3/2-Approximation for the Forest Augmentation Problem

Ali Çivril*

September 27, 2024

Abstract

We describe a $\frac{3}{2}$-approximation algorithm for the Forest Augmentation Problem (FAP), which is a special case of the Weighted 2-Edge-Connected Spanning Subgraph Problem (Weighted 2-ECSS). This significantly improves upon the previous best ratio 1.9973, and proceeds toward the goal of a $\frac{3}{2}$-approximation algorithm for Weighted 2-ECSS.

1 Introduction

The following is a well-studied problem in network design: Given an undirected simple graph $G = (V, E)$, find a 2-edge-connected spanning subgraph (2-ECSS) of G with minimum number of edges. We denote this problem briefly as 2-ECSS. It remains NP-hard and APX-hard even for subcubic graphs [10]. After a series of improvements beyond the trivial approximation factor 2 [8, 18, 21, 26], the current best approximation factor for the problem is $\frac{4}{3} - \epsilon$ for some constant $\frac{1}{130} > \epsilon > \frac{1}{140}$ [14]. The generalization of this problem in which there is a cost function $c : E \to \mathbb{Q}_{\geq 0}$, which we denote by Weighted 2-ECSS, admits 2-approximation algorithms [19, 21]. Improving this factor has been a major open problem for over three decades. In particular, by a result of Cheriyan et al. [7], the integrality gap of the natural LP relaxation for the problem is lower bounded by $\frac{3}{2}$, and it is likely that there is an approximation algorithm with the same ratio. As a progress towards this goal, intermediate problems between 2-ECSS and Weighted 2-ECSS have received tremendous attention, especially in the last decade. The most general of them is the Forest Augmentation Problem (FAP) in which the cost function is defined as $c : E \to \{0, 1\}$, and the zero-cost edges form a forest. A recent result [16] improves the approximation ratio to 1.9973 for FAP.

As the problem is wide open, even further special cases beyond FAP have been considered. One of them is the Matching Augmentation Problem (MAP) in which the zero-cost edges form a matching. This problem admits approximation ratios $\frac{7}{4}$ [4], $\frac{5}{3}$ [3], and $\frac{13}{8}$ [15]. A further special case is the Tree Augmentation Problem (TAP) in which the zero-cost edges form a tree. Several results with ratios better than 2 have appeared in the literature including [1, 5, 6, 9, 11, 12, 13, 17, 20, 22, 23, 24, 25, 27, 28]. The current best approximation ratio attained is 1.393 [2].

The purpose of this paper is to prove the following theorem via an elegant algorithm. This significantly improves upon the previous best ratio 1.9973, and gives a hint that there might indeed be a $\frac{3}{2}$-approximation algorithm for the more general Weighted 2-ECSS.

Theorem 1. *There exists a polynomial-time $\frac{3}{2}$-approximation algorithm for FAP.*

*Istanbul Atlas University, Computer Engineering Department, Kagithane, 34408 Istanbul, Turkey, e-mail: ali.civril@atlas.edu.tr

2 Preliminaries

We will use the lower bound derived from the dual of the natural LP relaxation for FAP. Here, $\delta(S)$ denotes the set of edges with one end in the cut S and the other not in S.

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c(e)x_e & & \text{(FAP)} \\ \text{subject to} \quad & \sum_{e \in \delta(S)} x_e \geq 2, & \forall\, \emptyset \subset S \subset V, & \\ & 1 \geq x_e \geq 0, & \forall\, e \in E. & \end{aligned}$$

The following is the dual of (FAP).

$$\begin{aligned} \text{maximize} \quad & \sum_{\emptyset \subset S \subset V} 2y_S - \sum_{e \in E} z_e & & \text{(FAP-D)} \\ \text{subject to} \quad & \sum_{S: e \in \delta(S)} y_S \leq c(e) + z_e, & \forall e \in E, & \\ & y_S \geq 0, & \forall\, \emptyset \subset S \subset V, & \\ & z_e \geq 0, & \forall\, e \in E. & \end{aligned}$$

We assume that the input graph G is 2-connected, since the value of an optimal solution for FAP is the sum of those of blocks (maximal 2-connected subgraphs), and one can argue the approximation ratio only within a block. Given a set of edges F, we define $c(F) := \sum_{e \in F} c(e)$. Given a vertex $v \in V$ and a 2-connected spanning subgraph (2-VCSS) of F, if the degree of v in the graph (V, F) is at least 3, it is called a *high-degree vertex* on F. For a path $P = v_1 v_2 \ldots v_{k-1} v_k$, v_1 and v_k are the *end vertices* of P, and all the other vertices are the *internal vertices* of P. A path whose internal vertices are all degree-2 vertices on F is called a *plain path* on F. A maximal plain path is called a *segment*. The length of a segment is the number of edges on the segment. If the length of a segment is ℓ, it is called an *ℓ-segment*. A 1-segment is also called a *trivial segment*. An ℓ-segment with $\ell \geq 2$ is called a *short segment* if $\ell \leq 3$, otherwise a *long segment*. If the removal of a segment from F violates 2-connectivity, it is called a *weak segment* on F, otherwise a *strong segment* on F. A set of edges $H \subseteq F$ is called a *special maximal set* on a 2-VCSS F if it satisfies the following: (1) It consists of zero-cost edges; (2) Contracting the edges in H results in a minimal 2-VCSS; (3) H is maximal.

3 The Algorithm for FAP

Recall that we assume G is 2-connected. The *first step* of the algorithm computes an inclusion-wise minimal 2-VCSS F as follows: Set $F = E$ and perform deletion of edges one by one starting from the unit-cost edges, followed by zero-cost edges, as long as feasibility is maintained. This reverse-delete operation gives priority to the zero-cost edges. In the *second step*, the algorithm first contracts all the zero-cost edges in $E \setminus F$ to form the graph $G' = (V', E')$. The rest of the algorithm is assumed to run on the blocks of G', as we did for G. Given this, F is updated to an inclusion-wise minimal 2-VCSS of G', prioritizing zero-cost edges, exactly as done in the first step. This is followed by contracting a special maximal set on F, thus also updating G'.

In the *third step*, the algorithm recursively modifies the running solution F via *improvement processes*. Given a strong 2-segment S on F and its internal vertex u, let $N_E(u)$ denote the set of

Algorithm 1: FAP($G(V, E)$)

// First step

Let F be an inclusion-wise minimal 2-VCSS of G, which prioritizes zero-cost edges over unit-cost edges

// Second step

Contract all the zero-cost edges in $E \setminus F$ to obtain $G' = (V', E')$

Let F be an inclusion-wise minimal 2-VCSS of G', which prioritizes zero-cost edges over unit-cost edges

Contract a special maximal set on F, also updating G'

// Third step: Improvement operations

while *there is a strong* 2*-segment* S *on* F *such that no improvement process has been called on its internal vertex* u **do**

  Improvement-Process(G', F, S, u)

return F

Algorithm 2: Improvement-Process(G', F, S, u)

if *there is an improvement operation that can be performed on* u **then**

  Apply the improvement operation on u

  return

for *each critical edge set* A *incident to* u **do**

  Let $\mathcal{S}$ be the set of strong 2-segments on $F \cup A$ that do not exist on F

  for *each strong* 2*-segment* T *in* $\mathcal{S}$ *and the internal vertex* v *of* T **do**

    if *no improvement process has been called on* (T, v) **then**

      Improvement-Process($G', F \cup A, T, v$)

      if *there is an improvement operation performed in* Improvement-Process($G', F \cup A, T, v$) **then**

        Perform deletion operation on $F \cup A$ in the order F, A

        return

if *there is no improvement operation performed in any of the recursive calls above* **then**

  Restore F to the original set considered before the function call

edges incident to u in E. An improvement process first tries to replace F by $(F \setminus B) \cup A$ while maintaining feasibility, where $A \subseteq E \setminus F$ is a set of k edges called a *critical edge set*, and $B \subseteq F$ is a set of $k+1$ edges, $1 \leq k \leq 2$. We seek such A to be a subset of $N_{E \setminus F}(u)$. If there is such A, the described operation is called an *improvement operation*. Two improvement operations and the corresponding critical edge sets are given in Figure 1 and Figure 2, where all the included and the excluded edges are of unit-cost.

If no improvement operation can be performed, fixing a critical edge set A incident to u, the algorithm checks if $F \cup A$ contains new strong 2-segments that do not exist on F. If it does, it calls the procedure described above for S and u *recursively* on the internal vertices of the newly appearing strong 2-segments provided that no improvement process has been previously called on the internal vertex of a given segment. These calls are performed for all A on u and for each $u \in S$. If there is an improvement operation in one of the recursive calls, the called function returns

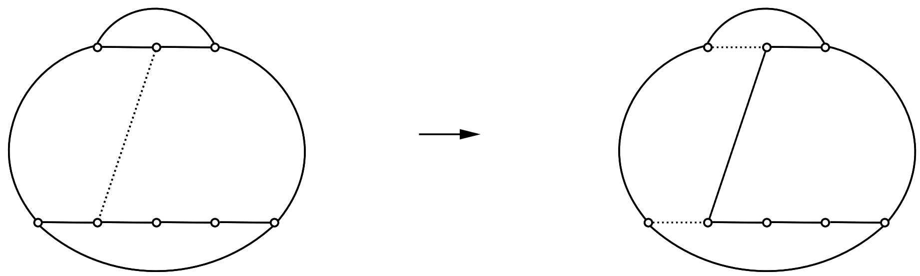


Figure 1: An example of an improvement operation in which the critical edge set is shown in dotted lines on the left

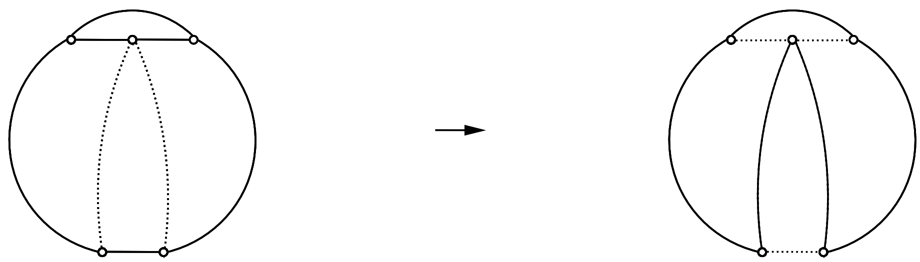


Figure 2: An example of an improvement operation in which the critical edge set is shown in dotted lines on the left

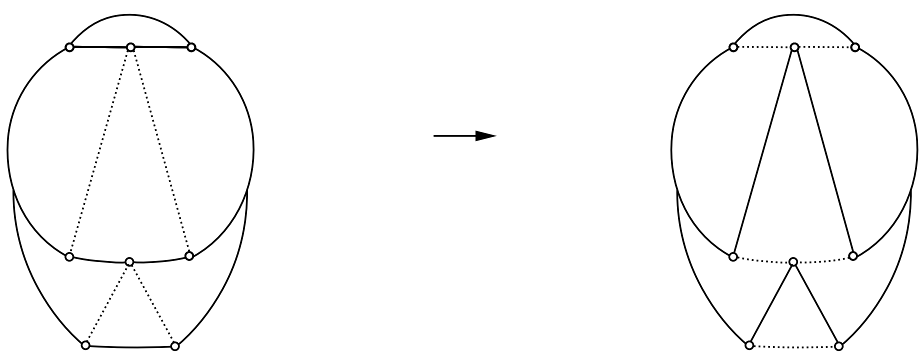


Figure 3: An example of an improvement process of recursion depth 2

and the caller performs a specific reverse-delete operation by attempting to delete the edges from $F \cup A$ in the order F, A, while maintaining feasibility. This enforces to keep the edges in A in the solution. An examples of this operation is given in Figure 3, where all the included and excluded edges are of unit-cost, and the depth of the recursion tree is 2 After the reverse-delete operation, the current function call returns. If after all the recursive calls from u there is no improvement operation performed, the solution F is restored back to the original one before the function call on u. The main iterations continue until there is no S and u on which we can perform an improvement process.

Proposition 2. *Algorithm 1 can be implemented in polynomial-time.*

Proof. It is clear that the first and the second step of the algorithm can be implemented in polynomial-time. To see that the third step also takes polynomial time, it suffices to see that the main loop of Improvement-Process terminates in polynomial number of operations. There are polynomially many critical edge sets A, since $|A|$ is constant. Starting from the internal vertex

u of a strong 2-segment S, consider the recursion tree in which each node represents a recursive function call. By definition, each node of this tree is associated to the internal vertex of a strong 2-segment. A vertex can be the internal vertex of a single strong 2-segment. This implies that the number of nodes in the tree is polynomially bounded. So the main loop of IMPROVEMENT-PROCESS terminates in polynomial number of operations. □

4 Proof of Theorem 1

Let $opt(G)$ denote the value of an optimal 2-ECSS on G, and $opt(G')$ denote the value of an optimal 2-ECSS on $G' = (V', E')$, the result of the second step of the algorithm. Let F be a solution returned by Algorithm 1.

Lemma 3. *$opt(G) \geq opt(G')$.*

Proof. Take an optimal 2-ECSS O on G. Let O' be the intersection of O with all the zero-cost edges contracted in the second step of the algorithm. Then O' is a feasible solution for G', which implies the result. □

Lemma 4. *There exists a graph G_1 and a 2-VCSS $F_1 \subseteq E(G_1)$ such that the following hold:*

1. *Given the internal vertex s of a strong 2-segment on F_1, there is no edge $e \in E(G_1) \setminus F_1$ incident to s.*
2. *F_1 is minimal with respect to inclusion.*
3. $\frac{c(F_1)}{opt(G_1)} \leq \frac{3}{2} \Rightarrow \frac{c(F)}{opt(G')} \leq \frac{3}{2}$.

Proof. We reduce G' to G_1 and F to F_1 by performing a series of operations. Let S be a strong 2-segment on F, and s be its internal vertex. Let O be an optimal 2-ECSS on G'. Then O contains two edges incident to s, say e_1 and e_2. Assume it contains a third edge e_3 incident to s. Let the other end vertices of these edges be w_1, w_2, and w_3, respectively. If O contains all the edges incident to w_i that are in F, we call w_i a *special vertex*, for $i = 1, 2, 3$. Note that none of e_1, e_2, and e_3 is a zero-cost edge by the construction of G'. To finish the proof of the lemma, we need the following two claims.

Claim 5. *We can switch to an optimal solution O such that there is at most one special vertex in the set $\{w_1, w_2, w_3\}$.*

Proof. Assume without loss of generality that w_1 and w_2 are special vertices. Then by the structure of a 2-ECSS, we can discard e_1 or e_2 from O without violating feasibility. □

Claim 6. *There exists an optimal 2-ECSS O' on G' such that O' contains 2 edges incident to s.*

Proof. By Claim 5, there are at least two vertices in the set $\{w_1, w_2, w_3\}$ that are not special. Let two of them be without loss of generality w_2 and w_3. By the structure of a 2-ECSS, one of these vertices, say w_2, satisfies the following. There is a neighbor w_2' of w_2 such that $f = (w_2, w_2') \in F \setminus O$, and $O' = O \cup \{f\} \setminus \{e_2\}$ is another optimal solution. In this case the degree of s on O' is 2, which completes the proof. □

Let $O'(S)$ be the set of edges in this solution incident to the internal vertex of S. Let $F' = F \cup O'(S) \setminus P$ be a minimal 2-VCSS on G, where $P \subseteq F \setminus O'(S)$. Let $E(S)$ denote the set of edges incident to the internal vertex of S on $E(G')$, which excludes the edges in $O'(S)$, and note that it contains P. Delete the edges in $E(S)$ from G' to obtain G''. Perform these operations, including the switch to an optimal solution implied by Claim 6, recursively on the new strong 2-segments that appear on F', which we call *emerging segments*. Note that since none of the aforementioned vertices w_1, w_2, and w_3 can be the internal vertex of a strong 2-segment due to an improvement operation, the switch from O to O' cannot be reversed. After the recursion starting from S terminates, continue performing the described operations on the strong 2-segments on the residual solution and the graph. Let the results be F_1 and G_1. Note that the first claim of the lemma also holds, since there is no edge in $E(G_1) \setminus F_1$ incident to the internal vertex of a strong 2-segment on F_1 by construction. The second claim of the lemma follows from Claim 6. We now show that the third claim holds.

Claim 7. *$c(F_1) \geq c(F)$.*

Proof. Let S be a strong 2-segment on which we start the recursive operations above or an emerging segment. The inequality $c(P) > c(O'(S))$ derives a contradiction to the algorithm and the construction of F_1, since there is no improvement process performed on S that has improved the cost of the solution. In particular, by all the listed improvement operations we cannot have the configurations on the left hand sides of Figure 1-Figure 3. We thus have $c(P) \leq c(O'(S))$, which implies $c(F_1) \geq c(F)$. □

We next note that $opt(G_1) \leq opt(G')$. This follows from our construction ensuring that there is an optimal solution O such that for any strong 2-segment S on F_1, $E(S)$ does not contain any edge from O. Combining this with Claim 7, we obtain $\frac{c(F_1)}{opt(G_1)} \geq \frac{c(F)}{opt(G')}$, which implies the third claim of the lemma, and completes the proof. □

Let F_0 be the result of the first step of the algorithm, and F_0' be the result of the second step of the algorithm. Let G_1 and F_1 be as implied by Lemma 4.

Lemma 8. *Let u be the internal vertex of a strong 2-segment on F_0. Then there is no zero-cost edge $e \in E \setminus F_0$.*

Proof. The existence of such an edge e contradicts the reverse-delete operation in constructing F_0, which gives precedence to zero-cost edges. In particular, this operation must include e into the solution. □

Lemma 9. *There is no self-loop $e \in F$ with $c(e) = 1$.*

Proof. Let $e \in F$ be a self-loop on $u \in V'$ with $c(e) = 1$. Let $N_{E\setminus F_0}(u)$ be the set satisfying the following: (1) Its elements belong to the (non-empty) set of zero-cost edges in $N_{E\setminus F_0}(u)$ contracted in the second step of the algorithm to obtain $G' = (V', E')$; (2) Its elements are incident to the vertices in V that are identified by $u \in V'$. Then $(F_0 \setminus \{e\}) \cup N_{E\setminus F_0}(u)$ is feasible, and hence $N_{E\setminus F_0}(u)$ must have been selected in the first step of the algorithm, which prioritizes zero-cost edges. This however excludes e from the solution, deriving a contradiction. □

Lemma 10. *E' either consists of a single double-edge e with $c(e) = 1$, or there is no double-edge $e \in E'$ with $c(e) = 1$.*

Proof. Recall that we assume $G' = (V', E')$ is 2-connected. Suppose E' is not equal to a set consisting of a single double-edge. Let $e = (u, v) \in E'$ be a double-edge with $c(e) = 1$. Let $N_{E \setminus F_0}(u)$ be the set satisfying the following: (1) Its elements belong to the (non-empty) set of zero-cost edges in $N_{E \setminus F_0}(u)$ contracted in the second step of the algorithm; (2) Its elements are incident to the vertices in V that are identified by $u \in V'$. Then the first step of the algorithm, which prioritizes zero-cost edges, keeps $N_{E \setminus F_0}(u) \cup N_{E \setminus F_0}(v)$ in the solution, and thus excludes one of the edges defining e. This derives a contradiction. $\square$

Lemma 11. *Let F' be the union of F and the set of contracted edges in the second step of the algorithm. Then F' is a feasible solution for G.*

Proof. The statement is clear if E' (hence F) consists of a single double-edge. Otherwise, we have by Lemma 10 that there is no double edge in E'. Given this, take an edge $e = (u, v)$ contracted in the second step of the algorithm. By Lemma 8, neither u nor v is an internal vertex of a strong 2-segment on F_0. Thus, neither u nor v is a degree-1 vertex on the union of F and the contracted edges. The result then follows by Lemma 9. $\square$

Lemma 12. *There is no zero-cost edge in $E' \setminus F'_0$.*

Proof. By the contractions performed in the second step of the algorithm, a zero-cost edge $f \in E' \setminus F'_0$ must belong to F_0. But f remains in F'_0 by the reverse-delete operation in computing F'_0, which prioritizes zero-cost edges. Thus, there is no zero-cost edge $f \in E' \setminus F'_0$. $\square$

Lemma 13. *There is no zero-cost edge in $E' \setminus F$.*

Proof. The result follows by Lemma 12 and the fact that the third step of the algorithm never excludes a zero-cost edge from F'_0. $\square$

Lemma 14. *There is no zero-cost edge in $E(G_1) \setminus F_1$.*

Proof. By Lemma 13, there is no zero-cost edge in $E' \setminus F$. The result follows from the fact that the reduction described in Lemma 4 does not introduce a zero-cost edge in $E(G_1) \setminus F_1$. $\square$

Lemma 15. *If S is a weak segment or a strong ℓ-segment on F_1 with $\ell \geq 3$, then S does not have a zero-cost edge.*

Proof. Recall by Lemma 12 that there is no zero-cost edge in $E' \setminus F'_0$. Thus, the third step of the algorithm does not include any zero-cost edge. Similarly, the reduction described in Lemma 4 does not introduce a zero-cost edge in $E(G_1) \setminus F_1$. These imply that it suffices to show the statement for such an S on F'_0. Assuming however that there is such a segment with a zero-cost edge contradicts the second step of the algorithm, which contracts a special maximal set. In particular, we have the following cases regarding the zero-cost segments of S:

- If S is a weak segment, then contracting all its zero-cost edges results in a minimal 2-VCSS.
- If S is a strong segment with at least two unit-cost edges, then contracting all its zero-cost edges results in a minimal 2-VCSS.
- If S is a strong segment with one unit-cost edge, then contracting all its zero-cost edges except one results in a minimal 2-VCSS.

$\square$

Lemma 16.

$$\frac{c(F_1)}{opt(G_1)} \leq \frac{3}{2}.$$

Proof. We construct a feasible dual solution in (FAP-D) with total value at least $\frac{2}{3}c(F_1)$. Given an internal vertex u of a strong 2-segment on F_1, we assign $y_{\{u\}} = 1$. Recall that at most one of the edges of a 2-segment can be of zero-cost. In order to maintain feasibility, if such a segment has a zero-cost edge e, we set $z_e = 1$. For any internal vertex v of a weak segment or a strong ℓ-segment on F_1 with $\ell \geq 3$, we assign $y_{\{v\}} = 1/2$. Recall that by Lemma 15, such a segment does not have a zero-cost edge. These assignments form a feasible solution in (FAP-D) by Lemma 14, Lemma 15, and the first claim of Lemma 4. We will mostly be tacitly assuming these facts in the rest of the proof while enlarging the dual assignment.

We distinguish a dual value we assign and its contribution in the objective function of (FAp-D), which is twice the dual value. The latter is called the *dual contribution*. We use a cost sharing argument, so that the cost of a specific set of edges is countered with a unique set of dual contributions with ratio at least $\frac{2}{3}$, which establishes the main result. In doing so, we will also make sure that we count all the z dual values of edges exactly once. For a given specific set of edges, we call the ratio of the dual value with the cost the *cover ratio*. If the cover ratio is 1, we say that the set is *optimally covered*.

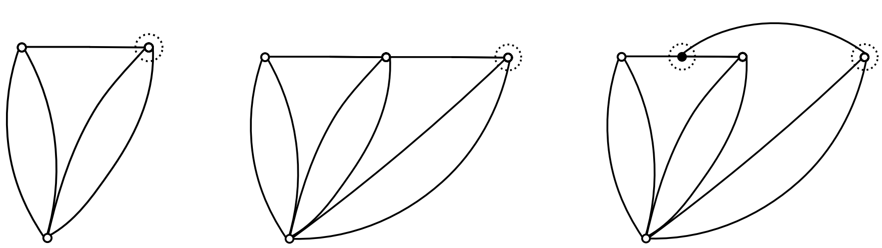
We first describe the argument for the strong segments. Given a strong segment S on F_1 and an internal vertex s of S:

- If S is a 2-segment and both of its edges are of unit-cost, the dual contribution 2 of $y_{\{s\}}$ results in a cover ratio of 1. If one of the edges is of unit-cost, and the other edge e is a zero-cost edge, then $2y_{\{s\}} - z_e = 1$, again optimally covering the edges of the segment.

- If S is an ℓ-segment with $\ell \geq 3$, then S does not have a zero-cost edge by Lemma 15. Then the dual contribution of the internal vertices of S is $\ell - 1$, which results in a cover ratio $\frac{\ell-1}{\ell} \geq \frac{2}{3}$.

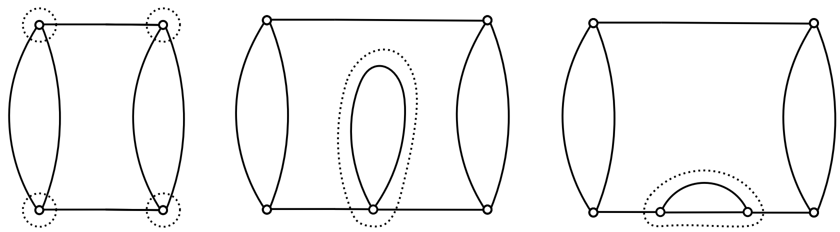
Given a weak ℓ-segment, the dual contribution of its internal vertices is also $\ell - 1$. We impose that this contribution pays for the cost $\ell - 1$ of the weak segment, thus covering the cost $\ell - 1$ of the segment optimally. There remains the cost 1 of each weak segment to be covered. Since the cover ratio of the strong segments is at least $\frac{2}{3}$ as shown above, it suffices to show that the remaining cost 1 of each weak segment is covered with cover ratio at least $\frac{2}{3}$.

We argue by induction on the number of weak segments k. We first consider one of the base cases $k = 1$. Let u be an end vertex of the weak segment. If u is not shared by any strong short segment, define $y_{\{u\}} = 1/2$, which optimally covers the weak segment (See Figure 4a for an illustration). Otherwise, let v be the other end vertex of the weak segment. Then we have at least one strong short segment incident to u and at least one strong short segment incident to v. We now incorporate the strong short segments into the analysis. Recall that the cost of a strong short



(a) (b) (c)

Figure 4



(a) **(b)** **(c)**

Figure 5

segment is at least 1. So, the cost together with that of the weak segment is $\ell + 1$, and total dual contribution is at least ℓ. This yields a cover ratio of at least $\frac{\ell}{\ell+1} \geq \frac{2}{3}$, since $\ell \geq 2$.

We next consider the other base case $k = 2$ in which the two weak segments do not share a common end vertex. Let u be an end vertex of a weak segment. If u is not shared by any strong 2-segment, we assign $y_{\{u\}} = 1/2$. Figure 5a shows this configuration with all the end vertices applied this assignment. Otherwise, let S be a set satisfying the following:

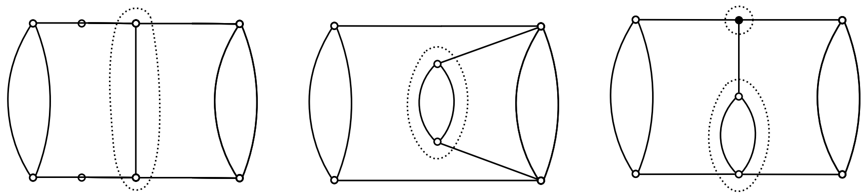
1. $u \in S$,
2. S consists of vertices of strong 2-segments,
3. S induces a connected subgraph,
4. S is maximal.

We call S an *augmented set on* u. Assign $y_S = 1/2$. Note that this is also feasible by the stated properties of S and the first claim of Lemma 4. We call the dual variables $y_{\{u\}}$ and y_S *augmented dual variables*. It is clear that in the base case, there are at least 2 augmented dual variables, thereby optimally covering the weak segments.

In the inductive step one may introduce one, two, three, or four new weak segments by extending the graph in the induction hypothesis. All the cases are given in Figure 5, Figure 6, and Figure 7, where we depict the extending subgraphs in their simplest form. In Figure 5b and Figure 5c one new weak segment is introduced. Let u be a newly introduced high-degree vertex. If there is no strong 2-segment with an end vertex u, we define $y_{\{u\}} = 1/2$. Otherwise, we define $y_S = 1/2$ for an augmented set S on u. In either case, the new weak segment is optimally covered.

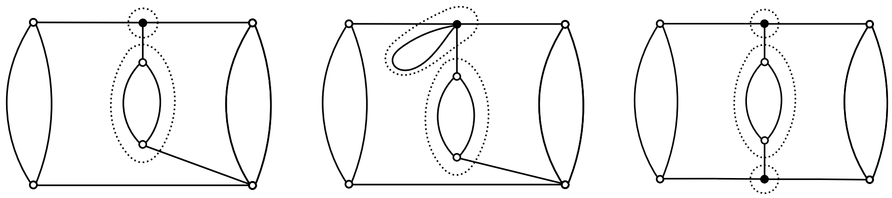
In Figure 6a two new weak segments are introduced. Let u and v be two newly introduced high-degree vertices, which are also end vertices of weak segments. If neither u nor v is an end vertex of a strong 2-segment, we define $y_{\{u\}} = y_{\{v\}} = 1/2$, which optimally covers the new weak segments. Next, assume without loss of generality that both of them are the end vertices of the same strong 2-segment. Then assign $y_S = 1/2$ for an augmented set S on u. In this case we incorporate the 2-segment into the analysis. Recall that at most one edge of a strong 2-segment can be of zero-cost. The total cost of the 2-segment and the new weak segments is then at least $\ell + 2$, where $\ell \geq 1$, and the total dual contribution including y_S and the dual value defined for the 2-segment is $\ell + 1$. This leads to a cover ratio of $\frac{\ell+1}{\ell+2} \geq \frac{2}{3}$. We do not depict the generalization of Figure 6a, analogous to the one from Figure 5b to Figure 5c, which does not change the analysis.

In Figure 6b two new weak segments are introduced together with at least two strong segments. The analysis is identical to that of Figure 6a, resulting in a cover ratio of at least $\frac{2}{3}$. In Figure 6c, Figure 7a, and Figure 7b three new weak segments are introduced. In all these cases the analysis essentially reduces to that of the previous case, since there is a high-degree vertex we can assign the dual value 1/2 on, which optimally covers the extra new segment, and hence only betters the cover



(a) (b) (c)

Figure 6



(a) (b) (c)

Figure 7

ratio of the previous case. Figure 7c is also a straightforward generalization of Figure 7a, where there are four new weak segments. We do not depict the generalizations of Figure 7c, analogous to the one from Figure 7a to Figure 7b, which does not change the analysis. This completes the induction and the proof. $\square$

By Lemma 16 and the third claim of Lemma 4, we have $\frac{c(F)}{opt(G')} \leq \frac{3}{2}$. Let F' be the solution implied by Lemma 11, so that $\frac{c(F')}{opt(G')} \leq \frac{3}{2}$. Combining this with Lemma 3, we have $\frac{c(F')}{opt(G)} \leq \frac{3}{2}$. Together with Proposition 2, this completes the proof of Theorem 1.

5 A Tight Example

A tight example for the algorithm is given in Figure 8. The bold lines represent the edges of cost 1, and the other lines represent zero-cost edges. The solution returned by the algorithm has cost $3k - 2$, where k is an even integer. Note that there is no improvement that can be performed on the depicted solution. The optimal solution has cost $2k$.

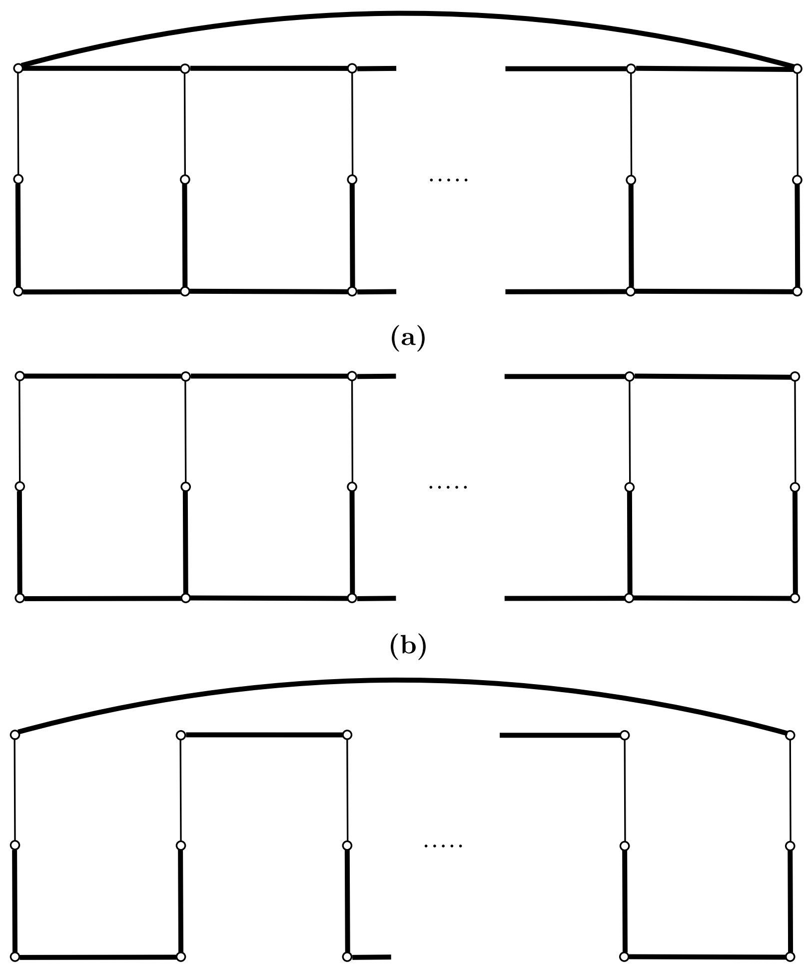
Acknowledgment

This work was supported by Scientific and Technological Research Council of Turkey (TUBITAK) under the Grant Number 123E530. The author thanks to TUBITAK for their supports.

References

[1] D. Adjiashvili. Beating approximation factor two for weighted tree augmentation with bounded costs. *ACM Trans. Algorithms*, 15(2):19:1–19:26, 2019.

[2] F. Cecchetto, V. Traub, and R. Zenklusen. Bridging the gap between tree and connectivity augmentation: unified and stronger approaches. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 370–383, 2021.



(a) **(b)** **(c)**

Figure 8: A tight example for the algorithm: (a) Input graph; (b) A solution returned by the algorithm; (c) An optimal solution. The bold lines represent the edges of cost 1, the other lines represent zero-cost edges.

[3] J. Cheriyan, R. Cummings, J. Dippel, and J. Zhu. An improved approximation algorithm for the matching augmentation problem. *SIAM J. Discret. Math.*, 37(1):163–190, 2023.

[4] J. Cheriyan, J. Dippel, F. Grandoni, A. Khan, and V. V. Narayan. The matching augmentation problem: a $\frac{7}{4}$-approximation algorithm. *Math. Program.*, 182(1):315–354, 2020.

[5] J. Cheriyan and Z. Gao. Approximating (unweighted) tree augmentation via lift-and-project, part I: stemless TAP. *Algorithmica*, 80(2):530–559, 2018.

[6] J. Cheriyan and Z. Gao. Approximating (unweighted) tree augmentation via lift-and-project, part II. *Algorithmica*, 80(2):608–651, 2018.

[7] J. Cheriyan, H. J. Karloff, R. Khandekar, and J. Könemann. On the integrality ratio for tree augmentation. *Oper. Res. Lett.*, 36(4):399–401, 2008.

[8] J. Cheriyan, A. Sebö, and Z. Szigeti. Improving on the 1.5-approximation of a smallest 2-edge connected spanning subgraph. *SIAM J. Discrete Math.*, 14(2):170–180, 2001.

[9] N. Cohen and Z. Nutov. A (1+ln2)-approximation algorithm for minimum-cost 2-edge-connectivity augmentation of trees with constant radius. *Theor. Comput. Sci.*, 489-490:67–74, 2013.

[10] B. Csaba, M. Karpinski, and P. Krysta. Approximability of dense and sparse instances of minimum 2-connectivity, TSP and path problems. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 74–83, 2002.

[11] G. Even, J. Feldman, G. Kortsarz, and Z. Nutov. A 1.8 approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Trans. Algorithms*, 5(2):21:1–21:17, 2009.

[12] S. Fiorini, M. Groß, J. Könemann, and L. Sanità. Approximating weighted tree augmentation via chvátal-gomory cuts. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 817–831, 2018.

[13] G. N. Frederickson and J. F. JáJá. Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2):270–283, 1981.

[14] M. Garg, F. Grandoni, and A. J. Ameli. Improved approximation for two-edge-connectivity. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2368–2410, 2023.

[15] M. Garg, F. Hommelsheim, and N. Megow. Matching augmentation via simultaneous contractions. In *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023*, volume 261 of *LIPIcs*, pages 65:1–65:17.

[16] F. Grandoni, A. J. Ameli, and V. Traub. Breaching the 2-approximation barrier for the forest augmentation problem. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1598–1611, 2022.

[17] F. Grandoni, C. Kalaitzis, and R. Zenklusen. Improved approximation for tree augmentation: saving by rewiring. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 632–645, 2018.

[18] C. Hunkenschröder, S. Vempala, and A. Vetta. A 4/3-approximation algorithm for the minimum 2-edge connected subgraph problem. *ACM Trans. Algorithms*, 15(4):55:1–55:28, 2019.

[19] K. Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[20] S. Khuller and R. Thurimella. Approximation algorithms for graph augmentation. *J. Algorithms*, 14(2):214–225, 1993.

[21] S. Khuller and U. Vishkin. Biconnectivity approximations and graph carvings. *J. ACM*, 41(2):214–235, 1994.

[22] G. Kortsarz and Z. Nutov. A simplified 1.5-approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Trans. Algorithms*, 12(2):23:1–23:20, 2016.

[23] G. Kortsarz and Z. Nutov. LP-relaxations for tree augmentation. *Discret. Appl. Math.*, 239:94–105, 2018.

[24] H. Nagamochi. An approximation for finding a smallest 2-edge-connected subgraph containing a specified spanning tree. *Discret. Appl. Math.*, 126(1):83–113, 2003.

[25] Z. Nutov. On the tree augmentation problem. *Algorithmica*, 83(2):553–575, 2021.

[26] A. Sebö and J. Vygen. Shorter tours by nicer ears: 7/5-approximation for the graph-TSP, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *Combinatorica*, 34(5):597–629, 2014.

[27] V. Traub and R. Zenklusen. A better-than-2 approximation for weighted tree augmentation. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 1–12, 2021.

[28] V. Traub and R. Zenklusen. Local search for weighted tree augmentation and steiner tree. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 3253–3272, 2022.