algorithm of [1], find a set F of edges such that $H \cup F$ is 3-connected and output F;

2. If $\{r_1, r_2\}$ is a cut of H, then decompose $\mathcal{G}$ with respect to H into $\{r_1, r_2\}$-components $\hat{\mathcal{G}}_1, \ldots, \hat{\mathcal{G}}_l$, and the corresponding split components $\mathcal{G}_1, \ldots, \mathcal{G}_l$;

(i) If $l = 2$ then using the algorithm [1] find, if they exist:

- edge set F' so that $H \cup F$ is 3-connected;
- for $i = 1, 2$, edge sets $F_i'' \subseteq E(\mathcal{G}_i) \setminus E(H_i)$ so that $H_i \cup F_i''$ is 3-connected;

Output F, the cheapest among F' and $F'' = (F_1'' \cup F_2'') \setminus \tilde{e}$;

(ii) If $l \geq 3$ then for $i = 1, \ldots, l$ do:

- Using the algorithm [6], find, if it exists, an optimal $\bar{s}_i \rightarrow$ 3-outconnected subdigraph D_i' in the directed version of $\hat{\mathcal{G}}_i$, and set F_i' to be the edge set of its underlying graph;
- Using the algorithm [1], find, if it exists, an edge set F_i'' in $\mathcal{G}_i$, such that $H_i \cup F_i''$ is 3-connected;
- Set F_i to be the cheapest among found F_i', F_i'';

Output $F = (\bigcup_{i=1}^{l} F_i) \setminus (E(H) \cup \tilde{e})$.

Observe that for the example in Fig. 1, the algorithm computes an optimal solution. Each set F_i' (F_i'') will consist of $E(H_i)$ and the corresponding dashed edge (of the corresponding two thick edges). Thus for each $i = 1, \ldots, 4$, $F_i = F_i' \setminus E(H_i)$, and thus F will consist of the two dashed edges, which is the optimal.

Theorem 3.3 *The above algorithm is a 2-approximation algorithm for the problem of augmenting a 2-connected graph to be* $(3,3)$*-outconnected from its two nodes* (r_1, r_2)*. The time complexity of the algorithm is* $O(n^2 m)$*.*

Proof: The correctness of the algorithm follows from Lemma 3.2.

We now show the approximation ratio. Let $F^* \subseteq E(\mathcal{G}) \setminus E(H)$ be an optimal edge set so that $H \cup F^*$ is $(3,3)$-out-connected from (r_1, r_2), and let F be the output of the algorithm. The other notations are also as in the algorithm.

If $\{r_1, r_2\}$ is not a cut of H, then F^* is an optimal edge set so that $H \cup F^*$ is 3-connected. The algorithm of [1] computes an edge set F so that $H \cup F$ is 3-connected within a factor 2 of the optimal. Thus for this case we have $c(F) \leq 2c(F^*)$.

Assume now that $\{r_1, r_2\}$ is a cut of H. Consider two cases: $l = 2$ and $l \geq 3$.

$l = 2$: We will show that $c(F) = \min\{c(F'), c(F'')\} \leq 2c(F^*)$. Consider two sub-cases:

- F^* has an edge (u, v) with $u \in S_1$ and $v \in S_2$:

  Then, via Lemma 3.2(i), the same analysis as for the previous case is applied, and shows that F' exists, and that $c(F') \leq 2c(F^*)$.

- F^* has no such edge:

  In this case, by Lemma 3.2(i), we can assume $(r_1, r_2) \notin F^*$. For $i = 1, 2$, let $F_i'' = \{(u, v) \in F : (u, v) \in G_i\}$, and $F_i^* = \{(u, v) \in F : (u, v) \in G_i\}$. For the same reason as before, $F_i'' \leq 2c(F_i^*)$. Note that the sets F_1^*, F_2^* are disjoint. Thus $c(F'') \leq c(F_1'') + c(F_2'') \leq 2c(F_1^*) + 2c(F_2^*) = 2c(F^*)$.

$l \geq 3$: by Lemma 3.2(ii), we can assume $(r_1, r_2) \notin F^*$. Let $D^* = D(F^*)$ be the directed version of F^*. Clearly, $c(D^*) = 2c(F^*)$. Let $S_1, \ldots, S_l$ be the sides of $\{r_1, r_2\}$ in H, and let $D_i^* = \{(u, v) \in D^* : u \in \bar{S}_i, v \in S_i\}$, and $F_i^* = \{(u, v) \in F^* : u \in \bar{S}_i, v \in S_i\}$, $i = 1, \ldots, l$. Observe that the sets D_i^* are pairwise disjoint and their union is D^*. We claim that $c(F_i) = \min\{c(F_i'), c(F_i'')\} \leq c(D_i^*)$, $i = 1, \ldots, l$. Consider two sub-cases:

- F has an edge (u, v) with $u \in \bar{S}_i, v \in S_i$:

  Then, by Lemma 3.2(ii), $\hat{H}_i \cup F_i^*$ is 3-outconnected from $\bar{s}_i$. This implies that $D(\hat{H}_i \cup F_i^*)$ is also 3-outconnected from $\bar{s}_i$, or, which is equivalent, $D(\hat{H}_i) \cup D_i^*$ is 3-outconnected from $\bar{s}_i$ (since $D(\hat{H}_i) \cup D_i^*$ is obtained from $D(\hat{H}_i \cup F_i^*)$ by deleting the arcs of $D(F_i^*)$ entering $\bar{s}_i$). Note however, that D_i' is an optimal $\bar{s}_i \rightarrow$ 3-outconnected subdigraph. Thus we have $c(F_i') \leq c(D_i') \leq c(D_i^*)$.

- F has no such edge:

  Then, by Lemma 3.2(ii), $H_i \cup F_i^*$ is 3-connected. The algorithm [1] computes an edge set F_i'' so that $H_i \cup F_i''$ is 3-connected within a factor 2 of the optimal, implying $c(F_i'') \leq 2c(F_i^*)$. Note also that in this case $D_i^* = D(F_i^*)$, and thus $c(F_i'') \leq 2c(F_i^*) = c(D_i^*)$.

This implies that $c(F_i) = \min\{c(F_i'), c(F_i'')\} \leq c(D_i^*)$. Finally,

$$c(F) \leq \sum_{i=1}^{l} c(F_i) \leq \sum_{i=1}^{l} c(D_i^*) = c(D^*) = 2c(F^*).$$

This finishes the proof of the approximation ratio.

We now show the time complexity. The dominating time is spent for finding edge sets F_i'' at step 2(ii). Let n_i and m_i denote the number of the nodes and of the edges of the $\{r_1, r_2\}$-component $\hat{\mathcal{G}}_i$ of $\mathcal{G}$, respectively. For each $i = 1, \ldots, l$, the time required for the algorithm [6] is $O({n_i}^2 m_i)$. Observe that $\sum_{i=1}^{l} n_i \leq n + 3l \leq 4n$, and that $\sum_{i=1}^{l} m_i \leq 3m$ (since an edge of $\mathcal{G}$ appears in at most two split $\{r_1, r_2\}$-components, and the virtual edge appears once in each component). Thus $\sum_{i=1}^{l} {n_i}^2 m_i \leq (\sum_{i=1}^{l} {n_i}^2)(\sum_{i=1}^{l} m_i) \leq (4n)^2 3m = O(n^2 m)$, and the overall time complexity follows. □

4 The general case $k = 3$

We now show how the algorithm described in the previous section can be used to obtain two algorithms for the general case $k = 3$ of our problem. One is a $(\frac{10}{3} + \frac{2}{3n})$-approximation algorithm with complexity $O(n^2 m)$, and the other is a $\frac{10}{3}$-approximation algorithm with complexity $O(n^3 m)$. First, the algorithm computes, using the algorithm [9], a 2-connected subgraph H of G. Then, H is augmented to be $(3,3)$-outconnected from (r_1, r_2), using the algorithm from the previous section. Since we show that $c(H) \leq (\frac{4}{3} + \frac{2}{3n})opt$, this results in a $(\frac{10}{3} + \frac{2}{3n})$-approximation algorithm. A slight modification of this approach leads to a $\frac{10}{3}$-approximation algorithm. We need some known facts and intermediate results before proving that $c(H) \leq (\frac{4}{3} + \frac{2}{3n})opt$.

For any digraph $D = (V, A)$, there is a bijective correspondence between the 0, 1-vectors x indexed by the arcs of A, and the spanning subdigraphs $D_x = (V, A_x)$ of D, where $a \in A_x$ if and only if $x_a = 1$. For any two disjoint subsets $S, T \subset V$ let us denote $x(S,T) = \sum\{x_a : a = (s,t) \in A, s \in S, t \in T\}$. Similar notations are used for graphs. It follows from Menger's Theorem that a digraph D_x is l-outconnected from r if and only if $|V \setminus (S \cup T)| + x(S,T) \geq l$ for every two non-empty disjoint subsets S, T of V so that $r \in S$ (see, for example, [4]). Therefore, there is a bijective correspondence between the feasible integral solutions of the corresponding linear program

$$\begin{array}{lll} \min & cx & \\ s.t. & |V \setminus (S \cup T)| + x(S,T) \geq l & r \in S,\ \emptyset \neq T \subset V \setminus S \qquad (1) \\ & 0 \leq x \leq 1 & \end{array}$$

and the $r \to l$-outconnected subdigraphs of D. A similar result is valid for graphs. For *digraphs* (but *not* for graphs), by [4, Theorem 2.2], this linear program always has an optimal solution which is integral; for such an integral solution x^*, D_{x^*} is an optimal $r \to l$-outconnected subdigraph of D.

The following Lemma implies that the 2-approximation algorithm for the one root problem produces a solution of cost at most twice the value of an optimal solution to the linear program (1).

Lemma 4.1 *Let c^*_{LP} be the value of an optimal solution to the linear program (1), where x is defined on the edges of an undirected graph G. Let D^* be an optimal spanning $r \to l$-outconnected subdigraph of $D(G)$, and let H^* be the (simple) underlying graph of D^*. Then $c(H^*) \leq 2c^*_{LP}$.*

Proof: Let x^* be an optimal solution to the linear program (1) for G. Let x^*_D be defined on the arcs of $D(G)$ as follows: for $e = (u, v) \in E$, $x^*_D(u, v) = x^*_D(v, u) = x^*(e)$. Consider the linear program (1), but with x being defined on the arcs of $D(G)$. By [4], this linear program always has an optimal solution which is integral, namely D^*. Observe that x^*_D is a feasible solution to this linear program. Thus we have: $c(H^*) \leq c(D^*) \leq cx^*_D = 2cx^* = 2c^*_{LP}$. □

Here is a short description of the 2-approximation algorithm of [9] for finding an optimal 2-connected spanning subgraph. Let (s, u) be the cheapest edge of $\mathcal{G}$. The algorithm:
(1) adds an *external* node r and connects it to each of s, u;
(2) finds in the directed version of the graph so obtained, an optimal 2-outconnected subdigraph $\tilde{D}$, and outputs $(\tilde{G} \setminus r) \cup (s, u)$, where $\tilde{G}$ is the (simple) underlying graph of $\tilde{D}$.

Lemma 4.2 *Let G be a graph, $1 \leq l \leq k$ integers, and $\{r_1, \ldots, r_l\}$ a subset of distinct nodes of V. Let opt be the cost of an optimal $(k, \ldots, k)$-outconnected from $(r_1, \ldots, r_l)$ subgraph of G, and let H be a subgraph of G computed as follows:*
(1) Construct a graph G_r by adding an external node r and connecting r to each of $\{r_1, \ldots, r_l\}$ by edges of cost 0;
(2) Find in $D(G_r)$ an optimal $r \to l$-outconnected subdigraph $\tilde{D}$ and output the underlying graph of $\tilde{D} \setminus r$.
Then $c(H) \leq \frac{2l}{k} opt$.

Proof: Consider the linear program (1) with x being defined on the edges of G_r. Note that for any such x, the value of x is the same as the value of the restriction of x to the edges of G. Thus we will assume $x(e) = 1$ for any external edge, since it does not affect the optimal value of (1). Consider also the linear program

$$\begin{array}{lll} \min & cx & \\ s.t. & |V \setminus (S \cup T)| + x(S, T) \geq t & r_1 \in S,\ \emptyset \neq T \subset V \setminus S \\ & \vdots & \\ & |V \setminus (S \cup T)| + x(S, T) \geq t & r_l \in S,\ \emptyset \neq T \subset V \setminus S \\ & 0 \leq x \leq 1 & \end{array} \qquad (2)$$

for which x is defined on the edges of G. Note that feasible integral solutions to (2) correspond to $(t, \ldots, t)$-outconnected from $(r_1, \ldots, r_l)$ spanning subgraphs of G.

Let c^*_1, $c^*_2(t)$ denote the value of an optimal solution to linear programs (1) and (2), respectively. We will show that $c^*_1 \leq c^*_2(l) \leq \frac{l}{k} c^*_2(k)$. Since, clearly, $c^*_2(k) \leq opt$, and, by Lemma 4.1, $c(H) \leq 2c^*_1$, the result follows.

We prove that $c^*_1 \leq c^*_2(l)$ by showing that if x is not feasible for (1) (with $t = l$), and if $x(e) = 1$ for every external edge e, then the restriction of x to G is not feasible for (2). Now, if x is not feasible for (1), and if $x(e) = 1$ for every external edge e, then there are sets $S, T \subset V$, $\emptyset \neq T \subset V \setminus S$, $r \in S$, so that $|V \setminus (S \cup T)| + x(S, T) < l$. By the assumption $x(e) = 1$ for every external edge e, there is $1 \leq i \leq l$ so that $r_i \in S$. Therefore, the restriction of S and x to G, violates for $t = l$ the ith constraint of (2).

Showing that $c^*_2(l) \leq \frac{l}{k} c^*_2(k)$ is straightforward. Indeed, if x is a feasible solution for (2) with $t = k$, then $\frac{l}{k} x$ is a feasible solution for (2) with $t = l$. The proof is complete. □

Lemma 4.1 together with Lemma 4.2 applied for $k = 3$ and $l = 2$ imply that the cost of the 2-connected subgraph found by the algorithm of [9], excepting the cheapest edge added at the end, is at most $\frac{4}{3}opt$, where opt is the value of an optimal solution to our problem. The cost of the cheapest edge of G is at most $\frac{2}{3n}opt$ (since any 3-outconnected subgraph has at least $\frac{3n}{2}$ edges). Thus we will obtain the following intermediate result:

Corollary 4.3 *Let H be the subgraph produced by the algorithm [9] and let opt be the cost of an optimal $(3,3)$-out-connected subgraph from (r_1, r_2). Then $c(H) \leq (\frac{4}{3} + \frac{2}{3n})opt$.*

Our algorithm for $k = 3$ finds a 2-connected spanning subgraph using the algorithm [9], and then augments it to be $(3,3)$-outconnected from (r_1, r_2) using the algorithm from the previous section.

The correctness and the time complexity $O(n^2m)$ of the algorithm follow from the correctness and the time complexity of the algorithm [9], and Theorem 3.3.

We now show the approximation ratio. By Corollary 4.3, $c(H) \leq (\frac{4}{3} + \frac{2}{3n})opt$. By Theorem 3.3, $c(F) \leq 2opt$. Thus $c(H \cup F) \leq c(H) + c(F) \leq (\frac{10}{3} + \frac{2}{3n})opt$.

We can slightly modify the algorithm to get rid of the $\frac{2}{3n}$ term, but the time complexity will increase to $O(n^3m)$. Let us choose an arbitrary node $s \in V(\mathcal{G})$. For every $u \in V(\mathcal{G})$ with $(s,u) \in E(\mathcal{G})$ we:
(1) set $c(s,u) = 0$;
(2) execute the algorithm of [9], connecting the external node to each of s, u; this results in a 2-connected spanning subgraph H_u of $\mathcal{G}$;
(3) Using the algorithm above, find an edge set $F \subseteq E(\mathcal{G})$ such that $H \cup F$ is $(3,3)$-outconnected from (r_1, r_2).
Finally, among the subgraphs $H_u \cup F$ obtained, the algorithm will output the cheapest one. In such a way we are guaranteed to "catch" an edge (s,u) belonging to some optimal solution to our problem. We do not need to pay for such an edge (s,u).

Theorem 4.4 *For the problem of finding an optimal $(3,3)$-outconnected from (r_1, r_2) spanning subgraph there exist two algorithms: one with approximation ratio $(\frac{10}{3} + \frac{2}{3n})$ and time complexity $O(n^2m)$, and the other with approximation ratio $\frac{10}{3}$ and time complexity $O(n^3m)$.*

Acknowledgment I would like to thank Joseph Cheriyan for fruitful discussions and his help. Thanks also to Tibor Jordàn for his comments on a previous version of this work.

References

[1] V. Auletta, Y. Dinitz, Z. Nutov, and D. Parente, A 2-approximation algorithm for finding an optimum 3-vertex connected spanning subgraph, *Journal of Algorithms* **32**, 1999, 21–30.

[2] J. Cheriyan, T. Jordán, and Z. Nutov, Approximating k-outconnected subgraph problems, Approximation Algorithms for Combinatorial Optimization, K. Jansen and

J. Rolim eds., Springer Lecture Notes in Computer Science 1444, 1998, 77-88. (A full version is to appear in *Algorithmica special issue on APPROX'98*.)

[3] Y. Dinitz and Z. Nutov, A 3-approximation algorithm for finding optimum 4,5-vertex-connected spanning subgraphs, *Journal of Algorithms* **32**, 1999, 31–40.

[4] A. Frank, Connectivity augmentation problems in network design, *Mathematical Programming*, State of the Art, J. R. Birge and K. G. Murty eds. (1994), 34–63.

[5] A. Frank and É. Tardos, An application of submodular flows, *Linear Algebra and its Applications*, **114/115** (1989), 329–348.

[6] H. N. Gabow, A representation for crossing set families with application to submodular flow problems, *Proc. 4th Annual ACM-SIAM Symp. on Discrete Algorithms* (1993), 202–211.

[7] S. Khuller, Approximation algorithms for finding highly connected subgraphs, In *Approximation algorithms for NP-hard problems*, Ed. D. S. Hochbaum, PWS publishing co., Boston, 1996.

[8] G. Kortsarz and Z. Nutov, Approximating node connectivity problems via set covers, *manuscript*.

[9] S. Khuller and B. Raghavachari, Improved approximation algorithms for uniform connectivity problems, *J. of Algorithms* **21**, (1996), 434–450.

[10] Z. Nutov, Approximating multiroot 3-outconnected subgraphs, In *Proc. 10th ACM-SIAM Symposium on Discrete Algorithms* (SODA'99), 1999, 951–952.

[11] Z. Nutov, M. Penn, and D. Sinreich, On mobile robots flow in locally uniform networks, *Canadian Journal of Information Systems and Operational Research* (a special issue on Intelligent Scheduling of Robots), Vol. 35, No. 4, 1997, 197-208.

[12] M. Stoer, Design of survivable networks, Lecture Notes in Mathematics 1531, Springer Verlag, Berlin, 1972.

[13] R. Ravi and D. P. Williamson, An approximation algorithm for minimum-cost vertex-connectivity problems, *Algorithmica* **18**, (1997), 21–43.

[14] R. Ravi and D. P. Williamson, *private communication*.

[15] W. T. Tutte, Connectivity in graphs, Ch. 11, Univ. of Toronto Press, 1966.