Improved Inapproximability Results for Steiner Tree via Long Code Based Reductions

Ali Çivril

February 15, 2017

Abstract

The best algorithm for approximating Steiner tree has performance ratio \( \ln(4) + \epsilon \approx 1.386 \) [J. Byrka et al., Proceedings of the 42th Annual ACM Symposium on Theory of Computing (STOC), 2010, pp. 583-592], whereas the inapproximability result stays at the factor \( \frac{96}{95} \approx 1.0105 \) [M. Chlebík and J. Chlebíková, Proceedings of the 8th Scandinavian Workshop on Algorithm Theory (SWAT), 2002, pp. 170-179]. In this article, we take a step forward to bridge this gap and show that there is no polynomial time algorithm approximating Steiner tree with constant ratio better than \( \frac{19}{18} \approx 1.0555 \) unless \( P = NP \). We also relate the problem to the Unique Games Conjecture by showing that it is \( UG \)-hard to find a constant approximation ratio better than \( \frac{17}{16} = 1.0625 \). In the special case of quasi-bipartite graphs, we prove an inapproximability factor of \( \frac{25}{24} \approx 1.0416 \) unless \( P = NP \), which improves upon the previous bound of \( \frac{128}{127} \approx 1.0078 \). The reductions that we present for all the cases are of the same spirit with appropriate modifications. Our main technical contribution is an adaptation of a Set-Cover type reduction in which the Long Code is used to the geometric setting of the problems we consider.

1 Introduction

In the Steiner tree problem, we are given an undirected graph \( G = (V, E) \), a cost function on the edges \( c : E \to \mathbb{Q}^+ \) and a subset of nodes \( T \subseteq V \) called terminals. The objective is to find a tree \( S \) of minimum cost \( c(S) := \sum_{e \in S} c(e) \) which connects all the terminals. Note that the solution might contain nodes that are not in \( T \). These nodes are often called Steiner nodes. The importance of the Steiner tree problem is mainly because it is a natural generalization of the minimum spanning tree problem and that it appears as a special case of a large number of network design problems that are of great interest in the field of approximation algorithms. As a result, it is one of the fundamental problems that have been intensively studied in theoretical computer science and operations research.

The Steiner tree problem already appears as an NP-hard problem in Karp’s classic paper [27], and the book by Garey and Johnson [15]. Thus, approximation algorithms have been sought. The fact that a minimum spanning tree on the terminals is twice the cost of an optimum solution has been exploited by early authors [16, 13, 24, 31, 33, 38] resulting in 2-approximation algorithms (see also [39]). The first algorithm breaking the barrier of factor 2 came from Zelikovsky [41] followed by a series of work [7, 31, 28, 22] culminating in the best purely combinatorial algorithm of factor \( 1 + \frac{\ln(3)}{2} + \epsilon \approx 1.55 \) by Robins and Zelikovsky [35, 36]. More recently, an LP-based algorithm was provided by Byrka et al. [9, 10] achieving the approximation ratio \( \ln(4) + \epsilon \approx 1.39 \), which stands as the current best result. The special case of the Steiner tree problem in which there are no edges between the Steiner nodes has been of usual interest in the literature. Such special instances are called quasi-bipartite graphs. The best approximation ratio on such instances is \( 73/60 + \epsilon \), again by [9, 10].
The intense activity on the positive results indicates that the Steiner tree problem exhibits a rich structure. In particular, the properties of the LP relaxations proposed for the problem and the venues for exploiting these relaxations to get approximation algorithms have not fully been understood. As a result, the issue of determining the approximability of Steiner tree continues to be a source of several interesting open problems and opportunities for research.

The aforementioned rich structure also leads to the fact that there is still a lot of work to be done about the negative results for the problem. Unlike many fundamental constraint satisfaction problems (CSPs) whose approximability are well understood as a result of direct relations to the PCP Theorem [5, 6] and the Label-Cover problem [4], the geometric nature of the Steiner tree problem is a source of resistance for strong inapproximability results. Many inapproximability results for such problems start from the hardness of the Max-3Lin(2) problem established by Håstad [21], and uses intricate gadgets. Such results state the hardness of approximation within $1 + 1/c$ for some moderately large constant $c$. For the Steiner tree problem, the best inapproximability result induces the value $c = 95$, which was proven by Chlebík and Chlebíková [11, 12] about a decade and a half ago, improving an earlier result of Bern and Plassman [8]. The same authors also prove an inapproximability ratio of $128/127$ for quasi-bipartite graphs.

It is an interesting fact that there is a plethora of problems for which we have a good reduction from the Label-Cover problem, which is ubiquitous in the PCP literature. The usefulness of this problem is due to its large completeness/soundness ratio and large alphabet size. A similar problem concerning much sparser instances but still having large alphabet size is Max-3Lin(q), the generalization of the previously mentioned Max-3Lin(2). A natural question is then (which can also be asked for many other combinatorial optimization problems) whether there is a direct reduction from these problems to the Steiner tree problem, making use of the large alphabet size via some construction akin to the Long code and hoping for a relatively strong inapproximability result. This possibility has also been posed as an interesting open problem by some researchers. The main goal of this paper is to show that there are such reductions yielding significantly improved inapproximability results for Steiner tree. In doing so, we also make use of the conditional hardness of Max-2Lin(q), thus relating the Steiner tree problem to the Unique Games Conjecture (UGC). The following are the main theorems we prove in this paper.

**Theorem 1.** Assuming UGC, it is NP-hard to approximate the Steiner tree problem within a constant ratio better than $17/16$.

**Theorem 2.** It is NP-hard to approximate the Steiner tree problem within a constant ratio better than $19/18$.

**Theorem 3.** It is NP-hard to approximate the Steiner tree problem on quasi-bipartite graphs within a constant ratio better than $25/24$.

### 1.1 Overview of the Results

Our result is inspired by an analogy between the Set-Cover problem and the Steiner tree problem. Consequently, the reduction we provide is a Set-Cover type reduction using the Long Code with a geometric flavor. Setting all the technical details aside, which will be fully proven in the analysis of the completeness and the soundness of the reduction, we give a brief overview of this analogy in this section.

We first give some terminology about the Steiner tree problem. A *component* is a tree whose leaves are all from the set of terminals $T$. A *k-component* is a component having at most $k$ terminals as leaves. A *k-restricted Steiner tree* $S$ is a collection of components whose union induces a Steiner...
Figure 1: An example of a 4-restricted Steiner tree

In this case, the cost $c(S)$ of $S$ is the total cost of its components, counting the duplicated edges with their multiplicity. An example of a 4-restricted Steiner tree is given in Figure 1, where terminals are depicted by squares, and Steiner nodes by filled circles.

Based on these definitions, one can define the directed-component cut relaxation (DCR) used by the authors of [9, 10]. In this relaxation, one considers components as directed graphs towards a unique sink. Specifically, given a subset $T' \subseteq T$ and an $r' \in T'$, consider the minimum-cost Steiner tree on $T'$ with edges directed towards $r'$, which we call a directed component. For a given directed component $C$, let $c(C)$ be its cost and $\text{sink}(C)$ be its unique sink terminal. Other nodes in the directed component will be called sources. Let the set of all components obtained this way be $C_n$. Let us say that a directed component $C \in C_n$ crosses a set $U \subseteq T$ if $C$ has at least one source in $U$ and the sink outside. Denote the set of directed components crossing $U$ by $\delta(U)$. Select an arbitrary terminal $r$ as a root. Then, the following is a relaxation for the Steiner tree problem:

$$\text{minimize} \quad \sum_{C \in C_n} c(C)x_C$$

$$\text{subject to} \quad \sum_{C \in \delta(U)} x_C \geq 1, \quad \forall U \subseteq T \setminus \{r\}, U \neq \emptyset,$$

$$x_C \geq 0 \quad \forall C \in C_n.$$

Strictly speaking, one needs to replace $C_n$ by $C_k$, which is the set of directed $k$-components since the cardinality of $C_n$ is exponential. However, the basic idea does not change. This relaxation is a covering LP. There is a variable for each directed component, and given any $U \subseteq T \setminus \{r\}$, we need to “cover” it by the directed components. This is quite similar to the standard LP relaxation of the Set-Cover problem where we have a collection $S$ of sets each being a subset of a universal set $U$, and the cost of a set $S \in S$ is denoted by $c(S)$:

$$\text{minimize} \quad \sum_{S \in S} c(S)x_S$$

$$\text{subject to} \quad \sum_{S : e \in S} x_S \geq 1, \quad \forall e \in U,$$

$$x_S \geq 0 \quad \forall S \in S,$$

where we obviously cover each element in $U$ by the sets in $S$. Thus, in our analogy, the set of elements $U$ in the Set-Cover problem correspond to the power set of terminals $T$, and the collection of sets $S$ correspond to the set of directed components $C_n$ in the Steiner tree problem.

With this analogy, we provide three reductions from Max-2Lin(q), Max-3Lin(3) and Max-3Lin(2) proving the three theorems mentioned in the introduction in order. Considering the bipartite
graph in which we have equations on one side and the variables on the other side (i.e. using the terminology of the Label-Cover problem), there will be a set of terminals in the Steiner tree instance for each edge. In the YES case, these terminals are almost completely “covered” by two complementary components corresponding to the two labels agreeing on the edge. The total cost of these two components will be 2, resulting in a total cost of $2|E|$, where $E$ is the edge set. In the NO case, where there is no labelling satisfying some fraction of the edges, we show that covering the terminals defined for an edge requires selecting a set of components whose total cost is strictly larger than 2 on average, yielding the results. The construction of the terminals for a given edge is inspired by the Long Code, which is ubiquitous in the PCP literature.

As the reader might also be wondering, the reason why one cannot get good results using the Label-Cover problem, the usual starting point for many hardness results, is worth mentioning. However, this will only be meaningful after detailed scrutiny of our constructions and is deferred to the discussion at the end of the paper, where we also talk about the possibility of improving upon the results of this paper using our ideas and the current PCP techniques.

1.2 Related Work

Steiner trees are crucial in many applications including VLSI and network routing [26, 30], phylogenetic tree construction, transportation and distribution networks [23]. From a theoretical viewpoint, the importance of the problem is due to its appearance as a special case or a subproblem of many other combinatorial optimization problems. Its elegance allows generalizations in many different directions. In particular, the following is a (far from complete) list of important problems which contains the Steiner tree problem as a special case, and for which the inapproximability result we prove are valid:

- Steiner forest [1, 19],
- Prize-collecting Steiner tree [19, 2],
- Prize-collecting Steiner forest [37, 20],
- Generalized Steiner network [17, 40, 25],
- Multicommodity rent-or-buy [14].

The complexity of the problem where the input is a Euclidean space is well understood. In that case, the problem admits a PTAS by celebrated results of Arora [3] and Mitchell [32]. In general, one can assume without loss of generality that input satisfies the axioms of a metric space. In this case, the integrality gap of the directed-component cut relaxation is lower bounded by $8/7$ [10], and upper bounded by $\ln(4)$ [18].

2 Preliminaries

In this section, we give relevant definitions and results about the starting points of our reductions. An instance of the problem Max-3Lin($q$) consists of linear equations modulo $q$, where each equation has three variables taking values from $\mathbb{Z}_q$. The goal of the problem is to find an assignment to the variables so as to satisfy as many equations as possible. We say that an algorithm is a $(c, s)$-approximation for the problem if it can find an assignment satisfying at least $s$-fraction of the equations given that $c$-fraction of the equations are satisfiable. The following is a well-known result by Håstad.
Theorem 4 (21). Given any $\epsilon, \delta > 0$ and $q \in \mathbb{N}$, it is NP-hard to $(1 - \epsilon, \frac{1}{q} + \delta)$-approximate Max-3Lin($q$).

The analogous problem in which we have 2 variables per equation is named Max-2Lin($q$). It is not possible to derive NP-hardness for this version using the standard PCP machinery. However, one can state hardness conditioned upon UGC.

Theorem 5 (29). Assuming UGC, given any $\epsilon, \delta > 0$, there exists $q \in \mathbb{N}$ such that it is NP-hard to $(1 - \epsilon, \delta)$-approximate Max-2Lin($q$).

Given any of these problems, we will consider the bipartite graph $G = (V, W, E)$, where $V$ is the set of equations and $W$ is the set of variables. $E$ consists of all the edges $e = (v, w)$ between $v \in V$ and $w \in W$ if the variable corresponding to $w$ appears in the equation corresponding to $v$. We consider a set of labels $\{1, \ldots, q\} = [q]$ for the vertices in $W$ representing the values that the corresponding variable might take. We also consider the set of labels $[r]$ for the vertices in $V$, where each label represents a satisfying assignment for the corresponding equation. This label may be seen as a triple (for Max-3Lin($q$)) or a tuple (for Max-2Lin($q$)) containing the values for all the variables that the equation contains. We say that an edge $e = (v, w)$ is satisfied if the label of $w$ agrees with the corresponding coordinate on the label of $v$. More formally, for each vertex-label pair $(v, j)$ and each edge $e = (v, w)$ incident to $v$, where $v \in V$ and $j \in [r]$, we define $\Pi_e(j)$ to be the label $i \in [q]$ of $w$ agreeing the label on the appropriate coordinate of $j$. Given this, we say that $e$ is satisfied if $\Pi_e(j) = i$. Let $\Pi$ be the set of such projection functions for all $e \in E$. As a result, our instance can be regarded as a quadruple $(G = (V, W, E), [q], [r], \Pi)$. It is important to note that the hardness results stated above are not valid for the number of satisfied edges. However, if an equation corresponding to $v$ can be satisfied, then all the edges incident to $v$ can be satisfied by the appropriate labelling. If it is not satisfied on the other hand, then we know that at least one edge incident to $v$ is not satisfied.

In a typical reduction from the Label-Cover type problem as given above, one defines problem specific elements for each vertex-label pair. The Long Code and sometimes more efficient versions such as the Hadamard code are useful tools in such reductions in the sense that if one designs elements for the labels conforming to the structure of their codewords, one can get good completeness/soundness ratio. Given the set of labels $[q]$, the Long Code encoding of $j \in [q]$ is defined to be the truth table of the function $f: \{0, 1\}^q \to \{0, 1\}$ such that $f(x_1, \ldots, x_q) = x_j$. Note that this amounts to codewords of length $2^q$. One interesting property of this code is that given any two codewords, the number of coordinates on which they have both 1 is $2^{q-2}$. The same holds for the 0s. Thus, they agree on half of the coordinates. For our purposes, we use a different terminology for the Long Code. Given a set of codewords, let us say that we “cover” the coordinates for which we have at least one 1 belonging to a codeword. Thus, a single codeword covers half of the coordinates. It is not difficult to see that the number of uncovered coordinates drops by half as we add a new codeword to our set. As a result, by selecting $q$ different codewords, we can cover all the coordinates except one.

3 Proving Theorem 1: Reduction from Max-2Lin($q$)

In order to illustrate the basic idea of our reductions in full generality, it is convenient to start from the conditional hardness result, where we use the large alphabet size of Max-2Lin($q$). The basic setting together with the terminology introduced in this section will also be used in the other two reductions with appropriate modifications.
Figure 2: Example of an edge-component for $q = 4$ with component trees of $(w, i)$ for $i = 1$ and $i = 3$, where the component edges of cost $1/4$ are in bold lines.

To avoid confusion, we use the word *vertex* when referring to a Max-$2\text{Lin}(q)$ instance and the word *node* for a Steiner tree instance. Let a Max-$2\text{Lin}(q)$ instance $(G(V, W, E), [q], [r], \Pi)$ be given as discussed in the Preliminaries section. Denoting $|V| = m$, we have $|E| = 2m$. We define a Steiner tree instance $S$ as follows: There exists a set of $2^q$ terminals for each edge in $E$, a total of $2^q|E| = 2^{q+1}m$ terminals. We call this set of terminals for an edge $e \in E$, the edge-component of $e$. The terminals of an edge-component represent entries of the truth table representing a function on $q$ binary variables. In other words, we enumerate the terminals of an edge-component from 1 to $2^q$ such that they correspond to the lexicographic order of the binary variables $(x_1, \ldots, x_q)$ taking values from the set $\{0, 1\}$, where 0 precedes 1 in the ordering. For each vertex-label pair $(w, i)$ and each edge $e$ incident to $w \in W$ and $i \in [q]$, we define a binary tree whose leaves are the set of terminals for which the Long Code encoding of $i$ is 1. For example, if $i = 1$, then the tree is defined on the terminals whose coordinates correspond to the second half. If $i = 2$, it is defined on the terminals with coordinates corresponding to the second and fourth quarter and so on (see Figure 2, where the ordering of the terminals increase from top to bottom and left to right). We call this tree the component tree of $(w, i)$ on $e$. Note that this is a $2^{q-1}$-component in the Steiner tree terminology. The root of the tree has four children, all having cost $1/4$. Each of the four children is itself the root of a binary tree. Each such binary tree has $2^{q-3}$ leaves. We recursively divide these leaves into two taking their ordering into account. Each level of a binary tree has two Steiner nodes with the appropriate leaf set. In particular, at the lowest level of the recursion, only the adjacent terminals are connected by an edge, thus creating a Steiner node (see Figure 2). The cost of the binary tree edges is set to 0.

We now define the structures for the $V$-side of the Max-$2\text{Lin}(q)$ instance. It is symmetric to the $W$-side modulo differences due to the projection function $\Pi$. For each vertex-label pair $(v, j)$, and each edge $e = (v, w)$ incident to $v \in V$ and $j \in [r]$, we define a tree whose leaves are the set of terminals for which the Long Code encoding of $\Pi_e(j)$ is 0. This is the crucial part of the reduction which enforces a low cost tree on $e$ for compatible labels. Similar to the definition for the $W$-side, we call this tree the component tree of $(v, j)$ on $e$. The root of the tree (which we call the component root of $(v, j)$ on $e$) has four children, all having cost $1/4$. Each of the four children is itself the root of a binary tree defined in the same way as for the $W$-side. The cost of each binary tree edges is 0.

In order to relate to the compatibility of the labels on both sides, we connect the component root of $(w, i)$ on $e$ with the component root of $(v, j)$ on $e$ provided that $\Pi_e(j) = i$. We call this edge a complementing edge of $i$ on $e$, which has cost 0 (See Figure 3). The endpoints of this edge are defined to be the relevant component roots. Note that there might be more than one such edge in
Max-3Lin(q) since it is possible that many labels map to $i$ under $\Pi_e$ (but not in Max-2Lin(q)). In fact, in such a case, there are more than one component trees on the same set of terminals whose roots are connected to the component root of $(w,i)$ on $e$.

The construction we have defined so far does not ensure the connectivity of the whole graph. One needs to relate the component trees across different edge-components. It is a crucial property of our construction that we allow these connections to be only via the $V$-side. We connect all the component roots of $(v,j)$ on edges incident to $v$ to a single root which we call the label root of $(v,j)$. The edges incident to this node are called label tree edges of $(v,j)$, each having cost 0. Note that there are only two such edges since the degree of the vertices in $V$ is two in Max-2Lin(q). Of course, this number is three for Max-3Lin(q). The whole tree rooted at the label root is called the label tree of $(v,j)$.

Strictly speaking, the Steiner tree instance we have defined so far is still disconnected. As a final step, we connect its connected components with edges of total cost 0. This can easily be done by selecting a terminal from an edge-component in one (graph-theoretic) component of the Steiner tree instance and adding an edge between this terminal and another one in an edge-component residing in another (graph-theoretic) component. Since the number of components in the Steiner tree instance is at most $m$ each defined by the vertices in $V$, we can perform this operation by introducing at most $m-1$ edges.

### 3.1 Constructing a Valid Steiner Tree

Before proving the completeness and the soundness, we describe in detail how to construct a valid Steiner tree in the reduced instance and the related costs. Note that if an edge is satisfied in the Max-2Lin(q) instance, then taking the component trees corresponding to a satisfying assignment together with the complementing edge will cover all the terminals in that edge-component. However, even in the YES case of the reduction, there are edges that cannot be satisfied. Thus, they cannot be covered using this method. Of course, this does not mean that there are no valid Steiner trees covering all the terminals. The key observation is that, given an edge-component, one might need to take several component trees corresponding to a subset of the label set. One needs to examine the cheapest way of constructing a valid Steiner tree by looking at a subset of edges of the set of aforementioned component trees and argue that there is a bound for its total cost. One way of covering all the terminals in an edge-component other than selecting two complementing components is depicted in Figure 4. Here, we have component trees corresponding to two labels $i,j \in [q]$, and one label $k$ which is a complement of these, say satisfying $\Pi_e(k) = j$. In this case,
Figure 4: An example of 3 component trees covering all the terminals of an edge-component and the component edges in bold lines that must be taken in the corresponding Steiner tree.

one is forced to take at least 9 component edges of total cost $9/4$ in order to make sure that the tree contains all the terminals of the edge-component. It is not difficult to see that this is not the only way of connecting all the terminals. In particular, one can continue selecting components corresponding to labels $i_1, i_2, i_3, \ldots \in [q]$ before selecting a complement of one these components.

Motivated by the discussion thus far, we define a procedure for constructing a valid Steiner tree as follows in order to define all possible ways of selecting a Steiner tree on a given edge-component. For all vertices $v \in V$, take a subset $Q(v) \subseteq [q]$ of labels and for all vertices $w \in W$, take a subset $R(w) \subseteq [r]$ of labels such that the component trees in the Steiner tree instance corresponding to the set

$$\bigcup_{v \in V} Q(v) \cup \bigcup_{w \in W} Q(w)$$

covers all the terminals. Consider an arbitrary subset of edges of this set which forms a Steiner tree. By construction, there are two fundamentally different ways of covering all the terminals of an edge-component corresponding to an edge $e = (v, w)$. In the first case as mentioned previously, one selects two complementing component trees $(w, i)$ on $e$ and $(v, j)$ on $e$ such that $\Pi_e(j) = i$ together with their complementing edge. If the terminals of an edge-component are covered in this manner, we call the corresponding edge a Type-I edge. Note that the cost of a Type-I edge is 2, since we have to take all the 8 component edges of the two complementing components, each having cost $1/4$.

In the more complicated second case, one selects 9 component edges as described in Figure 4 or the cheapest subset of edges of a set of component trees

$$\{(v, j_1), \ldots, (v, j_k)\} \cup \{(w, i_1), \ldots, (w, i_{q-k+1})\}$$

all on $e$ such that

1. The component trees $(v, j_1), \ldots, (v, j_k)$ on $e$ are all distinct.
2. $\Pi_e(j_\ell) = i_{\ell'}$ for some $\ell \in [k]$ and $\ell' \in [q-k+1]$.
3. There are no other label pairs agreeing on $e$ other than the one in the previous item.

Let us discuss and justify this second case in more detail. As discussed in the Preliminaries section, by the structure of the Long Code, one can cover all the terminals except one by selecting $q$ distinct component trees which do not contain any two that are complements of each other.
since the addition of a new tree decreases the fraction of uncovered terminals by half. This is why we require the first condition of distinctness for the component trees on the $V$-side. The component trees on the $W$-side are already distinct. In order to cover the last terminal, one needs the second condition, which says that at least one more component tree which is a complement of another should be selected. Thus, one selects $q + 1$ component trees in total satisfying all the three conditions above. Note that such a selection also comes with a cost of $9/4$ since we are forced to select 9 component edges (See Figure 5). The set of 3 component trees from which we select the 9 component edges does not matter as the Long Code and hence the component trees are symmetric. 

Note also that after selecting the first 3 component trees, the remaining component trees can be connected to these by using edges from the levels that are lower than those of the component edges, without introducing extra cost. If the terminals of an edge-component are covered as in the second case, we say that the corresponding edge is a Type-II edge.

Since we are in need of constructing a tree, we also need to analyze the cost of connecting the edge-components, which can only happen via the vertices in $V$. Let the connection between two Type-I edges via label tree edges be called a Type-I-I connection. Similarly, let the connection between two Type-II edges be called a Type-II-II connection. We also call the connection between a Type-I edge and Type-II edge a Type-I-II connection. We now analyze the cost of these connections.

Given a Type-I-I connection via a vertex $v$, if the component trees corresponding to the labels assigned to the edges agree on $v$, we only need to take two label tree edges for the connection with a total cost of 0 since these edges directly connect the roots of the component trees representing the same label. Such a connection will be called a cheap Type-I-I connection. Otherwise, i.e. if the labels do not agree on $v$, then we need to take one component edge together with two label tree edges with a total cost of $1/4$. The reason is that starting from the root of a component tree of label $i_1$ on an edge, we can only “arrive” at the component tree of again label $i_1$ on the other edge. Yet, in the solution the component tree of some other label $i_2$ is taken. Thus, to create a tree one is forced to take at least one component edge of the component tree of $i_1$. We call this type of connection an expensive Type-I-I connection. This situation is illustrated in Figure 6 where we have two edge-components of both Type-I and the total number of component edges that must be taken is 17.

A similar difference exists for Type I-II connections. Note that a Type-II edge takes component edges from at least 3 different component trees (See Figure 4). If the Type-I edge agrees with the Type-II edge in the sense that it takes a component tree from this set, then the connection involves two label tree edges with a total cost of 0. We call such a connection a cheap Type-I-II connection. Otherwise, if the label of the Type-I edge does not agree with the 3 labels associated with the
Figure 6: An example of an expensive Type I-I connection, where $q = 3$ and the component edges that must be taken are shown in bold lines.

Type-II edge, then the connection involves two label tree edges and one component edge with a total cost of $1/4$ because of the same reason explained in the previous paragraph. We call this type of connection an expensive Type I-II connection.

As for a Type II-II connection, all we know is that it must use two label tree edges. Thus, the only lower bound on the total cost of a Type II-II connection is 0.

### 3.2 Completeness and Soundness of the Reduction

**Lemma 6.** If the fraction of satisfiable equations in the $\text{Max-2Lin}(q)$ is at least $(1 - \epsilon)$, then there exists a Steiner tree of cost at most $(4 + \frac{\epsilon}{4})m$ in the Steiner tree instance $S$.

**Proof.** Take an assignment which satisfies at least $(1 - \epsilon')$-fraction of the equations where $\epsilon' \leq \epsilon$. Take the component trees corresponding to this assignment together with the complementing edges, one for each edge-component defined for all the satisfied equations (a total of $2(1 - \epsilon')m$). By definition, they “cover” all the corresponding terminals since two component trees agree on an edge-component via a complementing edge. The total cost of the Steiner tree induced by the satisfied equations is then

$$2(1 - \epsilon')m \cdot \left(8 \left(\frac{1}{4}\right)\right) = 4m - 4\epsilon'm.$$

There remains to analyze the cost for the unsatisfied equations. We let all the $2\epsilon'm$ edges incident to the relevant vertices be of Type-I. Then, the connection can be at most $\epsilon'm/4$ via expensive Type I-I connections. Thus, the total cost is at most

$$2\epsilon'm \left(8 \left(\frac{1}{4}\right)\right) + \epsilon'm \left(\frac{1}{4}\right) = \frac{17}{4}\epsilon'm.$$

Summing the last two expressions and recalling that $\epsilon' \leq \epsilon$, we get the bound in the lemma.

**Lemma 7.** If the fraction of satisfiable equations in the $\text{Max-2Lin}(q)$ is at most $\delta$, then the cost of any Steiner tree in $S$ is at least $(\frac{17}{4} - \frac{\delta}{4})m$.

**Proof.** Take any assignment which satisfies $\delta'$-fraction of the equations where $\delta' \leq \delta$. Consider the satisfied equations. The terminals of the edge-components corresponding to the edges incident to these equations can be covered with cost at least

$$2\delta'm \left(8 \left(\frac{1}{4}\right)\right) = 4\delta'm.$$
For any unsatisfied equation, we know that at least one edge incident to the corresponding vertex is unsatisfied. There are 4 possibilities for the two edges incident to that vertex with respect to their types. Eliminating 3 of them, if at least one of the edges is of Type-II, then the associated cost is at least

$$(1 - \delta')m \left( 9 \left( \frac{1}{4} \right) \right) + (1 - \delta')m \left( 8 \left( \frac{1}{4} \right) \right) = \frac{17}{4}m - \frac{17}{4}\delta'm,$$

making the total cost at least

$$\left( \frac{17}{4}m - \frac{17}{4}\delta'm \right) + 4\delta'm = \frac{17}{4}m - \frac{1}{4}\delta'm$$

$$\geq \frac{17}{4}m - \frac{1}{4}\delta'm.$$

The possibly problematic case which would result in a low cost is when we have both edges of Type-I. In this case, we have a Type I-I connection. If this is a cheap Type I-I connection, we can use the label via which this connection is made for the vertex to satisfy both of the edges, thus satisfying the equation. As a result, the number of such connections cannot be more than $(\delta - \delta')m$, i.e. at least $(1 - \delta)$-fraction of such connections must be expensive, which has cost 1/4. Otherwise, we could satisfy more than $\delta$-fraction of the equations contradicting the premise of the lemma. Then, the cost of interest is at least

$$2(1 - \delta')m \left( 8 \left( \frac{1}{4} \right) \right) + (1 - \delta)m \left( \frac{1}{4} \right) = \frac{17}{4}m - 4\delta'm - \frac{1}{4}\delta'm,$$

making the total cost at least

$$\left( \frac{17}{4}m - 4\delta'm - \frac{1}{4}\delta'm \right) + 4\delta'm = \frac{17}{4}m - \frac{1}{4}\delta'm,$$

which proves the lemma.

The proof of the theorem now easily follows. For any $\epsilon' > 0$, one can select $\epsilon$ and $\delta$ such that the ratio between the soundness and the completeness satisfies

$$\frac{17}{16} - \frac{\delta}{3 + \frac{\epsilon}{4}} = 17 - \epsilon'.$$

Combining this with Theorem 5, we get the desired UG-hardness.

4 Proving Theorem 2: Reduction from Max-3Lin(3)

The most natural approach to prove an unconditional hardness would be to imitate the reduction of Theorem 1 using Max-3Lin(q). However, there is a better option giving better inapproximability ratio. In order to see this, let us explain first why we have used a component tree with a root of four children in the first reduction, which also stands as the reason that prevents us from pushing the inapproximability ratio further than 17/16. Observe that a binary tree with a root of two children cannot force a Type-II edge with cost strictly larger than 2. Indeed, in such a case it is not difficult to see that one can connect all the terminals in an edge-component by selecting only four component edges with total cost 2 even in the case of a Type-II edge. Thus, one resorts to designing component edges of cost 1/4 in order to create the gap between the cost of Type-I and
Type-II edges. Instead of repeating the same for the proof of Theorem 2, we observe that there is a way of designing component edges of cost $1/2$ for this case by considering a subset of the Hadamard code with codeword length 8 (hence, using alphabet size 3), and also creating the desired gap. Of course, in this case the soundness is $1/3 + \delta$ instead of $1/q + \delta$, which can be made arbitrarily close to 0. However, the gain from considering heavy-weight component edges is greater than the disadvantage of large soundness.

4.1 The Reduction

We use the same approach and the terminology introduced in the previous section and only highlight the main differences. Let a Max-3Lin(3) instance $(G(V, W, E), [3], [9], \Pi)$ be given. Denoting $|V| = m$, we have $|E| = 3m$. In the Steiner tree instance $S$, there exists a set of $2^3 = 8$ terminals for each edge in $E$, a total of $8|E| = 24m$ terminals. For the vertices in $W$, the 3 component trees that we make use of correspond to the 3 codewords $00001111$, $00110011$, $11000011$ which form a subset of the Hadamard code (See Figure 7a). The root of the component trees have two children and the component edges have cost $1/2$.

The cost of a Type-I edge is 2 upon selecting complementing component trees and their complementing edge. One difference of the reduction in this section from the previous one is that there are more than one component trees defined for the labels on the $V$-side, since there are more than one label mapping to a single label on the $W$-side. This gives freedom of selecting edges from different component trees defined on the same set of 4 nodes. However, this does not change the cost of a Type-I edge. One necessarily has to select two component edges either from the same component tree or from two different component trees in order to cover those 4 nodes. Note also that in this construction, one has to take 5 component edges to cover the 8 terminals if Type-I is to be avoided. Thus, the cost of a Type-II edge is $5/2$ (See Figure 7b). Similar to the previous reduction, because of the symmetry of the code we use, this cost is invariant under different selections of the component trees. The cost of an expensive Type I-I connection is $1/2$. Similarly, the cost of an expensive Type I-II connection is $1/2$. All other connections can be at a cost of 0.
4.2 Completeness and Soundness of the Reduction

**Lemma 8.** If the fraction of satisfiable equations in the \textit{Max-3Lin}(3) is at least \((1 - \epsilon)\), then there exists a Steiner tree of cost at most \((6 + \epsilon)m\) in the Steiner tree instance \(S\).

**Proof.** Take an assignment which satisfies at least \((1 - \epsilon')\)-fraction of the equations where \(\epsilon' \leq \epsilon\). Take the component trees corresponding to this assignment together with the complementing edges, one for each edge-component defined for all the satisfied equations (a total of \(3(1 - \epsilon')m\)). These edges cover all the corresponding terminals and their total cost is

\[
3(1 - \epsilon')m \cdot \left(4 \left(\frac{1}{2}\right)\right) = 6m - 6\epsilon'm.
\]

There remains to analyze the cost for the unsatisfied equations. Similar to the proof of Lemma 6, we see that their cost can be bounded by

\[
3\epsilon' m \left(4 \left(\frac{1}{2}\right)\right) + 2\epsilon' m \left(\frac{1}{2}\right) = 7\epsilon'm,
\]

where we let all the \(3\epsilon'm\) edges be of Type-I, and all the \(2\epsilon'm\) connections are expensive Type I-I connections. Summing the last two expressions and recalling that \(\epsilon' \leq \epsilon\), we get the bound in the lemma.

**Lemma 9.** If the fraction of satisfiable equations in the \textit{Max-2Lin}(q) is at most \(\frac{1}{3} + \delta\), then the cost of any Steiner tree in \(S\) is at least \((\frac{19}{3} - \frac{1}{2}\delta)m\).

**Proof.** Take any assignment satisfying \(\delta'\)-fraction of the equations where \(\delta' \leq 1/3 + \delta\). Consider first the satisfied equations. The terminals of the edge-components corresponding to the edges incident to these equations can be covered with cost

\[
3\delta' m \left(4 \left(\frac{1}{2}\right)\right) = 6\delta'm.
\]

For any unsatisfied equation, at least one edge incident to the corresponding vertex is unsatisfied. There are 8 possibilities for the two edges incident to that vertex with respect to their types. Eliminating 7 of these possibilities, if at least one of the edges is of Type-II, then the associated cost is at least

\[
(1 - \delta')m \left(5 \left(\frac{1}{2}\right)\right) + 2(1 - \delta')m \left(4 \left(\frac{1}{2}\right)\right) = \frac{13}{2}m - \frac{13}{2}\delta'm,
\]

making the total cost at least

\[
\frac{13}{2}m - \frac{13}{2}\delta'm + 6\delta'm = \frac{13}{2}m - \frac{1}{2}\delta'm
\geq \frac{13}{2}m - \frac{1}{2} \left(\frac{1}{3} + \delta\right)m
= \frac{13}{2}m - \frac{1}{6}m - \frac{1}{2}\delta'm
= \frac{19}{3}m - \frac{1}{2}\delta'm.
\]

There only remains the case where we have all Type-I edges. One can consider two further cases in it to analyze. In the first case, only one of the edges is satisfied. In the second, two of the edges are
satisfied. In both of these cases though, there must exist at least one expensive Type I-I connection of cost $1/2$. Indeed, if all the connections are cheap between the 3 edges, one can satisfy all of them (hence, the whole equation) by using the label via which these cheap connections are made. Consequently, the number of expensive Type I-I connections is at least $(2/3 - \delta)m$. Otherwise, we could satisfy more than $(1/3 + \delta)$-fraction of the equations contradicting the premise of the claim. Then, the cost of interest is at least

$$3(1 - \delta')m \left( 4 \left( \frac{1}{2} \right) \right) + \left( \frac{2}{3} - \delta \right)m \left( \frac{1}{2} \right) = \frac{19}{3}m - 6\delta'm - \frac{1}{2}\delta m,$$

making the total cost at least

$$\left( \frac{19}{3}m - 6\delta'm - \frac{1}{2}\delta m \right) + 6\delta'm = \frac{19}{3}m - \frac{1}{2}\delta m.$$

proving the lemma. 

For any $\epsilon' > 0$, one can select $\epsilon$ and $\delta$ such that the ratio between the soundness and the completeness satisfies

$$\frac{19 - \delta}{6 + \epsilon'} = \frac{19}{18} - \epsilon'.
$$

By Theorem 4, this completes the proof of Theorem 2.

5 Proving Theorem 3: Reduction from Max-3Lin(2)

Recall that in the quasi-bipartite graphs, there are no edges between Steiner nodes. In both of the constructions we have analyzed, there are such edges since we have considered trees of depth greater than 1. One quick idea to extend these results to the case of quasi-bipartite graphs is to introduce terminals on the edges defined between the Steiner nodes, thus creating subdivisions. Unfortunately, if we do so, even the YES case of the reductions cannot be attained by taking complementing component trees as they do not fully cover all the terminals. A good idea is to keep the depth of the component trees at 1, which calls for a reduction from Max-3Lin(2).

5.1 The Reduction

Let a Max-3Lin(2) instance $(G(V, W, E), [2], [4], \Pi)$ be given. Denoting $|V| = m$, we have $|E| = 3m$. We define a Steiner tree instance $S$ as follows: There exists a set of $2^2 = 4$ terminals for each edge in $E$, a total of $4|E| = 12m$ terminals. For the vertices in $W$, the 2 component trees that we make use of correspond to the 2 codewords of the Long Code. The root of the component trees have two children which are the usual terminals of the edge-component. The component edges have cost $1/2$. We should not have edges between Steiner nodes. Thus, we do not introduce complementing edges. Instead, the component trees for the agreeing labels meet at the same Steiner node. Note also that given an edge $e = (v, w)$, there are two distinct labels on $v$ agreeing with a label on $w$. In order to handle this, we create two copies of the labels for the vertices in $W$ and connect these to the appropriate two labels on the $V$-side (See Figure 8a). Thus, there are 4 Steiner nodes each being the root of a tree with all the terminals of an edge-component. Given a label for $v \in V$, we identify all the 3 Steiner nodes belonging to the 3 edges incident to $v$ corresponding to that label, i.e. there is a single Steiner node as the root of all the 12 terminals defined for $v$. This makes sure that we have a quasi-bipartite graph.
Complementing component trees for a single label

An example of 5 component edges of a Type-II edge

Figure 8: The construction of the quasi-bipartite instance

Similar to the reduction in the previous section, the cost of a Type-I edge is 2 upon selecting complementing component trees, which cover the terminals. A Type-II edge must take 5 component edges with a total cost of $5/2$ (See Figure 8b). The cost of an expensive Type I-I connection is $1/2$, so is the cost of an expensive Type I-II connection. All other connections can be at a cost of 0.

### 5.2 Completeness and Soundness of the Reduction

The proof of the following lemma is exactly the same as the proof of Lemma 8, which we do not repeat.

**Lemma 10.** If the fraction of satisfiable equations in the Max-3Lin(2) is at least $(1 - \epsilon)$, then there exists a Steiner tree of cost at most $(6 + \epsilon)m$ in the Steiner tree instance $S$.

**Lemma 11.** If the fraction of satisfiable equations in the Max-2Lin($q$) is at most $1/2 + \delta$, then the cost of any Steiner tree in $S$ is at least $\left(\frac{25}{4} - \delta \right)m$.

**Proof.** Take any assignment which satisfies $\delta'$-fraction of the equations where $\delta' \leq 1/2 + \delta$. The terminals of the edge-components corresponding to the edges incident to the satisfied equations can be covered with cost

$$3\delta' m \left(4 \left(\frac{1}{2}\right)\right) = 6\delta' m.$$  

For any unsatisfied equation, at least one edge incident to the corresponding vertex is unsatisfied. If at least one of the edges is of Type-II, then the associated cost is at least

$$(1 - \delta')m \left(5 \left(\frac{1}{2}\right)\right) + 2(1 - \delta')m \left(4 \left(\frac{1}{2}\right)\right) = \frac{13}{2} m - \frac{13}{2} \delta' m,$$

which makes the total cost at least

$$\frac{13}{2} m - \frac{13}{2} \delta' m + 6\delta' m = \frac{13}{2} m - \frac{1}{2} \delta' m \geq \frac{13}{2} m - \frac{1}{2} \left(\frac{1}{2} + \delta\right) m = \frac{13}{2} m - \frac{1}{4} m - \frac{1}{2} \delta m = \frac{25}{4} m - \frac{1}{2} \delta m.$$
There remains the case where we have all Type-I edges. Similar to the reasoning in the proof of Lemma 9 there must exist at least one expensive Type I-I connection of cost \( 1/2 \) for a given vertex in \( V \). Thus, the number of expensive Type I-I connections is at least \( (1/2 - \delta) m \). Otherwise, we could satisfy more than \((1/2 + \delta)\)-fraction of the equations contradicting the premise of the lemma. Then, the cost of interest is at least

\[
3(1 - \delta')m \left( 4 \left( \frac{1}{2} \right) \right) + \left( \frac{1}{2} - \delta \right) m \left( \frac{1}{2} \right) = \frac{25}{4} m - 6\delta' m - \frac{1}{2} \delta m,
\]

which makes the total cost at least

\[
\left( \frac{25}{4} m - 6\delta' m - \frac{1}{2} \delta m \right) + 6\delta' m = \frac{25}{4} m - \frac{1}{2} \delta m,
\]

proving the lemma. \( \square \)

For any \( \epsilon' > 0 \), one can select \( \epsilon \) and \( \delta \) such that the ratio between the soundness and the completeness satisfies

\[
\frac{25}{4} - \frac{\delta}{6 + \epsilon} = \frac{25}{24} - \epsilon'.
\]

By Theorem 14 this completes the proof of Theorem 3.

6 Discussion

The usual starting point for many hardness results is the Label-Cover problem. One would wonder if this was possible in our case. Indeed, our first attempt was along this direction. However, high degree of vertices in the bipartite graph defined via this problem turns out to be a hindrance. Consider the NO case, where at most some small \( \epsilon \)-fraction of the edges can be satisfied and let the degree of the vertices on the \( V \)-side be \( d \). Among these edges, it is possible to have \( 1/\epsilon \) groups of size \( \epsilon d \) such that in each group the connections are of cheap Type I-I. Indeed, this can happen even when one selects \( 1/\epsilon \) distinct labels on a vertex \( v \in V \). As a result, the number of expensive Type I-I connections can be as low as \( 1/\epsilon \). Unfortunately, the parameters of the Label-Cover problem enforces that \( d \) is much larger than \( 1/\epsilon \), which means that one cannot get large enough cost in the NO case.

The problems we have considered circumvent this issue by simply considering much sparser bipartite graphs, i.e. the degrees of the vertices in \( V \) are either 2 or 3. This allows one to force expensive Type I-I connections such that their number is comparable to the total number of edges. An interesting point about this issue in general is that even if one can force a large number of expensive connections in the Label-Cover problem, the inapproximability factor one can prove stays at \( 17/16 \). The reason is that in an extreme case, the number of Type-I edges might be equal to the number of Type-II edges. Together with 0 connection cost between these edges, we would have an extra cost of \( 1/8 \) for each edge-component on average, again leaving us with the ratio between 2 and 17/8. This suggests that we have not lost the power of the usual PCP machinery by considering problems that reside rather down in the hierarchy. As a result, using our approach inspired by the Long Code, the ratio \( 17/16 \) seems to be the limit provided by the PCP machinery.

There are other reasons that might convince one that the bounds we have proved might as well be considered strong. The two natural LP relaxations for Steiner tree, namely the bidirected cut relaxation (BCR) and the directed-component cut relaxation (DCR) have integrality gaps at least \( 8/7 \) and \( 36/31 \) [10], respectively. In particular, the true value for BCR is believed to be very close
to the lower bound. The fact that we do not yet have a proof of this claim is quite possibly due to lack of finer algorithmic techniques. We suspect that there are much better algorithms for Steiner tree. It wouldn’t even be surprising to see that there are convex relaxations for the problem with integrality gaps closer to the bounds given in this paper.

Finally, whether the ideas in this paper can be extended to other notoriously difficult problems such as Metric TSP is an interesting open problem.

ACKNOWLEDGMENT

This work was supported by TUBITAK (Scientific and Technological Research Council of Turkey) under Project No. 112E192.

References

[1] A. Agrawal, P. N. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. SIAM J. Comput., 24(3):440–456, 1995.

[2] A. Archer, M. Bateni, M. Hajiaghayi, and H. Karloff. Improved approximation algorithms for prize-collecting Steiner tree and TSP. SIAM J. Comput., 40(2):309–332, 2011.

[3] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. J. ACM, 45(5):753–782, 1998.

[4] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. J. Comput. Syst. Sci., 54(2, part 2):317–331, 1997.

[5] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. J. ACM, 45(3):501–555, 1998.

[6] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. J. ACM, 45(1):70–122, 1998.

[7] P. Berman and V. Ramaiyer. Improved approximations for the Steiner tree problem. J. Algorithms, 17:753–782, 1994.

[8] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. Inf. Proc. Lett., 32(4):171–176, 1989.

[9] J. Byrka, F. Grandoni, T. Rothvoss, and L. Sanitá. An improved LP-based approximation for Steiner tree. In Proceedings of the 42th Annual ACM Symposium on Theory of Computing (STOC), pages 583–592, 2010.

[10] J. Byrka, F. Grandoni, T. Rothvoss, and L. Sanitá. Steiner tree approximation via iterative randomized rounding. J. ACM, 60(1), 2013.

[11] M. Chlebík and J. Chlebíková. Approximation hardness of the Steiner tree problem on graphs. In Proceedings of the 8th Scandinavian Workshop on Algorithm Theory (SWAT), pages 170–179, 2002.

[12] M. Chlebík and J. Chlebíková. The Steiner tree problem on graphs: Inapproximability results. Theor. Comput. Sci., 406(3):207–214, 2008.
[13] E.-A. Choukhmane. Une heuristique pour le problème de l'arbre de Steiner. *RAIRO Rech. Opér.*, 12:207–212, 1978.

[14] L. Fleischer, J. Könemann, S. Leonardi, and G. Schäfer. Strict cost sharing schemes for Steiner forest. *SIAM J. Comput.*, 39(8):3616–3632, 2010.

[15] M. R. Garey and D. S. Johnson. *Computers and Intractability*. W. H. Freeman and Co., 1979.

[16] E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM J. Appl. Math.*, 16(1):1–29, 1968.

[17] M. X. Goemans, A. V. Goldberg, S. A. Plotkin, D. B. Shmoys, É. Tardos, and D. P. Williamson. Improved approximation algorithms for network design problems. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 223–232, 1994.

[18] M. X. Goemans, N. Olver, T. Rothvoss, and R. Zenklusen. Matroids and integrality gaps for hypergraphic Steiner tree relaxations. In *Proceedings of the 43th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1161–1176, 2011.

[19] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995.

[20] M. Hajiaghayi and K. Jain. The prize-collecting generalized Steiner tree problem via new approach of primal-dual schema. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 631–640, 2006.

[21] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48:798–859, 2001.

[22] S. Hougardy and H. J. Prömel. A 1.598 approximation algorithm for the Steiner problem in graphs. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 448–453, 1999.

[23] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*. North-Holland, 1992.

[24] A. Iwainsky, E. Canuto, O. Taraszow, , and A. Villa. Network decomposition for the optimization of connection structures. *Networks*, 16:205–235, 1986.

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

[26] A. B. Kahng and G. Robins. *On Optimal Interconnections for VLSI*. Kluwer, 1995.

[27] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[28] M. Karpinski and A. Zelikovsky. New approximation algorithms for the Steiner tree problem. *J. Comb. Optim.*, 1(1):47–65, 1997.

[29] S. Khot, G. Kindler, E. Mossel, and R. O’Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM J. Comput.*, 37(1):319–357, 2007.

[30] B. Korte, H. J. Prömel, and A. Steger. Steiner trees in VLSI-layout. In B. Korte et al., editor, *Paths, Flows, and VLSI-Layout*, pages 185–214. Springer, 1990.

[31] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for Steiner trees. *Acta Inform.*, 15:141–145, 1981.
[32] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999.

[33] J. Plesník. A bound for the Steiner tree problem in graphs. *Math. Slovaca*, 31:155–163, 1981.

[34] H. J. Prömel and A. Steger. A new approximation algorithm for the Steiner tree problem with performance ratio 5/3. *J. Algorithms*, 36:89–101, 2000.

[35] G. Robins and A. Zelikovsky. Improved Steiner tree approximation in graphs. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 770–779, 2000.

[36] G. Robins and A. Zelikovsky. Tighter bounds for graph Steiner tree approximation. *SIAM J. Disc. Math.*, 19(1):122–134, 2005.

[37] Y. Sharma, C. Swamy, and D. P. Williamson. Approximation algorithms for prize collecting forest problems with submodular penalty functions. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1275–1284, 2007.

[38] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Math. Jap.*, 24:573–577, 1980.

[39] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[40] D. P. Williamson, M. X. Goemans, M. Mihail, and V. V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. *Combinatorica*, 15(3):435–454, 1995.

[41] A. Zelikovsky. An 11/6-approximation algorithm for the network Steiner problem. *Algorithmica*, 9:463–470, 1993.