A 3/2-approximation algorithm for augmenting the edge-connectivity of a graph from 1 to 2 using a subset of a given edge set

(EXTENDED ABSTRACT)

Guy Even[1], Jon Feldman[2], Guy Kortsarz[3], and Zeev Nutov[3]

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

[2] Massachusetts Institute of Technology,
Laboratory for Computer Science, Cambridge, MA, 02139, USA.
jonfeld@theory.lcs.mit.edu

[3] Open University of Israel,
16 Klauzner St, Tel-Aviv, Israel. {guyk,nutov}@oumail.openu.ac.il

Abstract. We consider the following problem: given a connected graph $G = (V, \mathcal{E})$ and an additional edge set E, find a minimum size subset of edges $F \subseteq E$ such that $(V, \mathcal{E} \cup F)$ is 2-edge connected. This problem is NP-hard. For a long time, 2 was the best approximation ratio known. Recently, Nagamochi reported a $(1.875 + \varepsilon)$-approximation algorithm. We give a new algorithm with a better approximation ratio of 3/2 and a practical running time.

1 Introduction

The 2-Edge Connected Subgraph Problem Containing a Spanning Tree (2-ECST) is defined as follows. The input consists of an undirected graph $G(V, \mathcal{E})$ and a set of additional edges (called "links") $E \subseteq V \times V$. A subset $F \subseteq E$ is called an augmentation of G into a 2-edge-connected graph if $G(V, \mathcal{E} \cup F)$ is 2-edge connected. The goal is to find an augmentation F of G into a 2-edge-connected graph with fewest links. This problem is NP-Complete [2]. We present a 1.5-approximation algorithm for the 2-ECST problem.

The 2-edge-connected components of G form a tree. It follows that by contracting these components, one may assume that G is a tree. Hence, the 2-ECST Problem is equivalent to the Tree Augmentation Problem (TAP) defined as follows. The input consists of a tree $T(V, \mathcal{E})$ and a set of links $E \subseteq V \times V$. The goal is to find a smallest subset $F \subseteq E$ such that $G(V, \mathcal{E} \cup F)$ is 2-edge connected.

Previous Results. Frederickson & Jájá [7] presented a 2-approximation to the weighted version of TAP. Improving the approximation ratio below 2 was posed by Khuller [12] as one of the main open problems in graph augmentation. Nagamochi & Ibaraki [11] presented a 12/7-approximation algorithm for TAP. However, the proof of the approximation ratio in [11] contains an error. Recently, Nagamochi [10] reported a $(1.875+\varepsilon)$-approximation algorithm. The time complexity of the algorithm in [10] depends exponentially on $1/\epsilon$ (alternatively, the approximation ratio can be increased to 1.9). Our paper uses a few techniques that appear in [10, 11] including: leaf-close trees, shadows, stems, and the usage of maximum matchings that forbid certain links. Our lower bound (Claim 12) is a strengthening of [10, Lemma 4.2].

Related Results. The weighted version of 2-ECST (or equivalently TAP) is called Bridge-Connectivity Augmentation (BRA) in [7]. In the BRA problem, the input consists of a complete graph $G(V,E)$ and edge weights $w(e)$. The goal is to find a minimum weight subset of edges F such that $G'(V,F)$ is 2-edge connected. The 2-ECST is simply the case in which the edges have $\{0,1,\infty\}$ weights; the edges of the connected graph have zero weight, links have unit weight, and all the rest have infinite weight.

There are a few 2-approximation algorithms for the BRA problem. The first algorithm, by Frederickson and Jájá [7] was simplified later by Khuller and Thurimella [13]. These algorithms are based on constructing a directed graph and computing a minimum weight arborescence. The primal-dual algorithm of Goemans & Williamson [8] and the iterative rounding algorithm of Jain [9] are LP-based 2-approximation algorithms. The approximation ratio of 2 for all these algorithms is tight. Fredrickson and Jájá [7] showed also that when the edge costs satisfy the triangle inequality, the Christofides heuristic leads to a 3/2-approximation algorithm.

Cheriyan et. al. [3] presented a 17/12-approximation algorithm for the special case of BRA in which edges have $\{1,\infty\}$ weights. A 4/3-approximation algorithm was presented for $\{1,\infty\}$ weights by Vempala & Vetta [14]. Eswaran & Tarjan [4] presented a linear time algorithm for the special case of BRA in which all edges have unit weights. Surveys for broad classes of augmentation problems can be found in [12] and [5].

Equivalent Problems. The following two problems are equivalent to TAP, in the sense that the corresponding reductions preserve approximation ratios (e.g., see [2]): **(1)** Given a laminar family $\mathcal{S}$ of proper subsets of a ground set U, and an edge set E on U, find a minimum subset $F^* \subseteq E$ that "covers" $\mathcal{S}$ (that is, for every $S \in \mathcal{S}$, there is an edge in F, with one endpoint in S and the other in $V-S$). **(2)** Given a k-connected graph $G=(U,E_0)$ with k odd, and an edge set E on U disjoint to E_0, find a min size edge set $F^* \subseteq E$ such that $G \cup F$ is $(k+1)$-edge connected. Due to space limitations, all proofs as well as details of the algorithm are omitted. See the full version for these parts.

2 Preliminaries

Let $T(V,\mathcal{E})$ denote a tree defined over a vertex set V. A *rooted tree* is a tree T with a node $r \in V$ designated as a root. The root induces a partial order on V as follows. For $u,v \in V$, we say that u is a *descendant* of v, and that v is an *ancestor* of u, if v lies on the path from r to u in T; if, in addition, (u, v) is an edge of T, then u is a *child* of v, and v is the *parent* of u. We denote the parent of u by $p(u)$. The *least common ancestor*, $\textsc{lca}(u, v)$, of u and v is the common ancestor of u and v with the largest distance to the root. We refer to a rooted tree as a pair (T, r), and do not mention the root when it is clear which vertex is the root. In the TAP problem, we designate an arbitrary node r of T as the *root*.

The *leaves* of a rooted tree T are the nodes in $V - r$ having no descendants. We denote the leaf set of T by $L(T)$, or simply by L, when the context is clear.

Consider a rooted tree (T, r) and a vertex $v \in V$. The *rooted subtree* of T induced by the descendants of v (including v) is denoted by T_v. Note that v is the root of T_v. A subtree T' of T is called a *rooted subtree* if $T' = T_v$, for some vertex $v \in V$. We say that a rooted subtree T' of T is *minimal w.r.t. property $\mathcal{P}$* or that T' is *$\mathcal{P}$-minimal* if T' satisfies property $\mathcal{P}$, but no proper rooted subtree of T' satisfies property $\mathcal{P}$.

To *contract* a subset X of V is to combine all nodes in X to make a single new node x. All edges and links with both endpoints in X are deleted. All edges and links with one endpoint in X now have x as their new endpoint, but retain their correspondence with the edge or link of the original graph. If $r \in X$, then x becomes the root of the new tree. If parallel links arise, we consider an arbitrary underlying set of links. For simplicity, we say that such x contains a node v if $v \in X$. For a set of links $F \subseteq E$ let T/F denote the tree obtained by contracting every 2-edge connected component of $T \cup F$ into a single node.

Let $p(u, v)$ denote the path in T between u and v. Consider a link $(u, v) \in E$. If we decide to add (u, v) to the solution F, then the vertices along the path $p(u, v)$ belong to the same 2-edge connected component of $T \cup F$. Hence, we would like to contract the vertices along $p(u, v)$. Instead of defining a new node into which the vertices are contracted, it is convenient to regard this contraction as a contraction of the vertices along $p(u, v)$ into the least common ancestor, $\textsc{lca}(u, v)$. The advantage of this convention is that the contracted tree is a subtree of the original tree. Note that we may obtain T/F simply by contracting the paths corresponding to the links of F one by one. Moreover, the resulting tree does not depend on the order by which the links are contracted.

We say that a link (u, v) *covers* all the edges along the path $p(u, v)$. This terminology enables us to formulate TAP as a covering problem. Namely, find a minimum size set of links that covers the edges of T. We extend this terminology and refer to covering a an edge (u, v) where v is the parent of u as *covering the node u*.

We say that a link (u', v') is a *shadow* of a link (u, v) if $p(u', v') \subseteq p(u, v)$, and (u', v') is a *proper shadow* of (u, v) if the inclusion is proper, that is $p(u', v') \subset p(u, v)$. A link is *maximal* if it is not a proper shadow of any other link. We say that a cover F of T is *shadows-minimal* if for every link $(u, v) \in F$ replacing (u, v) by a proper shadow of (u, v) results in a link set that does not cover T.

An edge (u, v) of a tree T is *reducible* w.r.t. a link set E if there is an edge (u', v') such that every maximal link that covers (u', v') also covers (u, v).

Definition 1. *A tree T and an edge set E defined over the same vertex set are* proper *if (a) for every edge (u, v) of T, there are at least 2 maximal links covering it, and (b) no edge in T is reducible.*

Note that non-proper TAP instances can be reduced without effecting the size of the solution. Hence from this point we assume that every tree we wish to cover is proper. One convenient property of proper trees is that every parent of a leaf has at least two children. By induction, it follows that if (T, E) is a proper pair, then there cannot be a path (of length 2 or more) consisting of degree-2 vertices ending at a leaf.

The following assumption simplifies many of the arguments we make.

Assumption 2. *The set of links E is closed under shadows, that is, if $(u, v) \in E$ and $p(u', v') \subseteq p(u, v)$ then $(u', v') \in E$.*

Every instance can be made to be closed under shadows by adding shadows of existing links as new links (i.e. shadow completion). Shadow completion does effect the solution size, since every shadow can be replaced by a maximal link.

In what follows, the TAP instance consists of the input tree $T = (V, \mathcal{E})$ and the set of links E. We assume that E covers T, that is, the graph $(V, \mathcal{E} \cup E)$ is 2-connected; this can be tested beforehand by contracting all the links.

3 Motivation: a 2-approximation algorithm

We start the discussion with a 2-approximation algorithm. This rather simple algorithm introduces the notions of leaf-closed subtrees, up-links, and the disjointness condition. Our 3/2-approximation extends these ideas.

For a set of links or edges F and a vertex v, let $\deg_F(v)$ denote the degree of v in the graph (V, F). Consider an arbitrary cover $F \subseteq E$ of T. A simple well known lower bound follows from the degree-sum of F, namely, $\sum_v \deg_F(v)$, as follows. Every leaf of T has at least one link of F incident to it, so the degree-sum of F is at least $|L(T)|$, and hence $|F| \geq |L(T)|/2$.

The ratio between the value of an optimal solution and this lower bound can be arbitrarily large; if T is a path, and E consists of links parallel to the edges of T, then $|L(T)|/2 = 1$, but the value of an optimal solution is $|V| - 1$. An approximation algorithm based on this lower bound requires applying the lower bound to a subproblem of the original problem and that a local ratio argument be used. This means that we compare the cost invested in covering the subproblem with the cost the optimum invests in this subproblem. We recurse on the remaining subproblem. Since we compare the cost of our cover of the subproblem with the cost of an optimal solution of the subproblem, it is necessary that the an optimal solution covers each of the charged subproblems by disjoint sets of links. We refer to this requirement as the *disjointness condition.*

Suppose the subproblem we choose is to cover the leaves (namely, the edges in $\mathcal{E}$ incident to the leaves of T). This can be done using $|L(T)|$ links; we just choose a set of links F with one link per leaf. The cost of covering the leaves is

$|L(T)|$, and the leaf lower bound is $|L(T)|/2$, so the local approximation ratio for solving the subproblem is 2. However, recursing on T/F and repeating the same approximation argument fails. The reason is that a link in an optimal solution can cover a leaf in $L(T)$ as well as a leaf in $L(T/F)$. Hence, the disjointness condition is not met. This motivates the following definition and lemma, which appeared in the paper of Nagamochi and Ibaraki [11] for $U = L(T)$.

Definition 3. *Let $U \subseteq V$. A rooted subtree T' of T is* U-closed *(w.r.t. E) if every link in E having one node in $U \cap V(T')$ has it other node in $V(T')$.*

For every $U \subseteq V$, the tree T is U-closed. Hence the set of rooted subtree that are U-closed is not empty. This implies that, for every $U \subseteq V$, there exists a rooted subtree that is U-closed minimal.

The highest link incident to a node u is denoted by $up(u)$. Namely, among the links emanating from u and whose other endpoint is along the path $p(r, u)$, $up(u)$ is the link that covers the largest subpath of $p(r, u)$. For $U \subseteq V$, let $up(U) = \bigcup\{up(u) : u \in U\}$.

Lemma 4. *Let $U \subseteq V$, and let T' be a U-closed minimal rooted subtree of T. Then* $\mathrm{up}(U \cap V(T'))$ *covers T'.*

Lemma 4 is interesting when $L(T') \subseteq U$ (otherwise T' consists of single leaf).

A rooted subtree T' of T is *leaf-closed* if it is $L(T')$-closed. Lemma 4 implies the following claim, that appears in [11].

Claim 5. *[11] A leaf-closed minimal subtree T' of T is covered by* $\mathrm{up}(L(T'))$.

We now present a simple 2-approximation algorithm. The algorithm is recursive and its parameter is a tree T and a link set F. The algorithm starts with the whole tree T and an empty link set $F = \emptyset$. It finds a minimally leaf-closed subtree T', adds $up(L(T'))$ to F, and recurses on $T/up(L(T'))$ and F, until T is contracted into a single node.

```
Initialization: $F \leftarrow \emptyset$.
Algorithm $cover(T, F)$:
1. If $|V(T)| = 1$, then return $F$ and stop.
2. Compute a leaf-closed minimal subtree $T'$ of $T$.
3. $F \leftarrow F \cup up(L(T'))$, $T \leftarrow T/up(L(T'))$.
4. Recurse: $cover(T, F)$.
```

Clearly, the algorithm returns a feasible cover. We can prove that the size of the computed cover is 2-optimal by induction on k, the number of iterations of the algorithm. The induction base $k = 1$ follows from Claim 5. For the induction step, consider a leaf-closed minimal subtree T' of the first iteration of the algorithm. Let F^* be an arbitrary optimal cover of T. Let F^*_L be the links in F^* incident to the leaves of T', and let F^*_{res} be the links in F^* that cover at least one link not in T'. Then $|F^*_L| \geq |L(T')|/2$, and since T' is leaf closed, all the links in F^*_L have their both endpoints in $V(T')$. Hence, F^*_L and F^*_{res} are disjoint. By the induction hypothesis, the recursive call for covering $T/up(L(T'))$ produces a solution F_{res} with at most $2|F^*_{res}|$ links. Hence, the total number of links in the solution produced by the algorithm is:

$$|up(L(T'))| + |F_{res}| = |L(T')| + |F_{res}| \leq 2|F^*_L| + 2|F^*_{res}| = 2|F^*|.$$

In the following section we present a lower bound that strengthens the leaf lower bound, as well as a generalization of the idea of leaf-closed minimal subtrees in order to improve the approximation ratio to 3/2.

4 Lower bound and the credit scheme

In this section we present a new lower bound on the size of a minimum cover. This lower bound is based on a maximum matching consisting of a subset of links. A credit scheme is established based on the lower bound.

Notation and preliminaries. A non-root node $s \in V - r$ is a *stem* of a tree T if it has exactly two children, both of them are leaves, such that there is a link joining them. Such two leaves are called a *twin pair*, and the link between them is called a *twin link*. The edge connecting a stem s and its parent is called a *stem-edge*. We denote the the set of stems of a tree T by $St(T)$.

Leaves and stems play a special role in our algorithm. For short, we say that node of T is *special* it is a leaf or a stem. We denote the set of special nodes in T by $Sp(T)$. We denote the set of non-special nodes in T by $\overline{Sp}(T)$. We often use L, St, Sp for short to denote the set of leaves, the set of stems, and the set of special nodes if the context is clear. A rooted subtree T' of T is *leaf-stem-closed* if it is $L(T) \cup St(T)$-closed.

For $X, Y \subseteq V$ and a set of links F, let $F_{X,Y}$ denote the be the set of links in F that have one endpoint in X and the other in Y. We denote the number of links in F that emanate from a node v by $\deg_F(v)$. For a subset $U \subseteq V$ of nodes we denote the degree-sum of nodes in U with respect to a set of links F by $\deg_F(U)$. Namely, $\deg_F(U) = \sum_{v \in U} \deg_F(v)$.

A link (u, v) is called a *backward link* if u is a descendant of v or vice-versa. A link that is not a backward link is called a *side link.* A link e with a non-special endpoint is called a *bad* link.

We can also take advantage of another property of shadow-minimality. Let F be a shadow minimal cover of T. Every two links in F that share an endpoint cover disjoint sets of edges. Therefore, there is exactly one link in F emanating from every leaf v of T, and $F_{L(T),L(T)}$ is a matching.

Consider a matching M that consists of leaf-to-leaf links. A leaf $\ell \in L$ is *unmatched* with respect to M if M lacks a link incident to ℓ. The set of unmatched leaves in T with respect to M is denoted by $UL_T(M)$. A stem $s \in S$ is *uncovered* with respect to M if the corresponding stem-edge is not covered by the links of M. The set of uncovered stems in T with respect to M is denoted by $US_T(M)$.

Let F^* denote an optimal shadows minimal cover of T. The optimality of F^* implies that it is not possible that both the links (s, ℓ_1) and (s, ℓ_2) are in F^*, for a stem s and its children ℓ_1 and ℓ_2. The reason is that these two links can be replaced by the link (ℓ_1, ℓ_2).

Assumption 6. *Without loss of generality, a cover F^* of T lacks links from a stem to one of its children.*

We use Assumption 6 to assume that a stem s is uncovered with respect to $F^*_{L,L}$ iff F^* contains the twin-link between the children of s. The following claim follows.

Claim 7. *Let s denote a stem. If F^* satisfies Assumption 6, then, $deg_{F^*}(s) \leq 1$. Moreover, $deg_{F^*}(s) = 1$ iff s is uncovered with respect to $F^*_{L,L}$.*

4.1 Two special small trees

In this section we consider two special subtrees. Subtrees of the first kind are called H-structures and play an important role in the algorithm. Subtrees of the second kind can be reduced to H-structures. Part (A) of Figure 1 depicts a *twin-thorn subtree.* This is a rooted subtree with 3 leaves: twin leaves a_1, b_1, and a leaf b_2 called the "thorn". The leaves a_1 and b_1 are a twin-pair that are the children of a common stem s. The stem s is connected to the root-node v by a path, all the internal nodes along which are nodes of degree 2. The node v has two children, one is an ancestor of s, and the other is a leaf b_2.

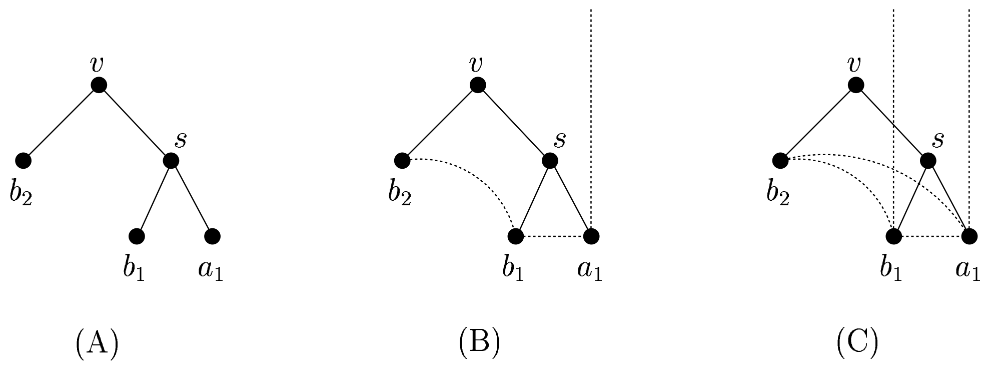

Fig. 1. (A) a twin-thorn subtree (B) an H-structure (C) A twin-thorn reducible to an H-structure.

Definition 8. *A subtree T_v is defined as an H-*structure *if: (i) T_v is isomorphic to a twin-thorn subtree, (ii) one of the twin-leaves is linked only to nodes that are its ancestors, and (iii) the other twin-leaf is linked to the thorn (b_1 may have other links incident to it). Part (B) of Figure 1 depicts an H-structure.*

The twin-leaf in an H-structure that is linked only to ancestors is called a *locked-leaf* (node a_1 in Fig. 1). The twin-link in an H-structure that has at least one side-link emanating from it is called a *locking-leaf* (node b_1 in Fig. 1). The link between the locking-leaf and the thorn is called the *locking-link* (link (b_1, b_2) in Fig. 1). A key observation is that if (a_1, b_1) does not belong to a feasible cover, then a_1 is covered by a bad link. Let F denote a set of links. An H-structure T_v is called F*-activated* if F contains the locking-link in T_v.

Reducible twin-thorn subtrees Consider the twin-thorn subtree T_v depicted in part (C) of Figure 1. Suppose that the links incident to nodes in T_v satisfy the following conditions: (a) (a_1, b_2) and (b_1, b_2) are links in E, and (b) the only side links emanating from a_1 and b_1 are (a_1, b_2) and (b_1, b_2), respectively.

Observe that if $(b_1, b_2) \in F^*$, then without loss of generality $up(a_1) \in F^*$. Similarly, if $(a_1, b_2) \in F^*$, then $up(b_1) \in F^*$. Assume that the path covered by the link $up(a_1)$ is not shorter than the path covered by the link $up(b_1)$. Informally, this means that $up(a_1)$ is "higher reaching" than $up(b_1)$. It follows that the links $up(a_1)$ and (b_1, b_2) cover all the edges that are covered by $up(b_1)$ and (a_1, b_2). Hence, we may discard the link (a_1, b_2) from E without increasing the size of an optimal solution. Note that this reduces T_v to an H-structure.

The above discussion justifies the first stage of the algorithm in which every twin-thorn subtree that satisfies the conditions described above is reduced to an H-structure. We summarize this reduction in the following assumption.

Assumption 9. *Let T_v denote a twin-thorn subtree. If the twin-leaves are linked only to the thorn or to their ancestors, then exactly one of the twin-leaves is linked to the thorn.*

4.2 Matchings

Our algorithm first computes maximum matching consisting of leaf-to-leaf links except for twin-links and locking-links. We denote this matching by M.

Now, consider an H-structure T_v. If both endpoints of a locking link (b_1, b_2) are unmatched leaves in $UL(M)$, we can add the locking-link (b_1, b_2) into the matching M. Let M^+ denote a maximal matching obtained by augmenting the matching M with all possible locking links. Let LL_{M^+} denote the set of locking-links in M^+, namely, $LL_{M^+} = M^+ - M$.

The following notation is now defined. Let M^* denote the set of leaf-to-leaf links in F^*, except for twin-links. Let $\mathit{diff}(M, M^*)$ denote the set of links in M both endpoints of which are not matched by links in M^*. Observe that $\mathit{diff}(M, M^*)$ is non-empty only if M^* is not a maximal matching. A link $e \in \mathit{diff}(M, M^*)$ is referred to as a *singleton*; the reason is that e shares no endpoints with other links in the symmetric difference of M^* and M. Let LL_{F^*} be the set of locking links in F^*. Note that $|LL_{F^*}|$ equals the number of F^*-activated H-structures.

Consider the following two terms: (a) $|M^*| - |M^+| + \mathit{diff}(M, M^*)$; the number of non-twin links in $F^*_{L,L}$ minus the number of links in M^+ plus the number of singletons, and (b) $|LL_{F^*}|-|LL_{M^+}|$; the number of F^*-active H-structures minus the number of M^+-active H-structures.

Claim 10. $|M^*| - |M^+| + |\text{diff}(M, M^*)| \leq |\text{LL}_{F^*}| - |\text{LL}_{M^+}|$

Observe that Claim 10 also holds if one considers a leaf-stem closed subtree T' and only considers the matching links and the H-structures within T'.

The lower bound Consider a leaf-stem closed subtree T' of T. Recall that F^* denote a shadows-minimal optimal cover of T that satisfies Assumption 6. The following claim proves a lower bound on the number of links of F^* both endpoints of which are inside T' in terms of the number of leaves, uncovered stems, unmatched leaves and bad links in T'.

Claim 11. *Let T' denote a leaf-stem closed subtree, then*

$$|F^*_{V(T'),V(T')}| \geq \frac{1}{2} \cdot L + \frac{1}{3} \cdot |\text{US}_{T'}(F^*_{L,L})| + \frac{1}{6} \cdot |\text{UL}_{T'}(F^*_{L,L})| + \frac{1}{3} \cdot deg_{F^*_{V(T'),V(T')}}(\overline{\text{Sp}}).$$

In the following lower bound claim, T' is a leaf-stem closed subtree, F^* is a shadows-minimal optimal cover of T, and M is a maximum matching consisting of leaf-to-leaf links without twin links or locking links. Let M^+ denote its augmentation with locking links. We denote by $M_{T'}$ and $M^+_{T'}$ the set of links in M and M^+ that incident to leaves in T'.

For every non-special node v let $\text{deg}'_{F^*_{V(T'),V(T')}}(v)$ denote the degree of v with respect to links of F^* both endpoints of which are in T' not including links connecting it to locked leaves in F^*-activated H-structures.

In the next claim M_S refers only to singletons internal to T'.

Claim 12. (The matching lower bound)

$$\frac{3}{2} \cdot |F^*_{V(T'),V(T')}| \geq \frac{3}{2} \cdot |M^+_{T'}| + \left(|L| - 2 \cdot |M^+_{T'}|\right) + \frac{1}{2}|\text{LL}_{M_{T'}}| + \frac{|M_S(F^*)|}{2} + \frac{1}{2} \cdot deg'_{F^*_{V(T'),V(T')}}(\overline{\text{Sp}}).$$

5 A 3/2-approximation algorithm

In this section we present the techniques used in our 3/2-approximation algorithm. The algorithm is listed at the end of this section for reference. The algorithm is rather elaborate; more details and analysis will appear in the full version.

5.1 The coupon scheme

In order to use our lower bound, we need to apply a credit scheme. We say that a *coupon* can pay for a contraction of one link. We distribute coupons based on the matching lower bound from Claim 12. We assign $3/2$ coupons to every pair of vertices that are matched by a link in M^+, and we assign 1 coupon to every leaf of T not covered by M^+. Furthermore, every M^+-activated H-structure gets half a coupon. By the matching lower bound, the number of coupons used is no more than $3|F^*|/2$.

In addition to coupons, we have tickets. Each ticket is worth half a coupon. We consider two types of tickets: (a) matching tickets - a matching ticket is given to each link in M_S. (b) golden tickets - a non-special node that has a link of F^* incident to it receives a golden ticket.

The difference between coupons and tickets is that the algorithm can assign coupons directly from the matching M^+ it computes. Tickets are harder to reveal and require a proof that they exist.

5.2 Algorithm techniques

In the 2-approximation algorithm, we covered a leaf-closed subtree and recursed. In this algorithm, we are looking to cover a leaf-stem-closed subtree, and recurse. The main difficulty is to find enough coupons to pay for some leaf-stem-closed subtree. One important insight is that by using the coupon distribution described above, and maintaining the proper invariants, we can contract parts of the tree in order to find a leaf-stem-closed tree on which to recurse.

One invariant we maintain is a set A of *active* nodes, each of which always contain a coupon. We also build a set J of candidate links. As we add links to J, we contract the paths that the links of J cover. Hence we operate mostly on the tree T/J. The set A begins as the set of unmatched leaves.

The first step we take is to apply *greedy contractions.* An *α-greedy contraction* is defined as follows. Let T' be a subtree of T/J, (not necessarily rooted) coverable by α links (i.e. T' is a connected union of α paths, each covered by a link). If there are at least $\alpha+1$ coupons contained in T', then we can add these α links to J. The new node (created by contracting all the vertices of T' together) inherits the extra coupon, and becomes an active node. Thus we maintain the invariant that all active nodes contain a coupon.

After applying all possible 1-greedy and 2-greedy steps, we attempt to find a leaf-stem-closed subtree by finding an A-closed (or *active-closed*) subtree T' of $T/(J \cup M^+)$, the tree obtained by contracting all the links of J and those of the matching M^+. If this is leaf-stem-closed, we are done, and we can recurse, since we have enough coupons to pay for the links of J and M^+. The disjointness condition requires that the portions of F^* that are charged for solving each of the subproblems be disjoint. This way "double charging" is avoided. Note that the matching lower bound is applied to the links that are contained in a leaf-stem closed subtree T'. If the algorithm succeeds in contracting T', then the disjointness condition holds since F^* does not cover edges in $T - T'$ using links that are contained in T'. Hence the algorithm may recurse with $T - T'$.

If T' is not leaf-stem-closed, we need to be more careful. This requires some sophisticated case-analysis, and the details will appear in the full version of the paper.

```
Algorithm Tree-Cover($T$)

1. If $T$ contains a single node, then Return($\emptyset$).
2. Reduce the pair $(T, E)$ to a proper pair.
3. Compute a maximum matching $M$ consisting of leaf-to-leaf links
   that are not twin-links and not locking links. Augment it into $M^+$.
4. Define the set of active nodes $A$ to be the set of unmatched leaves.
5. Initialize the set of candidate links: $J \leftarrow \emptyset$.
6. Apply 1-greedy contractions or 2-greedy contractions to $T/J$ while
   possible. Update $M^+, A$, and $J$ accordingly.
7. Find an active-closed minimal subtree $(T/(J \cup M^+))_v$ of $T/(J \cup M^+)$.
   Let $T'$ denote the subtree $(T/J)_v$.
8. While $T'$ is not leaf-stem closed, do:
   (a) If $T'$ exhibits certain technical conditions, apply case analysis on
       the size and structure of $T'$ (see full version for details).
   (b) Otherwise, cover $T'$ by the basic cover. Formally,
        i. $J \leftarrow J \cup (M^+ \cap T') \cup up(active(T'))$,
       ii. $A \leftarrow A - active(T') + v$, and
      iii. $M^+ \leftarrow M^+ - T'$.
9. Cover $T_v$ and recurse. Formally,
   (a) $J \leftarrow (J \cap T_v) \cup up(active((T/J)_v)) \cup (M^+ \cap (T/J)_v)$, and
   (b) Return($J \cup$ Tree-Cover($T/J$)).
```

References

1. R. Bar-Yehuda, "One for the Price of Two: A Unified Approach for Approximating Covering Problems", Algorithmica 27(2), 2000, 131-144.
2. J. Cheriyan, T. Jordán, and R. Ravi, "On 2-coverings and 2-packing of laminar families", *Lecture Notes in Computer Science*, 1643, Springer Verlag, ESA'99, (1999), 510–520.
3. J. Cheriyan, A. Sebö, and Z. Szigeti, "An improved approximation algorithm for minimum size 2-edge connected spanning subgraphs", *Lecture Notes in Computer Science*, 1412, Springer Verlag, IPCO'98, (1998), 126–136.
4. K. P. Eswaran and R. E. Tarjan, "Augmentation Problems", *SIAM J. Computing*, 5 (1976), 653–665.
5. A. Frank, "Connectivity Augmentation Problems in Network Design", *Mathematical Programming*, State of the Art, Ed. J. R. Birge and K. G. Murty, 1994, 34–63.
6. A. Frank, "Augmenting Graphs to Meet Edge-Connectivity Requirements", *SIAM Journal on Discrete Mathematics*, 5 (1992), 25–53.
7. G. N. Frederickson and J. Jájá, "Approximation algorithms for several graph augmentation problems", *SIAM J. Computing*, 10 (1981), 270–283.

8. M. X. Goemans and D. P. Williamson, "A General Approximation Technique for Constrained Forest Problems", *SIAM J. on Computing*, 24, 1995, 296–317.
9. Kamal Jain, "Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem", FOCS 1998, 448-457.
10. H. Nagamochi, "An approximation for finding a smallest 2-edge connected subgraph containing a specified spanning tree", TR #99019, (1999), Kyoto University, Kyoto, Japan. `http://www.kuamp.kyoto-u.ac.jp/labs/or/members/naga/TC/99019.ps`
11. H. Nagamochi and T. Ibaraki, "An approximation for finding a smallest 2-edge-connected subgraph containing a specified spanning tree", *Lecture Notes In Computer Science*, vol. 1627, Springer-Verlag, 5th Annual International Computing and Combinatorics Conference, July 26-28, Tokyo, Japan, (1999) 31-40.
12. 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.
13. S. Khuller and R. Thurimella, "Approximation algorithms for graph augmentation", *J. of Algorithms*, 14 (1993), 214–225.
14. S. Vempala and A. Vetta, "On the minimum 2-edge connected subgraph", Proc. of the 3rd Workshop on Approximation , Saarbrücken, 2000.