A 1.8 approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2 *

Guy Even[†] Jon Feldman[‡] Guy Kortsarz[§] Zeev Nutov[¶]

October 20, 2008

Abstract

We present a 1.8-approximation algorithm for the following NP-hard problem: given a connected graph $G = (V, \mathcal{E})$ and an edge set E on V disjoint to $\mathcal{E}$, find a minimum size subset of edges $F \subseteq E$ such that $(V, \mathcal{E} \cup F)$ is 2-edge-connected. Our result improves and significantly simplifies the approximation algorithm with ratio $1.875 + \varepsilon$ of Nagamochi [14].

*Preliminary version appeared in Proceedings of APPROX 2001, Lecture Notes in Computer Science 2129, Springer, pp. 90-101, 2001.

[†]Dept. of Electrical Engineering-Systems, Tel-Aviv University, Tel-Aviv 69978, Israel. E-mail:guy@eng.tau.ac.il.

[‡]Google , NY. E-mail:jonfeld@google.com.

[§]Rutgers University, Camden. Currently Visiting IBM Research, Yorktown Heights. Partially supported by NSF Award Grant number 072887 E-mail:guyk@crab.rutgers.edu.

[¶]The Open University of Israel, 108 Ravutski Street, Raanana, Israel. E-mail:nutov@openu.ac.il.

1 Introduction

1.1 Problem definition and our result

A graph (possibly with parallel edges) is *k-edge-connected* if there are k pairwise edge-disjoint paths between every pair of its nodes. We study the following fundamental connectivity augmentation problem: given a connected undirected graph $G = (V, \mathcal{E})$ and a set of additional edges (called "links") E on V disjoint to $\mathcal{E}$, find a minimum size edge set $F \subseteq E$ so that $G + F = (V, \mathcal{E} \cup F)$ is 2-edge-connected. The 2-edge-connected components of the given graph G form a tree. It follows that by contracting these components, one may assume that G is a tree. Hence, our problem is:

Tree Augmentation Problem (TAP)
Instance: A tree $T = (V, \mathcal{E})$ and a set of links E on V disjoint to $\mathcal{E}$.
Objective: Find a minimum size subset $F \subseteq E$ of edges such that $T \cup F$ is 2-edge-connected.

TAP is sometimes posed as the problem of covering a laminar family (see e.g., [2]). Namely, given a laminar family $\mathcal{E}$ on a groundset V, and an edge set E on V, find a minimum size $F \subseteq E$ such that for every $S \in \mathcal{E}$, there is an edge in F with one endpoint in S and the other in $V - S$. TAP is also equivalent to the problem of augmenting the edge-connectivity from k to $k+1$ for any odd k; this is since the family of minimal cuts of a k-connected graph with k odd is laminar.

The first 2-approximation for TAP (which also holds for the weighted version) was given by Frederickson & Jájá [5]. TAP is APX-hard even if the set E of links forms a cycle on the leaves of T [2]. Improving the approximation ratio below 2 was a long-standing open problem posed by Khuller in [15] as one of the main open problems in connectivity augmentation. The first approximation ratio below 2 was given by Nagamochi [14]; he gave a $(1.875+\varepsilon)$-approximation algorithm for TAP. In the conference version [3] we presented and sketched a proof of a 1.5-approximation algorithm for the problem. Here we give a full proof of the following weaker result:

Theorem 1.1: TAP admits a 1.8-approximation algorithm.

Presenting this result has the following advantages: (1) it improves the ratio $(1.857 + \varepsilon)$ of Nagamochi [14]; (2) our algorithm and its analysis are much simpler than the one in [14]; (3) the major techniques of our 1.5-approximation algorithm are presented.

The 1.5 approximation algorithm will be presented in a sequel paper that will be based on the 1.8-approximation algorithm in this paper.

Several ideas introduced by Nagamochi in [14] are used in this paper (e.g., minimally leaf-closed trees and the lower bound). Key to our analysis is a credit scheme for using the different parts of the lower bound. This credit scheme has a "static" component and a "dynamic" component. The static credit is computed in the beginning of the algorithm and is distributed to different parts of the graph. The dynamic credit is available only after the algorithm "reveals" it by proving that the optimum solution had to be larger than the initial static estimate. We believe that this technique has merit of its own and may be useful for other connectivity problems.

1.2 Related Results

Several particular cases of TAP were considered in the literature. Eswaran & Tarjan [4] presented a linear time algorithm for the case when any edge is legal, namely when E forms a complete graph. Garg et al. [7] presented a 5/3 approximation algorithm for a special case of TAP when the

endnodes of every edge in E are leaves of T. Recently, using some ideas of our paper, Y. Maduel and the last author [13] obtained a very simple 3/2-approximation algorithm for this case. In [13] it is proved that in this particular case the integrality gap of a standard LP-relaxation is at most 5/3, and is also presented an improved LP-relaxation with integrality gap 1.5. For general TAP considered in this paper, it is known that the integrality gap of a standard LP-relaxation is at most 2, and it was recently shown by Cheriyan et al. [1] that the integrality gap is at least 1.5.

A more general problem than TAP is the minimum weight 2-Edge-Connected Spanning Subgraph (2-ECSS) problem, where the goal is to find a minimum weight 2-edge-connected spanning subgraph H of a given complete graph G. TAP is a particular case, when G contains a tree T (or a connected graph) of weight 0, and any other edge in G has weight in $\{1, \infty\}$ (every link has weight 1, while any other edge of G not in T has weight ∞).

A problem sandwiched between 2-ECSS and TAP is the weighted version of TAP (when the edges in E have weights and the goal is to find a minimum weight augmenting edge set) was called in [5] the Bridge-Connectivity Augmentation (BRA) problem. There are few 2-approximation algorithms for the BRA problem. The first algorithm, by Frederickson and Jájá [5] was simplified later by Khuller and Thurimella [16]. These algorithms are based on constructing a directed graph and computing a minimum weight arborescence. The primal-dual algorithm of Goemans & Williamson [9] is another simple combinatorial 2-approximation algorithm for the problem. The iterative rounding algorithm of Jain [10] is an LP-based 2-approximation algorithms. The approximation ratio of 2 for all these algorithms is tight even for TAP. However, while TAP admits an approximation algorithm with ratio better than 2, no such algorithm is known for BRA.

The 2-ECSS problem was vastly studied. For general weights, the best known ratio is 2 by Fredrickson and Jájá [5], which can also be achieved by the the algorithms in [16] and [10]. For particular cases, better ratios are known. Fredrickson and Jájá [6] showed that when the edge weights satisfy the triangle inequality, the Christofides heuristic leads to a 3/2-approximation algorithm. For the special case of 2-ECSS in which edges have $\{1, \infty\}$ weights, the currently best known ratio is 5/4 [11]. Surveys on broad classes of connectivity problems can be found in [15, 12, 8].

2 Preliminaries

Let $T = (V, \mathcal{E})$ be a tree. For $u, v \in V$ let $(u, v) \in \mathcal{E}$ denote an edge in T and uv a link in E between u and v. Let $p(uv)$ denote the path between u and v in T. A link uv *covers* all the edges (and nodes) along the path $p(uv)$. We designate a node r of T as the *root*, and refer to the pair T, r as a *rooted tree* (we do not mention the root when it is clear from the context). The choice of r defines a partial order on V: u is a *descendant* of v (or v is an *ancestor* of u) if v belongs to $p(ru)$; if, in addition, $(u, v) \in T$, then u is a *child* of v, and v is the *parent* of u. The *least common ancestor* of u and v is the farthest from the root common ancestor of u, v (note that it is unique). The *leaves* of T are the nodes in $V - r$ that have no descendants. We denote the leaf set of T by $L(T)$, or simply by L, when the context is clear. The *rooted subtree* of T induced by v and its descendants is denoted by T_v (v is the root of T_v). A subtree T' of T is called a *rooted subtree* of T if $T' = T_v$, for some $v \in V$. For a node set U and a link set M, $U \cap T'$ is the set of nodes in U that belong to T', and $M \cap T'$ is the set of links in M with both endnodes in T'.

To *contract* a (not necessarily rooted) subtree T' of T/I is to combine all nodes in T' into a single node v, named after the least common ancestor of the nodes in T'. The edges and links with both endpoints in T' are deleted. The edges and links with one endpoint in T' now have v as

their new endpoint, but retain their correspondence with the edge or link of the original graph. For simplicity, we say that the new node v obtained by contracting T' "contains" a node u if $u \in T'$.

If we add a link uv to a partial solution I, then the nodes along the path $p(uv)$ belong to the same 2-edge-connected component of the augmented graph $(V, \mathcal{E} \cup I)$. Hence, we may contract the nodes along $p(uv)$. For a set of links $I \subseteq E$, let T/I denote the tree obtained by contracting every 2-edge-connected component of $T \cup I$ into a single node. Since all contractions are induced by subsets of links, we refer to the contraction of every 2-edge-connected component of $T \cup I$ into a single node simply as the contraction of the links in I.

Definition 2.1: A link $u'v'$ is a *shadow* of a link uv if $p(u'v') \subseteq p(uv)$.

A link is *maximal* if it is not a shadow of any other link. Every instance can be rendered closed under shadows by adding all shadows of existing links. We refer to the addition of all shadows as *shadow completion*. Shadow completion does not affect the optimal solution size, since every shadow can be replaced by some maximal link covering all edges covered by the shadow. Thus:

Assumption 2.2: The set of links E is closed under shadows, that is, if $uv \in E$ and $p(u'v') \subseteq p(uv)$ then $u'v' \in E$.

The *up-link* incident to a node u is defined as the link uv such that v is as close as possible to the root; under Assumption 2.2, v is unique and is an ancestor of u. We denote the highest link incident to u by $up(u)$. For a node set $U \subseteq V$, we let $up(U) = \{up(u) : u \in U\}$.

Definition 2.3: [14] Let T' be a subtree of T and let A be a subset of nodes of T. T' is *A-closed* if there is no link in E from $A \cap T'$ to $T \setminus T'$. T' is *leaf-closed* if it is $L(T)$-closed, namely, if there is no link from a leaf of T' to a node in $T - T'$. A tree T' is *minimally leaf-closed* if T' is leaf-closed but any proper subtree of T' is not leaf-closed.

It is easy to see that T' is A-closed if, and only if, all links in $up(A)$ have both endnodes in T'.

Claim 2.4: [14] A minimally leaf-closed subtree T' of T is covered by $up(L(T'))$.

The following reductions allow us to make some simplifying assumptions about TAP instances.

Definition 2.5: A cover F of T is *shadows minimal* if, for every link $uv \in F$, replacing uv by a proper shadow of uv results in a set of links that does not cover T.

Claim 2.6: Let F be a shadows minimal cover of T. Then, there is exactly one link in F incident to every leaf a of T. In particular, the leaf-to-leaf links in F form a matching on L.

Proof: Let a be a leaf. If $au, av \in F$ for $u \neq v$, then we can replace the link av by its shadow ending at the parent of a. Since every edge of T remains covered, this contradicts the shadow minimality of F. Consequently, nodes in the graph induced by $L(T)$ and F have degree at most 1. Hence the leaf-to-leaf links in F form a matching. □

Further simplifying assumptions can be made when we consider the relationship between pairs of edges in T.

Definition 2.7: An edge e of T is *reducible* with respect to a link set E if there is a unique maximal link that covers e, or if there exists an edge $e' \neq e$ such that every maximal link that covers e' also covers e.

If an edge (u, v) is covered by a unique maximal link ab, then adding ab to the cover is an optimal step (i.e., the size of an optimal solution in the residual problem is smaller by one). If an edge e is covered by any maximal link that covers $e' \neq e$, then e may be contracted without affecting the sizes of the optimal solution and the computed solution. Hence, we may preprocess the instance so that:

Assumption 2.8: There are no reducible edges in the tree $T = (V, \mathcal{E})$ with respect to the link set E.

The following statement shows that if there are no reducible edges, then paths in T consisting of degree 2 nodes can not end at leaves.

Proposition 2.9: If there are no reducible edges, then every parent of a leaf has at least two children.

Proof: For the sake of contradiction, suppose that v has a unique leaf child a in T. Let u the parent of v. Thus v has degree exactly 2. Note that if $av \in E$, then av can not be the only link incident to a (as there are no reducible edge). Hence av is not maximal. Thus any maximal link incident to a covers the edge (u, v), which is a contradiction as T has no reducible edges. □

For $X, Y \subseteq V$ and a link set F, let $F(X, Y) = \{xy \in F : x \in X, y \in Y\}$ denote the set of links in F that have one endpoint in X and the other in Y; for $x \in V$ let $\deg_F(x) = |F(x, V)|$ be the degree of x w.r.t. F. For the rest of the paper fix F to be some optimum shadows minimal solution for an instance $T = (V, \mathcal{E}), E$ of TAP.

3 The lower bound and the algorithm

3.1 The Leaf-Stem Lower Bound

Definition 3.1: A node $s \in V - r$ is a *stem* of T if it has exactly two children and both of them are leaves. The link between the two children of a stem, if it exists, is called a *twin-link*.

Claim 3.2: If no edge of T is reducible, then there is a twin-link between the children of every stem.

Proof: Let a, b be the children of a stem s, and let v be the parent of s. If $ab \notin E$, then since there are at least two links that cover the edge (a, s) (because no edge is reducible), every maximal link that covers (a, s) also covers the edge (s, v). Hence, (s, v) is reducible, which contradicts the assumption. □

Lemma 3.3: [The Leaf-Stem Lower Bound]
Let L be the set of leaves and S be the set of stems of T, let $X = V \setminus (L \cup S)$, and recall that F is any shadows-minimal cover of T. Let M be a maximum matching of leaf-to-leaf links among all matchings in E not containing twin links. Then:

$$|F| \geq \frac{2}{3}|L| - \frac{1}{3}|M| + \frac{1}{3}\sum_{x \in X} \deg_F(x). \tag{1}$$

Proof: Let $\hat{M}_F$ be the set of twin links in F and let M_F be the set of leaf-to-leaf links in F that are not twin links. Note that $\deg_F(a) = 1$ for every $a \in L$, by Claim 2.6. In particular, $F(L,L) = M_F \cup \hat{M}_F$ is a matching. Partition F into $F(L,L) = M_F \cup \hat{M}_F$, and all other possible combinations of endpoints of F in L, S and X to get:

$$|F| = |M_F| + |\hat{M}_F| + |F(L,S)| + |F(L,X)| + |F(S,S)| + |F(S,X)| + |F(X,X)| \ .$$

Clearly

$$|L| = 2(|M_F| + |\hat{M}_F|) + |F(L,S)| + |F(L,X)| \ . \tag{2}$$

Note that if a, b are children of a stem s and $ab \in \hat{M}_F$, then $\deg_F(s) \geq 1$. Therefore,

$$|\hat{M}_F| \leq |F(L,S)| + 2|F(S,S)| + |F(S,X)| \ . \tag{3}$$

Multiply (2) by 2/3 and multiply (3) by 1/3 and add and rearrange the terms to get:

$$\begin{aligned} \frac{2}{3}|L| - \frac{1}{3}|M_F| &\leq |M_F| + |\hat{M}_F| + F(L,S) + \frac{2}{3}|F(L,X)| + \frac{2}{3}|F(S,S)| + \frac{1}{3}|F(S,X)| \leq \\ &\leq |F| - \frac{1}{3}\left(|F(L,X)| + |F(S,X)| + 2|F(X,X)|\right) = |F| - \frac{1}{3}\sum_{x \in X} \deg_F(x) \ . \end{aligned}$$

Thus, since $|M_F| \leq |M|$ (as M a maximum matching of all matching without twin links and M_F is *some* matching without twin links), we get:

$$|F| \geq \frac{2}{3}|L| - \frac{1}{3}|M_F| + \frac{1}{3}\sum_{x \in X} \deg_F(x) \geq \frac{2}{3}|L| - \frac{1}{3}|M| + \frac{1}{3}\sum_{x \in X} \deg_F(x) \ .$$

□

The term $\frac{2}{3}|L| - \frac{1}{3}|M|$ in (1) is "known" to the algorithm in advance, as we can find a maximum matching M as described. But the last term $\frac{1}{3}\sum_{x \in X} \deg_F(x)$ depends on the degrees of nodes in X w.r.t. F. This part of the lower bound is revealed during the run of the algorithm. A slightly weaker lower bound already appeared in the paper of Nagamochi [14] (Nagamochi did not consider explicitly the last term in (1)). Note that a link in F with one endnode in X adds 1/3 to the lower bound, while a link with both endnodes in X adds 2/3 to the lower bound.

3.2 The credit scheme: coupons and tickets

We explain how we use the Leaf-Stem Lower Bound. Let U denote the set of leaves unmatched by M, so $|U| = |L| - 2 \cdot |M|$. By multiplying both sides of (1) by 1.8 and rearranging terms, we obtain

$$1.8 \cdot |F| \geq 1.2 \cdot |U| + 1.8 \cdot |M| + 0.6 \cdot \sum_{x \in X} \deg_F(x) \ . \tag{4}$$

We derive from (4) the "smallest" lower bound that is enough for our purposes. Let M' be the set of links in M incident to a child of a stem s so that the other child a of s is unmatched. For every such link in M', move 0.2 units of lower bound from a to this link; this gives at least 2 units of lower bound to the link and a unit of lower bound to a. Also, assume that every leaf in U contributes 1

and not 1.2 to the lower bound. Finally, replace the 1.8 multiplying $|M - M'|$ by 1.5, and the 0.6 multiplying the last term in by 0.5. This gives:

$$1.8 \cdot |F| \geq |U| + \frac{3}{2} \cdot |M - M'| + 2 \cdot |M'| + \frac{1}{2} \cdot \sum_{x \in X} \deg_F(x) \ . \tag{5}$$

We prove the following statement that implies Theorem 1.1.

Theorem 3.4: There exists a polynomial time algorithm that for an instance of TAP computes a solution I of size at most the right-hand size of (5). Thus $|I| \leq 1.8 \cdot |F|$.

Initially, the algorithm assigns *coupons* to unmatched leaves and to links in M so that the total number of coupons assigned equals the sum of the first three terms in the right hand side of (5). For technical reasons, we also assign 1 coupon to the root r.

Initial assignment of coupons:

- Every unmatched leaf gets 1 coupon, and r gets 1 coupon.
- Every link in $M - M'$ gets 3/2 coupons, and every link in M' gets 2 coupons.

Compound nodes. Our algorithm iteratively contracts certain subtrees of T. We refer to the nodes created by contraction as *compound nodes*. Compound nodes always own 1 coupon. Non-compound nodes are referred to as *original nodes* (of T). Each time a contraction takes place, the new compound node gets 1 coupon, which together with the links added is paid by the total credit of the contracted tree. For technical reasons, r is also considered as a compound node.

Tickets. To charge the last term $\frac{1}{2} \sum_{x \in X} \deg_F(x)$ in (5), we introduce *tickets*. If we *prove* that $\deg_F(x) \geq k$ for some $x \in X$, then x owns $k/2$ coupons, or k tickets (hence a ticket worth 1/2 coupon). In fact, even if we prove that $\sum_{x \in X \cap T'} \deg_F(x) \geq k$ (in our case we will have $k = 1$ or $k = 2$) for a subtree T' of T/I, then we may give $k/2$ coupons to the node set $X \cap T'$, without knowing exactly the degrees w.r.t. F of the nodes in $X \cap T'$. We refer to such an event as "claiming k tickets in T' " and say that "T' has k tickets" (note however that tickets are really claimed on nodes in $X \cap T'$ only). Of course, every ticket is claimed only once; this is so, since immediately after claiming tickets in T' (that is, in $X \cap T'$), the whole tree T' is contracted into a compound node, while tickets are never claimed on compound nodes. Note that this enables us to use tickets inside T' even if we do not know their exact "location" in $X \cap T'$.

Summarizing, we use the following notation for the credit distributed in a subtree T' of T/I.

1. Every unmatched leaf owns one coupon, every compound node owns one coupon, and every link in M with both endpoints in T' owns 3/2 coupons. Let $coupons(T')$ denote the total amount of coupons owned by T'.
2. Every node $x \in X \cap T'$ (namely, x is not a leaf neither a stem) owns $\deg_F(x)$ tickets. A ticket is worth half a coupon. Let $tickets(T') = \sum_{x \in X \cap T'} \deg_F(x)$ be the number of tickets in T'.
3. Let $credit(T') = coupons(T') + \frac{1}{2} tickets(T')$. We multiply the number of tickets by 1/2 because every ticket is worth half a coupon.

3.3 Greedy contractions and invariants of the algorithm

Our algorithm iteratively finds a subtree T' of T/I and its cover B', adds B' to I, contracts T', and assigns 1 coupon to the new compound node. Theorem 3.4 will be proved, if we follow the following simple rule. Whenever such an action is performed, we have:

$$credit(T') \geq |B'| + 1 \ . \tag{6}$$

This implies that at any step of the algorithm, $|I| + coupons(T/I)$ is at most the right hand side of (5) plus 1. In particular, when T is contracted into a single node that still owns 1 coupon (the one of r) we have: I covers T and $|I|$ is at most the right hand side of (5). We emphasize again, that as $tickets(T') = \sum_{x \in X \cap T'} \deg_F(x)$, tickets are claimed only on nodes in $X \cap T'$, which are all non-compound nodes. As immediately after tickets are claimed in $X \cap T'$, the whole tree T' is contracted into a compound node, tickets are never claimed twice.

Two such simple operations, that may not use rooted subtrees, and rely on coupons only, are as follows.

Definition 3.5: A *greedy contraction* is one of the following two types:

1. A *greedy* 1*-contraction* of a link ab can be performed if $p(ab)$ owns 2 coupons (this includes the cases: (i) each of a, b owns a coupon; (ii) $ab \in M$ and $p(a, b)$ has a compound node; (iii) $ab \in M'$). Then the link ab is added to I, $p(a, b)$ is contracted, and the new compound node gets a coupon.

2. A *greedy* 2*-contraction* of links $a_1a_2, b_1b_2 \in M$ can be performed if $p(a_1a_2) \cap p(b_1b_2) \neq \emptyset$. Then both a_1a_2, b_1b_2 are contracted and added to I, and the new compound node gets a coupon.

After the initial assignment of coupons, the next step of the algorithm is to apply all possible greedy contractions exhaustively; clearly, this can be done in polynomial time. After exhausting greedy contractions, the instance satisfies the following two invariants w.r.t. a partial solution I, which continue to hold during the algorithm:

Invariant 3.6: [Leaves-Stems Invariant]

1. Every original leaf is either a leaf of T/I or is contained in a compound node of T/I.

2. Every original stem is contained in a compound node of T/I.

Invariant 3.7: [Coupons Invariant]

1. Every link in M owns $3/2$ coupons.

2. Every unmatched leaf and every compound node of T/I (including r) owns 1 coupon.

The next statement shows that Part 2 of the Leaves-Stems Invariant and Part 1 of the Coupons Invariant hold after the greedy contractions are exhausted (for other parts, this is immediate).

Lemma 3.8: Let I be any edge set added to exhaust the greedy contractions right after M is computed (so there is no greedy contraction in T/I). Then every original stem of T, if any, is contained in some compound node of T/I, and all the links in M' are contracted.

Proof: Let a, b be a twin pair of an original stem s of T. If a and b are both unmatched, then each of them owns 1 coupon. Hence a greedy 1-contraction applies with the twin link ab, which exists by Claim 3.2. If a and b are both matched, then a greedy 2-contraction applies with the links in M incident to a, b. If b is matched and a is unmatched, then a greedy 1-contraction applies with the link in M' incident to the twin b of a; in particular, all links in M' are contracted. Note that every original stem s participates in one of such contractions. The case a is matched and b is unmatched is symmetric. Hence, after greedy contractions are exhausted, every original stem is contained in some compound node. □

Remark: One reason that the proof of the 1.8 ratio is much easier than that of the 1.5 ratio, is the immediate disappearance of stems after exhausting greedy contractions. The reason we give 2 coupons to a link in M', is to be able to contract all the original stems and place them into compound nodes already at the beginning of the algorithm. We could give instead to every unmatched leaf 1.2 coupons and to every link in M 1.8 coupons, but the invariant of having 1.2 coupons on every compound node is not sustainable, because in the last case in the proof of Lemma 3.8, the created compound node can get only one coupon.

Note that for the purpose of eliminating original stems, we could consider only greedy contractions of links incident to children of stems. However, we also need the following property:

Invariant 3.9: [Matching Invariant]

1. M is a matching on the original leaves (thus by Claim 2.6 $\deg_F(a) = 1$ for any matched leaf a).
2. Contracting any subset of M does not create a new leaf (after exhausting greedy contractions).

Lemma 3.10: If no greedy contraction applies, then Part 2 of the Matching Invariant 3.9 holds, namely, contracting any subset of M does not create a new leaf.

Proof: As no greedy 2-contraction applies, $p(a_1a_2) \cap p(b_1b_2) = \emptyset$ for any distinct $a_1b_1, a_2b_2 \in M$. Therefore, the contraction of a subset of M creates a leaf if, and only if, there is a single link $ab \in M$ whose contraction creates a leaf. Note that in the original tree T the contraction of ab does not create a leaf, since M contains no twin links (and by Proposition 2.9). Thus, if the contraction of ab creates a leaf in T/I, then $p(ab)$ contains a compound node in T/I. This holds as some structure that hanged out of a node of $p(ab)$ and contained a leaf is now gone (this structure was avoiding the contraction of ab in T from creating a new leaf). But then ab gives a greedy 1-contraction, contradicting the assumption. □

3.4 The algorithm

Our approach is iteratively to exhaust greedy contractions, and then to find a rooted subtree T' of T/I with its cover B', to contract T' with B', and to leave a coupon on the created compound node (in order to satisfy the Coupons Invariant 3.7). If we require not to over spend the credit provided by the right hand side of (5), we need (6) to hold. In the next section we will prove the following statement:

Lemma 3.11: Suppose that Invariants 3.6, 3.7, and 3.9 hold for T and I. Then there exists a polynomial time algorithm that finds a rooted subtree T' of T/I and a cover $B' \subseteq E$ of T' so that (6) holds, namely: $credit(T') \geq |B'| + 1$.

Algorithm 1 *Approx*$(T = (V, \mathcal{E}), E)$ (A 1.8-approximation algorithm)

1: $I \leftarrow \emptyset$.
2: Contract reducible edges of T and update I accordingly.
3: $M \leftarrow$ maximum matching of leaf-to-leaf links among non twin-links in T/I.
4: **while** T/I has more than one node **do**
5: Exhaust greedy contractions and update I accordingly.
6: Find a subtree T' of T/I and a cover B' of T' as in Lemma 3.11.
7: Contract T', give 1 coupon on the new compound leaf, and set $I \leftarrow I \cup B'$.
8: **end while**
9: Return I.

Algorithm *Approx* initiates $I \leftarrow \emptyset$ as a partial cover. It applies the reductions from Section 2 (specifically, we need Assumptions 2.2 and 2.8), computes a maximum matching M on the leaves among the non-twin links, and initiates the credit scheme as described in Section 3.2. In the main loop, the algorithm iteratively exhausts greedy contractions, computes T', B' as in Lemma 3.11, adds B' to I, contracts T', and leaves a coupon on the created compound leaf. The stopping condition is when I covers T, namely, when T/I is a single node. By Lemmas 3.8 and 3.10 we have:

Claim 3.12: Invariants 3.7, 3.6, 3.9, hold after greedy contraction are exhausted in Line 5 of *Approx*.

It is easy to verify that all actions needed to be taken by the algorithm can be implemented in polynomial time. The credit scheme used implies that the algorithm computes a solution I as in Theorem 3.4, namely $|I|$ is at most 1.8 times the right-hand size of (5). Hence the approximation ratio of 1.8 follows from the Leaf-Stem Lower Bound, and the proof of Theorem 1.1 is now complete.

4 Proof of Lemma 3.11

4.1 Semi-closed trees and their properties

A naive approach to find T', B' as in Lemma 3.11 is by taking T' to be minimally leaf-closed and $B' = up(L(T'))$. However, we do not have enough credit for that, since $|L(T')|$ can be much larger than $coupons(T')$, not to mention the one extra unit of credit. This is since every link $b_1b_2 \in M$ owns 3/2 coupons, while taking the up-links of b_1, b_2 requires 2 coupons. Hence we define T' and its cover B' so that $coupons(T') \geq |B'|$ holds, and one extra unit of credit is provided with the help of tickets. Our main new idea is to use minimally semi-closed trees, defined below, which admit such a cover B', assuming invariants from Section 3.3 hold. In what follows, we say that a matching M on the leaves of T/I is *proper* if it satisfies Part 2 of the Matching Invariant 3.9, namely, if contracting any subset of M does not create a new leaf.

Definition 4.1: (Semi-closed tree)
A rooted subtree T' of T/I is *semi-closed* (w.r.t. a proper matching M) if the following holds:

- For any $b_1b_2 \in M$ either both b_1, b_2 belong to T', or none of b_1, b_2 belongs to T'.
- T' is closed w.r.t. its unmatched leaves.

T' is *minimally semi-closed* if T' is semi-closed but any proper subtree of T' is not semi-closed.

Note that a semi-closed T' is *not* leaf-closed (this why we called it "semi-closed"); T' is closed with respect to every unmatched leaf, but matched leaves may have links to nodes outside T'.

The following statement gives a constructive definition of semi-closed trees (assuming M is a proper matching).

Claim 4.2: Any semi-closed subtree $T' = (T/I)_v$ of T/I can be obtained as follows.

(i) Contract every link in M to get $T/(I \cup M)$.

(ii) Find a leaf-closed rooted subtree $T'' = (T/(I \cup M))_v$ of $T/(I \cup M)$.

(iii) $T' = (T/I)_v$, namely T' is obtained from T'' by "decontracting" every link in M.

Thus T' is (minimally) semi-closed if, and only if, T'' is (minimally) leaf-closed.

Proof: Let $T' = (T/I)_v$ be a semi-closed subtree of T/I. By contracting every link in $M \cap T'$, we obtain a subtree T'' of $T/(I \cup M)$ rooted at v. The set of unmatched leaves of T' is exactly the set $L(T'')$ of leaves of T'', since the matching M is proper. Thus T'' is (minimally) leaf-closed if, and only if T' is (minimally) semi-closed. □

The following statement explains how we intend to cover *minimally* semi-closed trees.

Lemma 4.3: Let T' be a semi-closed subtree of T/I w.r.t. a proper matching M. Let $B(T')$ consist of $M \cap T'$ (namely, links in M with both endnodes in T') plus the up-links of the unmatched leaves of T'. If T' is minimally semi-closed, then $B(T')$ covers T', and we call $B(T')$ the *basic cover* of T'.

Proof: Let $T'' = T'/(M \cap T')$ be as in Claim 4.2. As T'' is a minimally leaf-closed subtree of $T/(I \cup M)$, $up(L(T''))$ covers T'', by Lemma 2.4. As M is proper, $L(T'')$ is exactly the set of unmatched leaves of T'. Consequently, $(T' \cap M) \cup up(L(T''))$ covers T'. The statement follows. □

Let T' be a (not necessarily minimally) semi-closed tree and let $C(T')$ be the set of non-leaf compound nodes of T' (recall that this includes r, if $r \in T'$). Note that then:

$$coupons(T') - |B(T')| = \frac{1}{2} \cdot |M \cap T'| + |C(T')| \ . \tag{7}$$

This is explained as follows. We have $B(T') = up(L(T')) \cup (M \cap T') = up(L(T'')) \cup (M \cap T')$. Thus

$$B(T') = |L(T')| + |M \cap T'| \tag{8}$$

Now count $coupons(T')$. By Coupons Invariant 3.7, every non-leaf compound node and every unmatched leaf adds one coupon to $coupons(T')$, and any matching link in T' adds 3/2 coupons. Thus

$$coupons(T') = |L(T')| + |C(T')| + \frac{3}{2}|M \cap T'|. \tag{9}$$

Then (7) follows by subtracting (8) from (9).

We will show that except one special type (see Definition 4.7), a semi-closed tree T' and its basic cover $B(T')$ satisfy (6). We will also show that if all semi-closed trees are of this special type, then we can find a "larger" tree T' and its cover B' that satisfy (6).

In what follows, let T' be a (not necessarily minimally) semi-closed tree and let v be the root of T'. Unless stated otherwise, we assume that $v \neq r$. Note that T' always contains at least one

unmatched leaf. To see that note that it can not be that $T' = v$ (such a tree will not be leaf closed or else there is no feasible solution). If T' contains matched pairs, by Matching Invariant 3.9 part 2 the contraction of the matched pairs does not create a new and in particular it does not result in a tree containing v only. Hence the tree (after the contraction hence before the contraction as well) must have a leaf. If T' does not contain a matched pair the claim follows since $T' \neq v$.

4.2 Semi-closed trees with $credit(T') \geq |B(T')| + 1$

Note that if $|M \cap T'| \geq 2$ or if $|C(T')| \geq 1$, then the right hand side of (7) is at least 1. This implies:

Claim 4.4: $coupons(T') \geq |B(T')| + 1$ in each one of the following cases:

- T' has a non-leaf compound node (a particular case is when $r \in T'$).
- T' has at least 2 matched pairs.

In the following statement we claim tickets for the first time in the paper. Intuitively, the next lemma says that (assuming no greedy 1-contraction applies) every link in F incident to a node in $U \cap T'$ contributes a ticket, unless its other endnode is a matched leaf.

Lemma 4.5: Suppose that all the compound nodes of T' are leaves, and let U be the set of unmatched leaves of T/I. Then:

$$tickets(T') \geq |U \cap T'| - 2 \cdot |M \cap T'| + 1 \ . \tag{10}$$

Furthermore, if T' is leaf-closed then $tickets(T') \geq 1$.

Proof: Note that there is no link between two nodes in U (as otherwise a greedy 1-contraction applies), and that T' is U-closed. Thus a link in F covering a leaf $a \in U \cap T'$ contributes a ticket, unless its other endnode is a matched leaf. E.g., if $a \in U \cap T'$ and $ax \in F$, and if x is not a matched leaf, then $x \in X$, and as T' is U-closed, $x \in X \cap T'$ and so T' owns a ticket. In addition, as we assume that $v \neq r$, there must be a link from T' to $T - T'$ covering the edge between v and its parent. This link cannot have an endnode in $U \cap T'$, as T' is U-closed. Thus the endnode of this link in T' contributes a ticket, unless this endnode is a matched leaf. In particular, T' has a ticket if T' is leaf-closed.

Summarizing, $|U \cap T'| + 1$ links are needed to cover the unmatched leaves and the edge between v and its parent, and every such link contributes a ticket unless it is incident to a matched leaf. Now recall that there is exactly one link in F incident to every matched leaf, by Part 1 of the Matching Invariant 3.9 and Claim 2.6. Thus exactly $2 \cdot |M \cap T'|$ links in F are incident to the matched leaves. The statement follows. □

Claim 4.6: Suppose that all the compound nodes of T' are leaves. Then:

- If T' has no matched pair (a particular case is when T' has at most 2 leaves) then T' has 2 tickets.
- If T' has one matched pair and at least 4 leaves, then T' has a ticket.

Thus in both cases $credit(T') \geq |B(T')| + 1$.

Proof: This follows from (10). If T' has no matched pair then $tickets(T') \geq |U \cap T'| + 1 \geq 2$. If T' has one matched pair and at least 4 leaves, then $|U \cap T'| \geq 4 - 2 = 2$, hence $tickets(T') \geq |U \cap T'| - 2 \cdot |M \cap T'| + 1 \geq 2 - 2 + 1 = 1$. In both cases, $credit(T') - |B(T')| \geq \frac{1}{2}(|M \cap T'| + tickets(T')) \geq 1$. □

The following type of trees is the only type of semi-closed trees for which our "usual methods" above can not be used to prove that $credit(T') \geq |B(T')| + 1$.

Definition 4.7: (Deficient tree)
Suppose that T' has exactly 3 leaves and one matched pair, all its compound nodes are leaves, and that T' is not leaf closed. Let a denote the unmatched leaf of T'. Then T' is *deficient* (see Figure 1) if there is an ordering b_1, b_2 of its matched pair so that:

(i) $ab_1 \in E$ and the contraction of ab_1 does not create a leaf.

(ii) There exists $b_2x \in E$ so that $x \in T - T'$ (namely, T' is not $\{b_2\}$-closed).

If the ordering b_1, b_2 is not unique (this can happen if T' is as in Figure 1(b)), then we assume that if $up(b_1) = b_1u_1$ and $up(b_2) = b_2u_2$ then u_2 is an ancestor of u_1.

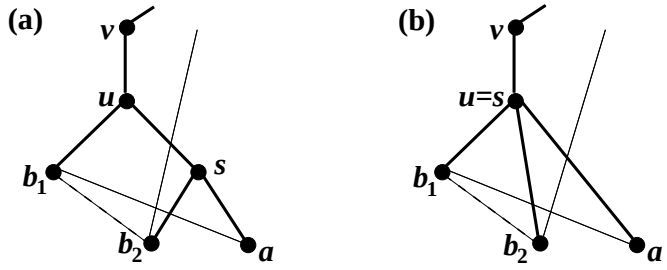

Figure 1: Deficient trees; tree edges are shown by bold lines, matching links by dashed lines. Edges vu, us in the figure can in fact be paths, possibly of the length zero; s is not a stem of the original tree T, and a may not be a leaf of T so its degree in F might be large, but b_1, b_2 are original leaves.

Clearly, checking if T' is deficient or not can be done in polynomial time.

Lemma 4.8: Suppose that T' has exactly 3 leaves and exactly one matched pair, all its compound nodes are leaves, and that T' is not leaf closed. If T' is not deficient, then T' has a ticket, and thus $credit(T') \geq |B(T')| + 1$.

Proof: We prove that if T' has no ticket then T' is deficient. Let b_1, b_2 be the matched pair of T'. Let ya be the link in F covering a. If $y \notin \{b_1, b_2\}$ then $y \in T' \cap X$, and $y \in T'$ since T' is $\{a\}$-closed. Also, y is not a compound node, by assumption. Hence in this case T' has a ticket. Thus assume $y \in \{b_1, b_2\}$.

If the contraction of ay creates a leaf s, it can only happen in the case $y = b_2$ in Figure 1(a), and the link b_2a exists and $b_2a \in F$. The created leaf s must be covered in F by a link incident to a node in $p(ab_2)$. This node can not be an internal node of $p(ab_2)$ as then it is either a compound node or has a ticket, and it cannot be b_1 as $\deg_F(b_1) = 1$. Thus the link in F covering the creates

leaf s must be incident to a, say az. As the tree is a-closed, z belongs to T' and z is not compound. We cannot have $z = b_2$, since b_2 has degree 1 in F. Thus $z \in X$ and T' has a ticket.

Thus w.l.o.g. we assume that $ab_1 \in F$ and the contraction of ab_1 does not create a leaf. Consider any link $wx \in F$ covering v, where $w \in T'$ and $x \in T - T'$. If w is not a leaf of T', then T' has a ticket at w. We have $w \neq b_1$ as $ab_1 \in F$. Also, $w \neq a$, as T' is $\{a\}$-closed. It follows therefore that $w = b_2$, and the statement follows. □

Summarizing, we obtain the following statement:

Corollary 4.9: Let T' be a (not necessarily minimally) semi-closed tree. If $credit(T') < |B(T')| + 1$ then T' is deficient.

4.3 Finding a good tree when all minimally semi-closed trees are deficient

Note that we can compute *all* semi-closed subtrees of T/I in polynomial time, as every semi-closed tree is a rooted subtree of T/I, and since we can check in polynomial time whether a given subtree is semi-closed. If T/I has a minimally semi-closed subtree T' that is not deficient, then T' and $B(T')$ satisfy the requirement of Lemma 3.11. Thus we may assume that all minimally semi-closed subtrees of T/I are deficient. In this case, we show that we can still find a semi-closed tree $\tilde{T}$ and its cover $\tilde{B}$ so that $credit(\tilde{T}) \geq |\tilde{B}| + 1$. But before that we need the following simple statement.

Claim 4.10: Let T' be a deficient tree and let a, b_1, b_2 be as in Definition 4.7. If $\tilde{T}$ is a rooted subtree of T/I containing b_2 that is $\{b_2\}$-closed, then $\tilde{T}$ properly contains T' and $\tilde{T}$ is not deficient.

Proof: As $\tilde{T}$ is $\{b_2\}$-closed, and there is a link $b_2x \in E$ with $x \in T - T'$, the root of $\tilde{T}$ is a proper ancestor of the root of T'. It remains to show that if $\tilde{T}$ is semi-closed then $\tilde{T}$ is not deficient. Note that $\tilde{T}$ is $\{a\}$-closed, since T' is. Thus the only possibility for $\tilde{T}$ to be deficient is if we are in the case of Figure 1(b), so that $b_2a \in E$ and $\tilde{T}$ is not $\{b_1\}$-closed. However, in Definition 4.7 we assume that b_2 has higher up-link than b_1, which implies that $\tilde{T}$ is $\{b_1\}$-closed. This is a contradiction. □

Definition 4.11: The tree $\tilde{T}$ and its cover $\tilde{B}$ are defined as follows, see Figure 2. Let $\tilde{M}$ be the matching obtained from M by replacing every link $b_1b_2 \in M$ that belongs to some minimally semi-closed deficient tree T' by the link b_1a, where a, b_1, b_2 are as Definition 4.7; so b_1a exists and its contraction does not create a leaf, thus $\tilde{M}$ is a proper matching. Then $\tilde{T}$ is any minimally semi-closed tree w.r.t. the new proper matching $\tilde{M}$. The link set $\tilde{B} = (\tilde{T} \cap \tilde{M}) \cup up(L(\tilde{T}))$ is defined as the basic cover of $\tilde{T}$ w.r.t. $\tilde{M}$ (note that $\tilde{B}$ indeed covers $\tilde{T}$, by Lemma 4.3).

Remark: Note that there might be links in $M \cap \tilde{M}$ (this is the set of links in M that do not belong to any deficient tree), see Figure 2(a). In fact, it might be that T/I has many leaves, but there is only one minimally semi-closed subtree T', and this T' is deficient. An example is as follows. Take a path $r = v_1, \ldots, v_{2k}$, and attach to every v_i a leaf u_i. Then attach to v_{2k} a tree T' as in Figure 1, so $v_{2k} = v$. The matching M is b_1b_2 plus the appropriate u_i, v_i edges. The tree T' is a unique minimally semi-closed subtree in this example, and T' is deficient.

Recall that we use the notation $B(T') = (T' \cap M) \cup up(L(T'))$ for any semi-closed tree T', and that by Lemma 4.3 $B(T')$ covers T' if T' is minimally semi-closed. Here and everywhere, unless stated otherwise, "semi-closed" means semi-closed w.r.t M. We will prove that $\tilde{T}$ is semi-closed,

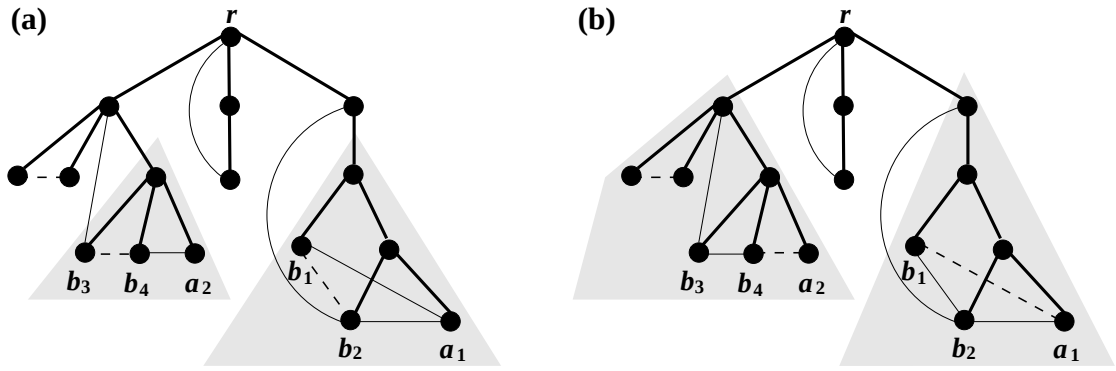

Figure 2: Illustration to Definition 4.11. Tree edges are shown by bold lines, matching links by dashed lines. (a) An example when all minimally semi-closed trees are deficient. (b) The matching $\tilde{M}$ and the minimally semi-closed trees w.r.t. $\tilde{M}$.

but this does not imply that $B(\tilde{T})$ covers $\tilde{T}$, since $\tilde{T}$ is not *minimally* semi-closed. Instead, we use $\tilde{B}$ to cover $\tilde{T}$, and note that $\tilde{B} \neq B(\tilde{T})$. We will prove that $|\tilde{B}| = |B(\tilde{T})|$. As $\tilde{T}$ is semi-closed, then by Corollary 4.9 (which is valid for *any* semi-closed tree) $credit(\tilde{T}) \geq |B(\tilde{T})| + 1 = |\tilde{B}| + 1$, unless $\tilde{T}$ is deficient. Consequently, the following statement finishes the proof of Lemma 3.11.

Lemma 4.12: Suppose that all semi-closed subtrees of T/I are deficient, and let $\tilde{T}$ and $\tilde{B}$ be as Definition 4.11. Then $\tilde{T}$ is semi-closed (w.r.t. M), $\tilde{T}$ is not deficient, and $|\tilde{B}| = |B(\tilde{T})|$. Consequently, $credit(\tilde{T}) \geq |\tilde{B}| + 1$.

Proof: We prove that if $\tilde{T}$ intersects a deficient tree T' with leaves a, b_1, b_2 as in Definition 4.7, then $\tilde{T}$ is $\{b_2\}$-closed. As $T', \tilde{T}$ are rooted trees, they share a leaf. If this leaf is b_2, we are done. If this leaf is b_1 or a, then both $b_1, a \in \tilde{T}$, since $\tilde{T}$ is semi-closed with respect to $\tilde{M}$. Thus the least common ancestor u of b_1, a is in $\tilde{T}$. This implies $b_2 \in \tilde{T}$, since b_2 is a descendant of u (see Figure 1).

Clearly, $\tilde{T}$ intersects some deficient tree T'. Thus by Claim 4.10, $\tilde{T}$ is not deficient.

We show that $\tilde{T}$ is semi-closed. Note that $b_1b_2 \in M \setminus \tilde{M}$ if, and only if, b_1, b_2 belong to some deficient tree T'; thus either $b_1, b_2 \in \tilde{T}$ (if T' is a subtree of $\tilde{T}$), or none of b_1, b_2 in $\tilde{T}$ (otherwise). The same holds for $b_1b_2 \in M \cap \tilde{M}$, by the definition of $\tilde{T}$. It remains to prove that if $a \in \tilde{T}$ is a leaf unmatched by M then $\tilde{T}$ is $\{a\}$-closed. If a is unmatched by $\tilde{M}$, this follows from the definition of $\tilde{T}$. Otherwise (a is matched by $\tilde{M}$), a is the unmatched leaf of a deficient T'. Then T' is $\{a\}$-closed, T' is a subtree of $\tilde{T}$, and thus $\tilde{T}$ is also $\{a\}$-closed.

To see that $|\tilde{B}| = |B(\tilde{T})|$, note that $\tilde{B}$ is obtained from $B(\tilde{T})$ by replacing every $b_1b_2 \in M \setminus \tilde{M}$ that belongs to a deficient tree T' contained in $\tilde{T}$ by the link b_1a (T', a, b_1, b_2 as in Definition 4.7). Hence $\tilde{B}$ and $B(\tilde{T})$ coincide on links not belonging to deficient trees, while for links belonging to deficient trees there is a bijective correspondence between links in $\tilde{B}$ and $B(\tilde{T})$.

The last statement follows from Corollary 4.9 and Claim 4.10. Indeed, as $|\tilde{B}| = |B(\tilde{T})|$ and $\tilde{T}$ is semi-closed, we can have $credit(\tilde{T}) < |\tilde{B}| + 1$ if, and only if, $\tilde{T}$ is deficient, by Corollary 4.9. But we proved that $\tilde{T}$ is not deficient, thus $credit(\tilde{T}) \geq |\tilde{B}| + 1$, as claimed. □

Acknowledgment: We thank two anonymous referees for many useful comments.

References

[1] J. Cheriyan, H. Karloff, R. Khandekar, and J. Koenemann, On the integrality ratio for tree augmentation. Manuscript, 2008.

[2] J. Cheriyan, T. Jordán, and R. Ravi, On 2-coverings and 2-packing of laminar families. In *ESA*, 510–520, 1999.

[3] G. Even, J. Feldman, G. Kortsarz and Z. Nutov, A 3/2-approximation for augmenting a connected graph into a two-connected graph. In *APPROX*, 90–101, 2001.

[4] K. P. Eswaran and R. E. Tarjan, Augmentation problems, *SIAM J. Computing* 5:653–665, 1976.

[5] G. N. Frederickson and J. Jájá, Approximation algorithms for several graph augmentation problems, *SIAM J. Computing* 10:270–283, 1981.

[6] G. N. Frederickson and J. Jájá, On the relationship between the biconnectivity augmentation and traveling salesman problem. *Theoretical Computer Science* 19(2):189–201, 1982.

[7] N. Garg, R. Khandekar, and K. Talwar, Covering a laminar family. Manuscript, 2000.

[8] M. X. Goemans and D. P. Williamson, The primal-dual method for approximation algorithms and its applications to network design problems. Chapter 4 in *Approximation algorithms for NP-hard problems*, Ed. D. S. Hochbaum, PWS Publishing co., Boston, 1996.

[9] M. X. Goemans and D. P. Williamson, A general approximation technique for constrained forest problems, *SIAM J. Computing*, 24:296–317, 1995.

[10] K. Jain, A factor 2 approximation algorithm for the Generalized Steiner Network problem. *Combinatorica* 21(1):39-60, 2001.

[11] R. Jothi, B. Raghavachari and S. Varadarajan, A 5/4-Approximation Algorithm for Minimum 2-Edge-Connectivity, In *SODA*, 725–734, 2003.

[12] G. Kortsarz and Z. Nutov, Approximating minimum cost connectivity problems. Chapter 58 in *Handbook of Approximation Algorithms and Metahueristics*, Ed. T. F. Gonzales, Chapman & Hall/CRC, 2007.

[13] Y. Maduel and Z. Nutov, Covering a laminar family by leaf to leaf links. Manuscript, 2008

[14] H. Nagamochi, An approximation for finding a smallest 2-edge connected subgraph containing a specified spanning tree. *Discrete Applied Mathematics*, 126:83-113, 2003.

[15] S. Khuller, Approximation algorithms for finding highly connected subgraphs, Chapter 6 in *Approximation algorithms for NP-hard problems*, Ed. D. S. Hochbaum, PWS Publishing co., Boston, 1996.

[16] S. Khuller and R. Thurimella, Approximation algorithms for graph augmentation, *J. of Algorithms*, 14:214–225, 1993.