Operations Research Letters 50 (2022) 693–698

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Coloring down: 3/2-approximation for special cases of the weighted tree augmentation problem

Jennifer Iglesias [a], R. Ravi [b,*]

[a] Waymo Inc., 1600 Amphitheatre Pkwy, Mountain View, 94043, CA, USA
[b] Tepper School of Business, Carnegie Mellon University, 15213, Pittsburgh, PA, USA

ARTICLE INFO

Article history:
Received 27 July 2021
Received in revised form 11 October 2022
Accepted 12 October 2022
Available online 26 October 2022

Keywords:
Approximation algorithm
Network design
Integrality gap

ABSTRACT

In this paper, we investigate the weighted tree augmentation problem (TAP), where the goal is to augment a tree with a minimum cost set of edges such that the graph becomes two edge connected. First we show that in weighted TAP, we can restrict our attention to trees which are binary and where all the non-tree edges go between two leaves of the tree. We then give a top-down coloring algorithm that differs from known techniques for approximating TAP.

The algorithm we describe always gives a 2-approximation starting from any feasible fractional solution to the natural tree cut covering LP. When the structure of the fractional solution is such that all the edges with non-zero weight are at least α, then this algorithm achieves a $\frac{2}{1+\alpha}$-approximation.

We also investigate a variant of TAP where every tree edge must belong to a cycle of length three (triangle) in the solution. We give a $\Theta(\log n)$-approximation algorithm for this problem in the weighted case in n-node graphs and a 4-approximation in the unweighted case.

1. Introduction

We consider the *weighted tree augmentation problem (TAP)*: Given an undirected graph $G = (V, E)$ with non-negative weights c on the edges, and a spanning tree T, find a minimum cost subset of edges $A \subseteq E(G) \setminus E(T)$ such that $(V, E(T) \cup A)$ is two-edge-connected. We will refer to the elements of $E(T)$ as (tree) edges and those of $E(L) = E(G) \setminus E(T)$ as *links* for convenience. A graph is *two-edge connected* if the removal of any edge does not disconnect the graph, i.e., it does not have any cut edges. Since cut edges are also sometimes called bridges, this problem has also been called *bridge connectivity augmentation* in prior work [12].

While TAP is well studied in both the weighted and unweighted case [12,18,22,9,5,19,1,11], it is NP-hard even when the tree has diameter 4 [12] or when the set of available links form a single cycle on the leaves of the tree T [7]. Weighted TAP was one of the simplest network design problems without a better than 2-approximation in the case of general (unbounded) link costs and arbitrary depth trees, until recently [24,23].

1.1. LP relaxations

TAP can also be viewed as a covering problem. The cuts in a tree which have a single tree edge crossing them are exactly the cuts that must be covered. A link ℓ is said to *cover* an edge e if the unique cycle of $\ell + T$ contains e. By rooting the tree at an arbitrary node, we define S_e be the node set of the subtree under the tree edge e. We use $\delta_L(S)$ and $\delta_T(S)$ for $S \subset V$ to denote the set of links and tree edges with exactly one endpoint in S respectively. Then $\delta_L(S_e)$ for a tree edge e denotes the set of links which cover e. The natural covering linear programming relaxation for the problem, EDGE-LP, is a special instance of a set covering problem with one requirement (element) corresponding to each cut edge in the tree (since the sets S_e form a laminar family, this is also equivalent to a laminar cover problem [7]).

$$\min \sum_{\ell \in E} c_\ell x_\ell$$

$$x(\delta_L(S_e)) \geq 1 \qquad \forall e \in E(T) \tag{1}$$

$$x_\ell \geq 0 \qquad \forall \ell \in E(L) \tag{2}$$

Fredrickson and Jájá showed that the integrality gap of the EDGE-LP can not exceed 2 [12] and also studied the related problem of augmenting the tree to be two-node-connected (bicon-

* Corresponding author.
E-mail address: ravi@andrew.cmu.edu (R. Ravi).

https://doi.org/10.1016/j.orl.2022.10.007

nectivity versus bridge-connectivity augmentation) [13]. Cheriyan, Jordán, and Ravi, who studied half-integral solutions to EDGE-LP and proved an integrality gap of $\frac{4}{3}$ for such solutions, also conjectured that the overall integrality gap of EDGE-LP was at most $\frac{4}{3}$ [7]. However, Cheriyan et al. [8] demonstrated an instance for which the integrality gap of the EDGE-LP is at least 3/2.

1.2. Related work

Weighted TAP has several 2-approximation algorithms. Frederickson and JáJá [12] convert the problem into one of finding a minimum weight arborescence in an appropriate directed graph. Khuller and Thurimella improved the runtime of this algorithm [18]. Later, other 2-approximation algorithms have been devised for weighted TAP using other techniques such as the primal-dual method [22] and iterative rounding [17].

Special cases of weighted TAP have also been investigated. Cheriyan, Jordán and Ravi [7] developed a $\frac{4}{3}$-approximation for TAP when the optimal fractional solution is half-integral. Another special case of weighted TAP is when the tree has bounded depth. In this special case, Cohen and Nutov showed there exists a $(1+\ln 2)$-approximation [9]. Very recently, this approach has been extended to provide an approximation to the general case of the problem with the same performance guarantee by Traub and Zenklusen [24]. A follow-up paper by the same authors [23] improved the approximation ratio to nearly 1.5. However, this work does not provide any new results on the integrality gap of any of the LP relaxations above.

Adjiashvili [1] showed a 1.96-approximation for another special case of weighted TAP where all link weights are between 1 and some constant M by using a bundling type linear program. Building on this work, Fiorini et al. [11] generalized the constraints from [20] and combined them with the bundle constraints from [1] to propose a new ODD-LP and achieved a $\frac{3}{2}+\epsilon$ approximation for the same special case (when all the costs are between 1 and some constant M). Another recent paper by Nutov takes a subset of Adjiashvili's constraints and achieves a $\frac{12}{7}+\epsilon$ approximation when all the costs are in the range $[1, O(\log n)]$ where n is the number of nodes in the tree [21]. A line of work leading to Cecchetto et al. [4] also supply better than 1.5 approximations for the unweighted case of TAP. Many of these techniques rely heavily on the bundle constraints that are focused on link weights being in a bounded range; hence they do not seem to be generalizable to the case of arbitrary weights. We believe the general problem requires a more polyhedral approach of the type we investigate.

Numerous papers attempted to reach a target $\frac{3}{2}$-approximation in the unweighted case of TAP when all links have the same weight. One paper by Kortsarz and Nutov [20] presents a new linear program with a 1.75-approximation for the unweighted case, in the hope that this linear program could help break the 2-approximation barrier for the weighted case. This LP used properties of an optimal solution for the unweighted case to add multiple new constraints; In retrospect, these additional constraints are all included in the ODD-LP. Two papers achieved a $\frac{3}{2}$-approximation for the unweighted case using the same algorithm but with very different analyses. The earlier paper by Kortsarz and Nutov [19] extends the approach of [10] that relies on a unique token giving argument. Later work by Cheriyan and Gao [5,6] shows an integrality gap of $\frac{3}{2}$ of a related semidefinite program. While both of these approaches to proving the gap are different, they still heavily rely on the fact that all the links have the same weight. For unweighted TAP, Nutov [21] showed that the integrality gap of the EGDE-LP is at most 28/15, and that of ODD-LP is at most 7/4.

1.3. Our results

Our results give new approaches to determine the integrality gap of weighted TAP: our methods provide constructive proofs of convex decompositions of given fractional solutions appropriately scaled into integer solutions.

1. We show that any instance of weighted TAP can be reduced to equivalent instances where the underlying tree is binary and all the links have their endpoints at leaves (Theorem 2.1 in Section 2). While the reduction to leaf-to-leaf instances was known before, the reduction to binary trees is new. The simpler structure of input instances help us in several of our proofs and may be useful in future approaches to settle the integrality gap of the standard EDGE-LP relaxation of weighted TAP.
2. We give a simple new top-down coloring algorithm that gives a constructive proof of the integrality gap of 2 for EDGE-LP by providing a convex decomposition. Furthermore, if the minimum non-zero value in the solution for any link is at least α then we can achieve an improved $\frac{2}{1+\alpha}$-approximation (Theorem 3.1 in Section 3). This generalizes the result of Cheriyan et al. [7] which we can recover by setting $\alpha=\frac{1}{2}$. Even more interestingly, this provides a new $\frac{3}{2}$-approximation when all nonzero values in the solution are at least $\frac{1}{3}$.
3. In Section 4, we provide a complete study of 3TAP in which every tree edge must be in a triangle in the final two-connected augmentation of the input tree. Via a reduction from set cover, we show an $\Omega(\log n)$-inapproximability result and give a matching approximation algorithm. In the unweighted case, we show that any minimal solution gives a 4-approximation.

Our approach is a top-down coloring algorithm on the scaled fractional solution where each color class is a feasible solution. In particular, $\frac{3}{2}$ times the fractional solution is decomposed into a convex combination of integer solutions. This provides not only an approximation algorithm but also directly proves the integrality gaps for the corresponding covering LP [3]. In addition, this technique of top-down coloring differs from all current $\frac{3}{2}$-approximation algorithms on unweighted TAP and all current algorithms which achieve better than 2-approximations for special cases of weighted TAP. Since our methods decompose scaled fractional solutions, they also have the potential to extend to give tight integrality gap proofs.

2. Reduction to binary trees and a stronger LP

In this section, we show that we can restrict our attention to only certain instances of weighted TAP. Our reduction restricts not only the structure of the links but also the structure of the resulting tree. Our reduction converts the tree to be covered to a binary tree. Using this feature, we extend the EDGE-LP constraint from a single tree edge to all odd cuts in the resulting binary tree and show its validity.

2.1. Reduction to binary trees

Theorem 2.1. *Any instance of weighted TAP (T, c, L) can be reduced to a corresponding instance of weighted TAP (T', c', L') where the tree T' is binary and all the links in L' go between two leaves. In addition, every feasible solution to (T, c, L) provides a feasible solution to (T', c', L') of equal cost and vice versa. If the number of nodes in T is n of which l are leaves, the number of nodes in T' is $4n-2l-1$.*

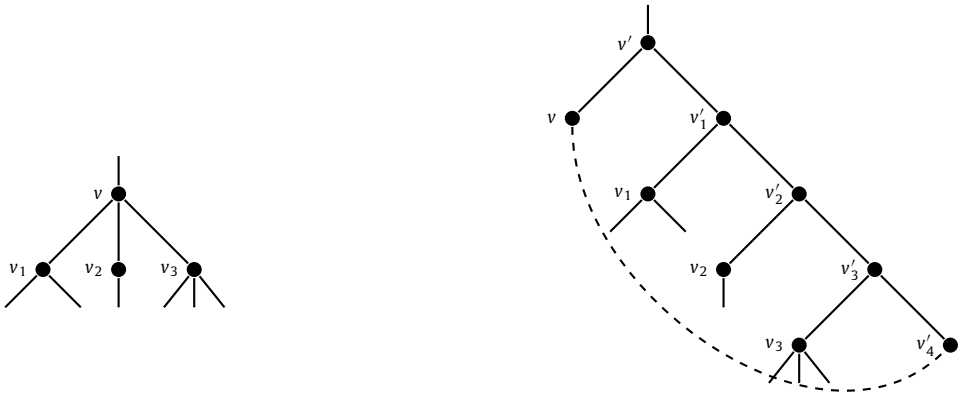

Fig. 1. An example of an internal node v with three children before and after the transformation at v.

Proof. The construction is a local operation performed on all the nodes of T in a top-down fashion. Let v be an internal node in the tree with children $v_1, v_2, \dots v_k$ (if v is a leaf then no operation will be done). Let (T, c, L) be the initial tree. The transformation on an internal node v will give us a new instance (T_v, c_v, L_v). We will add a dummy node v' for v, v'_i for every child i, and an additional dummy node v'_{k+1} also for v. We remove the edges $X = \{vv_i\}_i$ and add the edges $Y = \{vv'\} \cup \{v_i v'_i\}_i \cup \{v'_i v'_{i+1}\}_i$. We leave all the existing links at their corresponding nodes. The only link we add is a link called ℓ_v from v to v'_{k+1} of cost 0. The new instance has changed as follows:

$$V(T_v) = V(T) \cup \{v'\} \cup \{v'_i\}_i$$
$$E(T_v) = E(T) - X + Y$$
$$L_v = L \cup \{vv'_{k+1}\}$$

Fig. 1 gives an example of this transformation on a node with three children. Note that by adding the link vv'_{k+1} that has cost zero, all added dummy nodes v'_i are shrunk back into v, with the same set of children and the same link connections as in the original instance.

To transform the whole instance, we apply the above operation sequentially on internal nodes of the tree in any top-down order, i.e., the parent is processed with this operation before any node is. Note that for every internal node v with k children, we added two dummy nodes v' and v'_{k+1}, as well as one additional dummy node v'_i per child. Thus the total number of additional nodes of the former type is twice the number of internal nodes, and there is exactly one dummy node of the latter type per edge of the tree. This gives a total of $2(n-l) + (n-1)$ total additional nodes which along with the original n nodes in T gives the claimed size of T'. We will now show that transforming any TAP instance in this way produces an instance of TAP with a binary tree and leaf-to-leaf links with corresponding feasible solutions to the original problem.

First we observe that this transformation adds nodes $v', v'_1, \dots v'_k$ all of degree 3, adds node v'_{k+1} of degree 1, and node v ends with degree 1. The transformation also keeps the degree of $v_1, v_2 \dots v_k$ unchanged. Once this transformation has been applied to all non-leaves of T then the resulting tree T' will have only nodes of degree 1 and 3, giving a binary tree as desired.

Now observe that every original node is a leaf in T'. The only links we added were ℓ_v which have the form v'_{k+1} to v where v'_{k+1} is also a leaf under the transformation. The resulting set of links L' is thus leaf-to-leaf.

We will now consider any feasible solution A to (T, c, L). Denoting the non-leaf nodes of V by N, let $A' = A \cup \{\ell_v\}_{v \in N}$. The cost of A and A' are the same as we added only links ℓ_v which were given cost 0. First observe that ℓ_v covers all the edges of the form $v'_i v'_{i+1}$ and $v'v$. Now consider an edge $v'_i v_i$ after the transformation. There is some link $\ell \in A$ which covers vv_i in T and now that same link must cover $v'_i v_i$ in T'. So, A' is a valid solution to (T', c', L') of the same cost.

Now let A' be a feasible solution (T', c', L') and suppose there is a vertex $v \in N$ which was not initially a leaf node in T. It must be the case that A' contains ℓ_v as this is the only link in L' which covers $v'_k v'_{k+1}$. So, let $A = A' - \{\ell_v\}_{v \in N}$. Now by the same argument as before, as A' is a feasible solution for T' and the only edges in T' not in T are those covered by the ℓ_v then A is a valid solution to (T, c, L). Notice that A and A' have the same cost as we only removed links of cost 0 from the solution. □

2.2. A stronger LP relaxation than the EDGE-LP

Since we have shown that we can assume the tree is binary, every node has odd degree (1 or 3) in the input tree. Thus if $S \subseteq V$ is odd, then it follows that $\delta(S) \cap T$ is also odd. Using this observation, we can formulate a STRONG-LP as follows.

$$\min \sum_{\ell \in E} c_\ell x_\ell$$
$$x(\delta_L(S)) + \sum_{e \in \delta_T(S)} x(\delta_L(S_e)) \geq |\delta(S) \cap T| + 1 \qquad \forall S \subseteq V, |S| \text{ odd}$$
$$x_\ell \geq 0 \qquad \forall \ell \in E(L) \tag{3}$$

Lemma 2.2. *The constraints in STRONG-LP are valid for any integer solution to TAP.*

Proof. Consider an odd set of vertices S. By adding together the edge constraints for tree edges in $\delta_T(S)$ we get:

$$\sum_{e \in \delta_T(S)} x(\delta_L(S_e)) \geq |\delta(S) \cap T|.$$

Now we can add any non-negative terms to the left hand side and still remain feasible. Therefore

$$x(\delta_L(S)) + \sum_{e \in \delta_T(S)} x(\delta_L(S_e)) \geq |\delta(S) \cap T|$$

is also feasible. Now consider any link ℓ. If x_ℓ appears an even number of times in $\sum_{e \in \delta_T(S)} x(\delta_L(S_e))$ then ℓ is not in $\delta_L(S)$. Similarly, if x_ℓ appears an odd number of times in $\sum_{e \in \delta_T(S)} x(\delta_L(S_e))$

then ℓ is in $\delta_L(S)$. So, the coefficient of every x_ℓ on the left hand side of this expression is even. In particular, for any integer solution the left hand side is even and the right hand side is odd. Therefore, we can strengthen the right hand side by increasing it by one, and the resulting constraint will still be feasible for any integer solution. The constraint

$$x(\delta_L(S)) + \sum_{e \in \delta_T(S)} x(\delta_L(S_e)) \geq |\delta(S) \cap T| + 1$$

is thus valid for any integer solution to TAP as desired. □

3. Rounding LP-solutions with large non-zero values

The main result of this section is the following theorem.

Theorem 3.1. *Suppose we are given a solution x to the EDGE-LP with $x_\ell \geq \alpha$ whenever $x_\ell > 0$ and m is the number of non-zero links in x. Then there exists integer solutions $x^1, x^2, \ldots x^{2m}$ to EDGE-LP and non-negative multipliers $\lambda_1, \ldots, \lambda_{2m}$ with $\sum_{i=1}^{2m} \lambda_i = 1$ such that*

$$\frac{2}{1+\alpha} x \geq \sum_{i=1}^{2m} \lambda_i x^i$$

and this convex combination can be found in strongly polynomial time.

This gives an alternative proof of the main result of Cheriyan, Jordán and Ravi [7] that considered the case $\alpha = \frac{1}{2}$. In particular, it gives a $\frac{4}{3}$-approximation when we start with a fractional solution where all non-zero links have weight at least $\frac{1}{2}$.

3.1. Algorithm

We will be working with a tree rooted at an arbitrary node, r. The least common ancestor (LCA) of a link is the least common ancestor of its endpoints. For every link, we choose one endpoint as 'left' and the other as 'right' (say, based on a specific planar drawing of the tree). We let L_ℓ and R_ℓ be the path in the tree from the LCA of ℓ to the left and right endpoint of ℓ respectively (one of these paths could be empty).

Given a fractional solution, x let $\alpha = \min_{\ell: x_\ell \neq 0} x_\ell$, and let $\beta = \frac{2}{1+\alpha}$. Let k be the smallest integer such that $k\beta x$ is an even integer for all entries. In order to find our convex decomposition in the algorithm below, we will decompose $k\beta x$ into k different color such that each color is a feasible tree augmentation.

The main idea of how the algorithm works is that it goes down the tree looking at links which have their LCA at the current node and colors all the copies of each link with different colors so as to help cover the edges as much as possible with new colors (see Algorithm 1). This guarantees that the first $\alpha\beta k$ links (copies of one link) which are colored through an edge all get distinct colors. Afterward, we only guarantee that of the remaining links that cover an edge half of them give a new color to that edge.

We will now show that this coloring does indeed give us a convex combination as desired.

Theorem 3.2. *Algorithm 1 guarantees that every edge is covered by a link in every one of the k colors.*

Proof. For a given tree edge e without links of all k colors covering it, every time a link that covers e receives a pair of colors (in the inner while loop of the algorithm), one of those colors is new to e. Let us consider some link ℓ through e. Each inner while loop of the algorithm gives two colors to copies of ℓ. One of the two paths L_ℓ, R_ℓ must contain e; without loss of generality let $e \in L_\ell$.

Algorithm 1: The coloring algorithm.

```
Data: T a tree, x LP solution, β approximation factor, k colors
Result: Decomposition of kβx into k different colors where each color is a
        feasible tree augmentation
Make kβx_ℓ copies of each link ℓ;
while some link is not colored do
    ℓ has the highest LCA among uncolored links;
    while not all copies of ℓ are colored do
        Color a copy of ℓ with the first color not present on a link that
          covers some edge of L_ℓ;
        if all edges of L_ℓ are covered by all k colors then
            Color a copy of ℓ with any color not already on a copy of ℓ;
        end
        Color a copy of ℓ with the first color not present on a link that
          covers some edge of R_ℓ;
        if all edges of R_ℓ are covered by all k colors then
            Color a copy of ℓ with any color not already on a copy of ℓ;
        end
    end
end
```

Consider the highest edge $f \in L_\ell$ without links of all k colors covering it. If f is missing a color c among the colored links covering it, then e must also be missing color c among the colored links covering it. We have only colored links whose LCA is above f, therefore any link with a color which covers e must also cover f. So, for each pair of colors chosen for a link through e, at least one of them is a new color for e. In other words, half of the time a link covering e gets colored, it is a new color for e.

The first time a link through an edge e is colored, then all its colors are distinct (unless $\beta x_\ell > 1$). For a given link ℓ, every time a color is picked for a copy of ℓ it has to be a color not on one of the edges ℓ covers or a color not on any copy of ℓ. If $\beta x_\ell > 1$, then we can color the copies of ℓ with all k colors and all the edges which ℓ covers will be covered by all k colors. In this case, e would get all k colors.

Thus the first time an edge has one of its links colored it receives at least $\alpha\beta k$ distinct colors. Combining this with the fact that every edge gets colors at rate $\frac{1}{2}$ subsequently, the total number of colors e receives in this process is at least

$$\alpha\beta k + \frac{1-\alpha}{2}\beta k = \frac{1+\alpha}{2}\beta k = k. \quad \square$$

Now we will show how this implies Theorem 3.1.

Proof of Theorem 3.1. By scaling our x up by $k\beta$ we can write this scaled version as the sum of k different feasible colors (integer solutions). This gives us that:

$$k\beta x = \sum_{i=1}^{k} x^i$$

where the x^i are integer solutions. Dividing by k gives the desired result. □

Algorithm 1 does not need to first multiply by βk before being run. The algorithm can be run by just multiplying the solution by β. As the algorithm runs, it will keep track of a convex combination of integer partial solutions. In each while loop when a link ℓ is added, ℓ will be fully added to some integer partial solutions and added to a fraction of at most two partial integer solutions (one for R_ℓ and one for L_ℓ). This creates at most two more integer partial solutions. The number of different integer solutions at the end can be bounded by $2m$ where m is the number of non-zero links. This guarantees this algorithm can be run in strongly polynomial time.

4. Three-cycle TAP

Recall that the requirement in 3TAP is that every tree edge must be in a triangle in the final two-connected augmentation of the input tree. Such a triangle is either a link and two tree edges or two links and a single tree edge.

4.1. Weighted version

In this section, we will consider the weighted version of 3TAP where the weights on the links can have any non-negative values. We first present an $O(\log n)$ approximation algorithm, and then we present a matching lower bound of $\Omega(\log n)$, where n is the number of nodes in the tree.

Theorem 4.1. *There is a $O(\log n)$ approximation algorithm for weighted 3TAP on n nodes.*

Proof. Consider any feasible solution A to 3TAP, such that $T \cup A$ has every tree edge in a 3-cycle. For a vertex v, let $\delta(v)$ be the edges of $T \cup A$ adjacent to v. For the rest of this proof, assume that the edges of T have zero weight.

To turn this problem into a set cover problem, we let the edges of $E(T)$ be the elements. Any tree edge e is covered in a triangle that is centered at some node v that is not an endpoint of e; either both or one of the two edges covering e in the triangle is a link. Motivated by this, for any subset of edges and links adjacent to a vertex v, (the star) S_v, we construct a set with weight $c(S_v)$ (where tree edges in S_v have zero weight). This set covers the tree edges induced by the endpoints of the edges and links in S_v not containing v.

By doubling each edge in the feasible solution, then we can decompose the entire solution into these stars. In this way, given a solution to 3TAP of total weight C, the corresponding set cover has a solution of weight at most $2C$. Given any solution to the set cover problem, then we simply add all the edges specified by the stars to the tree (with maybe some duplicates when edges are added from both endpoints' stars). Thus, any solution of weight C to the set cover gives a solution to the original 3TAP solution of weight between $C/2$ and C. Therefore the optimal solutions to these two problems are within a factor of two of each other.

It is well known that minimum-cost set cover with n elements has an $O(\log n)$ approximation as long as the densest set (that has the maximum ratio of newly covered elements divided by the cost of the set) can be found in polynomial time. For a fixed vertex v, we can find the maximum density star centered at v as follows. Due to a result by Goldberg, one can find the maximum density subgraph $S \subset V$ which minimizes $\frac{|E(S)|}{c(S)}$ in polynomial time [14]. For the given center v, we build a graph G_v whose nodes are the neighbors of v using links and tree edges. The edges of G_v are the tree edges induced by these nodes. We set the cost of any vertex u to which v has a link to be $c(uv)$; if uv is a tree edge, then u has cost 0. We can now use the maximum density subgraph algorithm on G_v. By repeating this for every choice of center vertex v, we can find a maximum density star in polynomial time. This gives the maximum density set for the set cover problem in polynomial time. Then we can use the greedy algorithm for set cover to get an $O(\log n)$ approximation for 3TAP. □

Notice that in the above algorithm we used no properties of the original graph T. This algorithm will thus work for any graph T where the goal is to augment such that every edge in T is in a 3-cycle in the augmented graph.

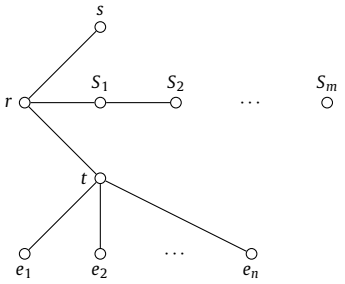

Fig. 2. The 3TAP instance created from a set cover instance.

Corollary 4.2. *The problem of finding a minimum cost augmentation of any graph G where every edge of G must be in a 3-cycle in the augmented graph has an $O(\log n)$ approximation.*

The above approximation is tight as the weighted 3TAP problem captures set-cover exactly. We will now show the matching lower bound.

Theorem 4.3. *3TAP does not have a $o(\log n)$-approximation unless $NP \subseteq P$.*

Proof. Consider an instance of set cover with sets $S_1, S_2, \ldots, S_m$ and elements $e_1, e_2, \ldots e_n$ and cost function c on the sets. We use the tree shown in Fig. 2. The vertex set is

$$\{r, s, t\} \cup \{S_i\}_{i=1}^m \cup \{e_j\}_{j=1}^n$$

with the following costs on the links:

- Links from s to vertices $\{r, t\} \cup \{S_i\}_{i=1}^m$ have zero cost
- Links from t to S_i have cost $c(S_i)$
- If $e_j \in S_i$ then the link from e_j to S_i has cost 0
- All remaining links have cost $1 + \sum_{i=1}^m c(S_i)$. Call the set of these remaining edges L.

In any optimal solution, we will not use any links from L as taking all the links not in that set has smaller cost and gives a feasible solution. The zero edges from s allow every edge except for the te_j edges to be in a three-cycle and they have cost 0. Now the only way to have an edge te_j in a three cycle is for tS_i and e_jS_i to be used for some S_i such that $e_j \in S_i$. Thus, the non-zero edges bought correspond to sets being chosen.

Given any feasible solution to set cover $S_{i_1}, \ldots S_{i_k}$, this can be turned into a feasible solution to 3TAP of the same cost. All the zero cost links in addition to the tS_{i_ℓ} edges form a feasible solution. All tree edges except for the te_j edges are in a three cycle with zero cost links. Consider any $j \in [n]$. There is some S_{i_t} that contains j. The edge te_j is then in a three cycle with tS_{i_ℓ} and $S_{i_\ell}e_j$. Hence, every feasible solution to the set cover instance gives a feasible solution of the same cost to the 3TAP instance.

Consider any feasible solution to our 3TAP instance. If the 3TAP solution contains a link from L then this solution has weight at least $1 + \sum_{i=1}^m c(S_i)$, but by taking all the sets S_i we get a feasible solution of lower cost. Therefore suppose there are no links from L in the feasible solution for the 3TAP instance. Let $tS_{i_1}, \ldots tS_{i_t}$ be the non-zero cost links in the solution. Then $S_{i_1}, \ldots S_{i_t}$ is a feasible solution to the set cover instance. Consider any element e_j. The edge te_j must be in a three cycle with t and some S_{i_ℓ}, therefore S_{i_ℓ} contains e_j and is a set in our solution to set cover. Therefore every feasible solution to 3TAP gives a corresponding solution to set cover with the same or smaller cost.

Any feasible solution to set cover gives a solution to 3TAP of the same cost. Any feasible solution to 3TAP, gives a feasible solution to

set cover of the same or smaller cost. Therefore, by the hardness of approximating set cover [2], it is impossible to approximate three-cycle TAP to within a $\Omega(\log n)$ factor unless $NP \subseteq P$. □

Remark: Suppose we were given an empty initial graph to augment and wish to find a minimum-cost two-edge-connected spanning subgraph where every edge is in a triangle, it is not hard to adapt the above hardness: We give all edges in the tree zero cost. By further subdividing the path of set nodes $S_1, S_2, \ldots, S_k$ to add new dummy nodes between every pair of set nodes, we can ensure that every element node e_j is covered only by triangles containing edge (t, e_j). This requires that the other edges in the cycle are of the form $(t, S_i), (S_i, e_j)$ for some set S_i containing the element e_j.

4.2. Unweighted version

While weighted 3TAP has many similarities to set cover, the unweighted version admits a constant approximation unlike set cover. Here we consider the case that every non-tree edge has cost either 1 or infinity, and every tree edge is present (and has cost 0). This 4-approximation comes from lower bounding the cost of every feasible solution to unweighted 3TAP.

Lemma 4.4. *Every feasible unweighted 3TAP solution has cost at least* $\frac{n-1}{2}$.

Proof. Consider any solution S. Duplicate all the links of S and edges T and decompose this doubled graph into stars around every vertex consisting of the edges adjacent to it. Call the star around v, S_v. This doubles the cost of the solution, but now we can see that every tree edge is covered by some star. At every vertex, we can further decompose S_v into $S_v^1, \ldots S_v^{\ell_v}$ such that we get stars that cover different connected components of the tree and every star contains at most one tree edge.

Now consider any star S_v^i. If S_v^i has l links, then the number of tree edges it can cover with 3-cycles is at most l. So, in the doubled instance of S there are at least $n-1$ tree edges that are covered. Every link is in at most 2 stars, so there must be at least $\frac{n-1}{2}$ links in any feasible solution. □

Corollary 4.5. *Unweighted 3TAP has a 4-approximation.*

Proof. We can get a 4 approximation by simply taking any minimal feasible solution. For every tree edge ab, pick a v such that av, bv both have cost 0 or 1. If no such vertex exists, then no feasible solution exists. Otherwise, the algorithm chooses at most $2(n-1)$ links. This gives a 4 approximation as desired. □

5. Conclusions

We have introduced a new top-down coloring method that gives a strict improvement over existing 2-approximation algorithms for weighted TAP, with better improvements for larger minimum values in the LP. Our method gives a constructive convex decomposition of a scaled solution of EDGE-LP into feasible integral solutions. When applied to the STRONG-LP, it has much potential to settle the integrality gap of TAP. Indeed, our coloring method and its extensions have already been applied to finding better convex combinations of two-edge-connected subgraphs in certain well-behaved instances [16,15]. We also settled the approximation complexity of the special case when all tree edges in the final solution must be in triangles – the extensions to short constant-length cycles in place of triangles is immediate.

Acknowledgements

This material is based upon work supported by the U.S. Office of Naval Research under award number N00014-21-1-2243 to RR.

References

[1] David Adjiashvili, Beating approximation factor two for weighted tree augmentation with bounded costs, ACM Trans. Algorithms 15 (2) (2018) 1–26.
[2] Sanjeev Arora, Madhu Sudan, Improved low-degree testing and its applications, Combinatorica 23 (3) (2003) 365–426.
[3] Robert Carr, Santosh Vempala, Randomized metarounding, in: Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, 2000, pp. 58–62.
[4] Federica Cecchetto, Vera Traub, Rico Zenklusen, Bridging the gap between tree and connectivity augmentation: unified and stronger approaches, in: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, 2021, pp. 370–383.
[5] Joseph Cheriyan, Zhihan Gao, Approximating (unweighted) tree augmentation via lift-and-project, part i: stemless tap, Algorithmica 80 (2) (2018) 530–559.
[6] Joseph Cheriyan, Zhihan Gao, Approximating (unweighted) tree augmentation via lift-and-project, part ii, Algorithmica 80 (2) (2018) 608–651.
[7] Joseph Cheriyan, Tibor Jordán, R. Ravi, On 2-coverings and 2-packings of laminar families, in: Algorithms-ESA'99, 1999, p. 72.
[8] Joseph Cheriyan, Howard Karloff, Rohit Khandekar, Jochen Könemann, On the integrality ratio for tree augmentation, Oper. Res. Lett. 36 (4) (2008) 399–401.
[9] Nachshon Cohen, Zeev Nutov, A (1+ ln2)-approximation algorithm for minimum-cost 2-edge-connectivity augmentation of trees with constant radius, Theor. Comput. Sci. 489 (2013) 67–74.
[10] Guy Even, Jon Feldman, Guy Kortsarz, Zeev Nutov, A 1.8-approximation for augmenting edge-connectivity of a graph from 1 to 2, ACM Trans. Algorithms 5 (2) (2009).
[11] Samuel Fiorini, Martin Groß, Jochen Könemann, Laura Sanità, Approximating weighted tree augmentation via Chvátal-Gomory cuts, in: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2018, pp. 817–831.
[12] Greg N. Frederickson, Joseph Jájá, Approximation algorithms for several graph augmentation problems, SIAM J. Comput. 10 (2) (1981) 270–283.
[13] Greg N. Fredrickson, Joseph Jájá, On the relationship between the biconnectivity augmentation and traveling salesman problem, Theor. Comput. Sci. 19 (2) (1982) 189–201.
[14] Andrew V. Goldberg, Finding a maximum density subgraph, Technical Report UCB/CSD-84-171, EECS Department, University of California, Berkeley, 1984.
[15] Arash Haddadan, Alantha Newman, Efficient constructions of convex combinations for 2-edge-connected subgraphs on fundamental classes, Discrete Optim. 42 (2021) 100659.
[16] Arash Haddadan, Alantha Newman, R. Ravi, Shorter tours and longer detours: uniform covers and a bit beyond, Math. Program. 185 (1) (2021) 245–273.
[17] Kamal Jain, A factor 2 approximation algorithm for the generalized Steiner network problem, Combinatorica 21 (1) (2001) 39–60.
[18] Samir Khuller, Ramakrishna Thurimella, Approximation algorithms for graph augmentation, J. Algorithms 14 (2) (1993) 214–225.
[19] Guy Kortsarz, Zeev Nutov, A simplified 3/2 ratio approximation algorithm for the tree augmentation problem, ACM Trans. Algorithms 12 (2) (2016) 23.
[20] Guy Kortsarz, Zeev Nutov, Lp-relaxations for tree augmentation, Discrete Appl. Math. 239 (2018) 94–105.
[21] Zeev Nutov, On the tree augmentation problem, in: 25th Annual European Symposium on Algorithms (ESA 2017), Leibniz International Proceedings in Informatics (LIPIcs), 2017, 61.
[22] R. Ravi, Steiner Trees and Beyond: Approximation Algorithms for Network Design, PhD thesis, Brown University, 1994.
[23] Vera Traub, Rico Zenklusen, Local search for weighted tree augmentation and Steiner tree, in: Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2022, pp. 3253–3272.
[24] Vera Traub, Riko Zenklusen, A better-than-2 approximation for weighted tree augmentation, in: Proceedings of the 62nd IEEE FOCS, 2021, CoRR, arXiv:2104.07114 [abs], 2021.