Bachelor's Programme in Science and Technology

The Steiner Tree Problem

The logic and efficiency of solving algorithms

Vesa Ala-Laurinaho

Aalto University

BACHELOR'S THESIS

Aalto University
BACHELOR'S THESIS 2025

The Steiner Tree Problem

The logic and efficiency of solving algorithms

Vesa Ala-Laurinaho

Thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Technology.
Espoo, June 24, 2025

Supervisor: Prof. Camilla Hollanti
Advisor: Prof. Sándor Kisfaludi-Bak

Aalto University
School of Science
Bachelor's Programme in Science and Technology

Author
Vesa Ala-Laurinaho

Title
The Steiner Tree Problem

School School of Science

Degree programme Bachelor's Programme in Science and Technology

Major Mathematics **Code** SCI3029

Supervisor Prof. Camilla Hollanti

Advisor Prof. Sándor Kisfaludi-Bak

Level Bachelor's thesis **Date** June 24, 2025 **Pages** 36 **Language** English

Abstract

STEINER TREE refers to a family of optimisation problems, where the goal is to find the shortest network connecting a given set of *terminals* K. We define the Euclidean and Graph versions of the Steiner tree problem and show the corresponding methods for solving them. The first finite-time algorithm for the EUCLIDEAN STEINER TREE was presented by Melzak in 1961. The algorithm is old, and the goal of this thesis is to present this algorithm in modern terms and discuss some of the computational difficulties. In addition, we present the algorithm of Dreyfus and Wagner for the GRAPH STEINER TREE. The goal is to show that the GRAPH STEINER TREE is much faster to solve, which is an essential result for modern approximation algorithms of the EUCLIDEAN STEINER TREE. To illustrate the complexity of the algorithms, two simple examples are presented in the thesis.

Keywords Steiner Tree Problem, Euclidean Steiner Tree, Graph Steiner Tree, Melzak's algorithm

urn https://aaltodoc.aalto.fi

Aalto University
Aalto-yliopisto, PL 11000, 00076 AALTO www.aalto.fi

Tiivistelmä

Tekijä
Vesa Ala-Laurinaho

Aihe
The Steiner Tree Problem

Korkeakoulu Perustieteiden korkeakoulu

Koulutusohjelma Teknillistieteellinen kandidaattiohjelma

Pääaine Matematiikka **Pääaineen koodi** SCI3029

Valvoja Prof. Camilla Hollanti

Ohjaaja Prof. Sándor Kisfaludi-Bak

Taso Kandidaatin työ **Päivämäärä** 24.6.2025 **Sivumäärä** 36 **Kieli** englanti

Tiivistelmä

Steiner-puu-ongelma viittaa joukkoon optimointiongelmia, joissa tavoitteena on löytää lyhyin verkko, joka yhdistää annetun joukon pisteitä K. Tässä työssä määritellään Steiner-puu-ongelma Euklidiselle tasolle ja verkolle, sekä esitetään algoritmit niiden ratkaisemiseen. Lisäksi algoritmien ratkaisemiseen vaadittavien laskutoimitusten lukumäärää arvioidaan.

Vuonna 1961 Zdzislaw Alexander Melzak esitti ensimmäisen ratkaisualgoritmin Euklidiselle Steiner-puulle. Algoritmi esitellään tässä työssä käyttäen modernia termistöä ja geometrisiä rakenteita. Erityisesti halutaan osoittaa, että parhaan ratkaisun rakenteen valitseminen on haastavaa. Toinen työssä esiteltävä algoritmi on verkko-Steiner-puun ratkaiseva algoritmi, jonka Dreyfus ja Wagner esittivät vuonna 1972. Olennaisesti algoritmin ratkaisu-aika on exponentiaalinen joukon K pisteiden määrän suhteen, mutta polynominen verkon solmujen kokonaismäärän suhteen.

Tavoitteena on osoittaa, että verkko-Steiner-puu on huomattavasti helpompi ratkaista kuin Euclidinen Steiner-puu. Tämä on keskeinen tulos likiarvojen löytämiselle moderneilla menetelmillä. Erityisesti polynominen kasvu suhteessa verkon solmujen määrään on hyödyllinen ominaisuus, koska käytännön ongelmissa pisteiden K osuus on pieni verkon kaikista solmuista. Työssä on esimerkit kummankin algoritmin käytöstä havainnollistamaan algoritmien toimintaa ja laskennan vaativuutta.

Avainsanat Steiner-puu-ongelma, Melzakin algoritmi, verkko-Steiner-puu

urn https://aaltodoc.aalto.fi

Contents

1. Introduction

STEINER TREE refers to a family of optimisation problems, where the goal is to find the shortest network connecting a given set of n number of *terminals* K. The original version of the problem is the EUCLIDEAN STEINER TREE, in which the set of terminals K are points in the Euclidean plane [7]. The problem was first introduced in 1811 by Joseph Diaz Gergonne [10]. After that, the problem was forgotten and rediscovered multiple times. Although the problem is easy to understand, it is difficult to solve and is proven to be NP-hard [6]. Due to the high number of computations, the problem was mainly a theoretical curiosity until the 1960s, when computers and related mathematical theory were developed. Since then, the problem has been studied extensively, as it has a wide range of practical applications in fields such as infrastructure [13], electronics and chip design [15], and communication networks [9].

A popular modification to Euclidean Steiner is the RECTILINEAR STEINER TREE problem, where the tree T connecting K consists of only horizontal and vertical lines. In a *rectilinear Steiner arborescence*, there is given a root $r \in K$ and the path from every other terminal $a \in K$ to k in T is the shortest path. The RECTILINEAR STEINER ARBORESCENCE problem asks for the shortest rectilinear Steiner arborescence for the set K and root $r \in K$. Both of these problems can be solved in subexponential $2^{\mathcal{O}(\sqrt{n}\log n)}$ time [5].

Another modification is EUCLIDEAN BOTTLENECK STEINER TREE, where the number of Steiner points is limited to be at most some parameter k. This problem can be solved in $k^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$, which implies problem to be fixed-parameter tractable (FPT) [2].

The STEINER TREE has a closely related problem in *Graphs*, where the terminals K are a subset of the vertices V of an undirected weighted graph G, and the goal is to find the connected subgraph $H \subset G$ where $K \subseteq V(H)$

with minimum total edge weight [4].

It remains a major open problem whether a subexponential algorithm for EUCLIDEAN STEINER TREE exists. Meanwhile, the exact algorithms for the EUCLIDEAN STEINER TREE are usually too slow to use in practice, and approximations are used instead. Often, these approximations use tools from graph theory and simplifying assumptions, such as the convex position of the terminals K [14]. In an approximation scheme, the achieved solution is at most $1 + \epsilon$ times longer than the optimum for any given $\epsilon > 0$.

One commonly used approximation scheme for EUCLIDEAN STEINER TREE is Arora's algorithm from 1998, in which a rectangle containing points K is recursively partitioned into smaller rectangles, yielding a polynomial time approximation scheme [1]. This was later improved to achieve a runtime of $2^{\mathcal{O}(1/\epsilon)} n \log n$ with so-called sparsity-sensitive patching [8]. Another approximation scheme was independently found in 1998 by Joseph S.B. Mitchell, which similarly uses subdivision of the plane, finding approximations within $n \log n$ complexity [12]. With the assumption that terminals lie on a convex hull, Provan represented in 1988 a fully polynomial approximation with runtime of $\mathcal{O}(n^4/\epsilon^6)$, using a polynomial algorithm for convex planar graphs.

In this thesis, we introduce the first finite-time algorithm to the EUCLIDEAN STEINER TREE by Melzak from 1961 [10]. The algorithm is old and not written with modern terminology. The goal of this thesis is to present this algorithm in modern terms and discuss some of the computational difficulties. In addition, we present an FPT algorithm to solve the STEINER TREE in graphs following [4]. The goal is to show that the GRAPH STEINER TREE is much faster to solve, which is an essential result for modern approximation algorithms of the EUCLIDEAN STEINER TREE. We define the Euclidean and Graph versions of the Steiner tree problem and show corresponding methods for solving them. At the beginning of Chapter 2, the problems and terminology are defined more precisely. In the subsequent sections, Melzak's algorithm is presented, with a simple example of its usage shown in the appendix. In Chapter 3, an FPT algorithm to solve the STEINER TREE in graphs. In the final chapter, we have the summary and conclusions, in which we compare the above methods and analyse their usefulness in practice.

2. Melzak's algorithm for Euclidean Steiner Tree

This chapter presents the first limited-time algorithm for the original problem, EUCLIDEAN STEINER problem. We start by defining terminology and introducing the problem. Then, we present a method to solve EUCLIDEAN STEINER for three terminals and show how Melzak's algorithm uses this result iteratively to solve larger trees for known configurations. Finally, we analyse the number of possible Steiner configurations.

2.1 Geometric graph

Before introducing the Euclidean problem, we define the geometric constructs used in this chapter.

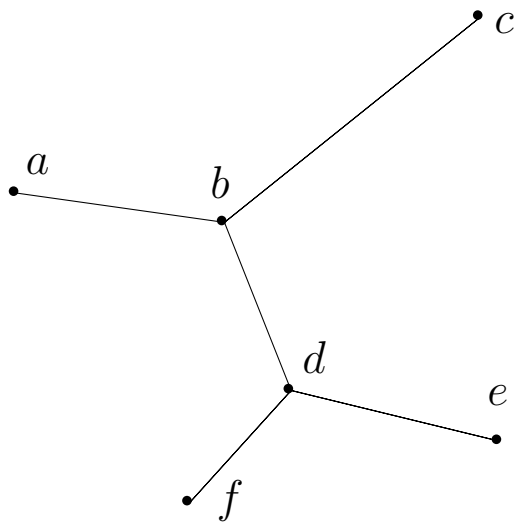
Definition 1. A *Geometric graph* (V, E, f) is a construct, where V is a set of vertices, $E \subseteq \binom{V}{2}$ is a set of edges, and $f : V \to \mathbb{R}^2$ is an injective function that maps vertices on Euclidean plane. The image of vertices $f(V)$ is a set of points, and edges are represented with straight lines that connect the corresponding points.

A geometric graph (V, E, f) is a representation of a graph (V, E) on an Euclidean plane. While a geometric graph can have crossing segments, we are mostly interested in those without them. A graph that can be represented on a plane without crossing segments is called a *planar graph*.

Definition 2. A *planar straight-line graph* (PSLG) is a geometric graph (V, E, f), where the edges E are embedded as non-crossing line segments connecting associated endpoints.

The weight of a PSLG is the sum of the lengths of its segments.

Definition 3. Given a set N of n points called *terminals* on the Euclidean plane, a PSLG $H = (V, E, f)$ connecting points N is called a *Steiner tree* H for N.



$V(T) = N \cup S$	$\{a, c, e, f, b, d\}$
E	$\{ab, cb, ed, fd, bd\}$
N	$\{a, c, e, f\}$
S	$\{b, d\}$

Figure 2.1. The Steiner tree H with terminals $N = \{a, c, e, f\}$ and Steiner points $S = \{b, d\}$ with table on the right showing the corresponding terminal-labelled graph T.

It is important to note that H may contain vertices outside of N. We call those *Steiner points* and denote the set of Steiner points as $S \subset V$ with $|S| = k$.

Definition 4. A *terminal-labelled graph* $T = (V, E)$ of a Steiner tree $H = (V, E, f)$ is a graph, where each vertex V is labelled as either terminal or Steiner-point, see Figure 2.1.

2.2 The EUCLIDEAN STEINER problem

This section presents an algorithm to find an exact solution to the EUCLIDEAN STEINER problem as presented by Hwang et al [7].

Definition 5. In the EUCLIDEAN STEINER problem, the goal is to find the Steiner Tree H for set N with the minimum weight, called the *minimum Steiner tree*.

Definition 6. A *tree* is a connected graph with two vertices connected by exactly one path.

Lemma 1. The terminal-labelled graph T corresponding to the minimum Steiner tree is a tree.

Proof. Consider that there is a cycle in the graph T. This means that there is an edge $e \in E(T)$ so that a graph T' with $V(T') = V(T)$, $E(T') = E(T) \setminus \{e\}$ would be connected. Consequently, the Steiner tree corresponding to T' would be shorter than the one corresponding to T. Thus, the terminal-labelled graph corresponding to the minimum Steiner tree is a tree. □

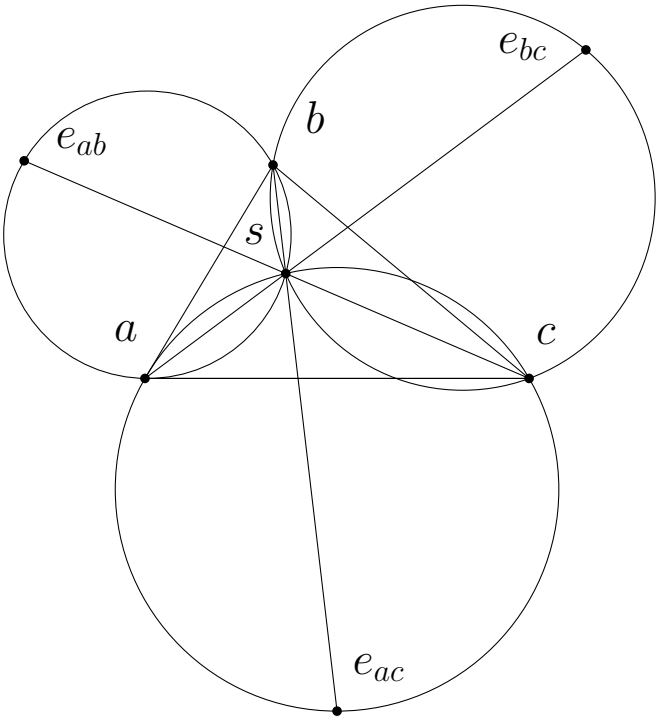


Figure 2.2. The Torricelli point s.

Consider EUCLIDEAN STEINER problem for a set of three terminals $N = \{a, b, c\}$.

In the case where no inner angle of triangle abc is over $120°$, Torricelli showed [7] that the minimum Steiner tree for $N = \{a, b, c\}$ has a Steiner point located in the intersection of three circles circumscribing the equilateral triangles constructed by the sides of the triangle abc, called the Torricelli point, denoted as s in Figure 2.2.

Consider the segment between an equilateral point e of the side of the triangle and the opposite terminal. In 1750, Simpson proved that the three segments constructed this way intersect at the Torricelli point, as shown in Figure 2.2. These segments $e_{ab}c$, $e_{ac}b$ and $e_{bc}a$ are called Simpson lines. Consequently, as all Simpson lines and circles circumscribing equilateral triangles constructed by the sides of triangle abc, only one equilateral triangle along the is needed to find the Torricelli point, as in Figure 2.5.

If there is an internal angle of at least $120°$ in the triangle abc, the point s is located outside of the triangle. In this case, [7] p.4 Heinen first observed in 1834 that the Steiner tree H for N consist of two segments that are the edges of the triangle abc adjacent to the obtuse angle, as shown in Figure 2.3. In this case, there are no Steiner points in the minimum Steiner Tree.

Select a point d so that a, b and d form an equilateral triangle, so that d and c are on opposite sides of the line ab, as in Figure 2.5. Locate the point

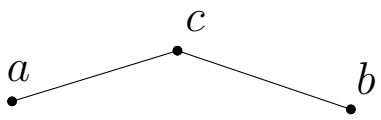


Figure 2.3. Minimum Steiner tree when triangle abc has an inner angle over $120°$.

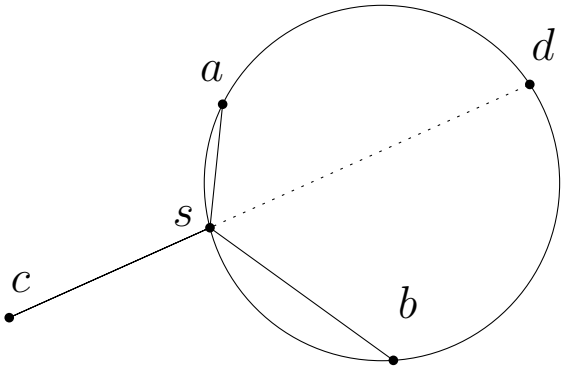


Figure 2.4. Minimum Steiner tree when triangle abc has no inner angles over $120°$.

s in the intersection of line cd and the circle circumscribing the triangle abd. If the intersection is not on the arc ab, the minimum Steiner tree does not contain a Steiner point s, and the triangle abc has an angle over $120°$.

The Steiner tree for $N = \{a, b, c\}$ consists of segments as, bs and cs as in Figure 2.4.

Lemma 2. The segments as, bs and cs meet at the angle $120°$.

Proof. Consider terminals a, b, c, the Steiner point s, and the point d forming an equilateral triangle with a and b as in Figure 2.4. The angle asd is an inscribed angle on the $120°$ arc ad on the circle, so $\angle asd = 60°$. Similarly, the angle dsb is an inscribed angle on the $120°$ arc bd, so $\angle dsb = 60°$. Thus, we get:

$$\angle csa = \angle csd - \angle asd = 180° - 60° = 120°$$
$$\angle bsc = \angle dsc - \angle dsb = 180° - 60° = 120°$$
$$\angle asb = \angle asd + \angle dsb = 60° + 60° = 120°$$

□

Now that we have found the minimum Steiner tree for three points, we can use these findings to solve the generic Euclidean Steiner tree problem.

Consider three points a, b, c so that segments ac and bc meet at the point c with angle $acb < 120°$. Then, the Torricelli point s is within the triangle abc and substituting segments ac and bc with three segments as, bs and cs results in a shorter Steiner tree, as shown in Figure 2.2. As a result, no segments in the minimum Steiner tree meet at an angle less than $120°$. Furthermore, terminals or Steiner points can have a degree of at most three without violating this requirement.

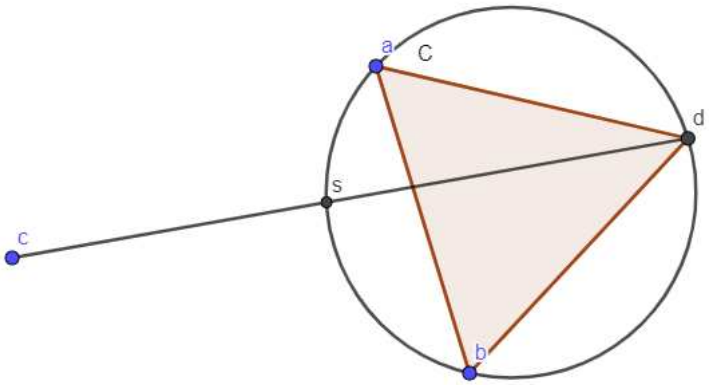


Figure 2.5. Locating the Steiner point s for $N = \{a, b, c\}$.

Lemma 3. No Steiner point has a degree of less than three in a minimum Steiner tree.

Proof. If a Steiner point s has degree one, it is a leaf and does not connect any parts of the Steiner tree H. Thus, the point s can be removed from the Steiner tree H with the segment connecting it to H, resulting in a shorter Steiner tree.

If a Steiner point s has degree two, the two segments and the point s can be substituted with a single segment connecting the points adjacent to s, resulting in a shorter Steiner tree. Thus, the Steiner points are of the degree of at least three. □

In conclusion, Steiner points have a degree of exactly three with segments meeting each other at exactly $120°$ angles.

Lemma 4. The number of Steiner points k in the minimum Steiner tree is at most $n - 2$.

Proof. In a tree, the number of edges is $|E(G)| = |V(G)| - 1$. The degree of a vertex is three for Steiner points and at least one for terminals, and each edge is connected to two vertices, so the number of edges is at least $(3k + n)/2$. It follows that

$$\begin{aligned} |E(G)| = |V(G)| - 1 &\geq (3k + n)/2 \\ n + k - 1 &\geq (3k + n)/2 \\ n - 2 &\geq k, \end{aligned} \tag{2.1}$$

□

So, the number of Steiner points is at most $n - 2$.

Definition 7. A *full Steiner tree* is a Steiner tree with exactly $n-2$ Steiner points.

Lemma 5. Each terminal is of degree one in a full Steiner tree.

Proof. If a Steiner tree H contains a terminal with a degree greater than one, the number of edges is more than $(3k+n)/2$. So it holds that

$$\begin{aligned} |E(G)| = |V(G)| - 1 &> (3k+n)/2 \\ n+k-1 &> (3k+n)/2 \\ n-2 &> k, \end{aligned} \tag{2.2}$$

and by definition, this is not a full Steiner tree. □

Definition 8. A *Steiner configuration* T for N is a terminal-labelled graph that satisfies the requirements of the minimum Steiner tree: a tree with vertices $V(T) = N \cup S$ with Steiner points S having a degree of exactly three and terminals N at most three.

Definition 9. A *full Steiner configuration* T is a Steiner configuration with $n-2$ Steiner points.

Definition 10. A *minimal Steiner tree* H for Steiner configuration T is a Steiner tree with no edges meeting in less than 120 degrees.

A minimal Steiner tree does not necessarily exist for every configuration T. As the name suggests, the minimal Steiner tree is the shortest Steiner tree H for T. In the book [7], Hwang stated in Corollary 1.3 that the Steiner tree is unique for a given Steiner configuration T. While proving this is out of scope for this thesis, it is an important result: if we can find a Steiner tree fulfilling the requirement in 10, we can be sure it is the shortest for the given Steiner configuration T.

Lemma 6. A Steiner configuration T can be uniquely decomposed into full Steiner configurations.

Proof. Consider an edge $e \in E(T)$ connecting vertices u and v.

If both $u, v \notin S$, the edge uv forms a full Steiner configuration T_i with points u and v. This is the only full Steiner configuration that uv can be a part of, as terminals have a degree of one in full Steiner trees.

If $u \in S$, it has three neighbours in T, as it is a Steiner configuration. Define a terminal-labelled graph T_i, and add the Steiner point u and its neighbours to T_i. For vertex $a \in V(T_i)$, if $a \in S$, add both new vertices

adjacent to it to $V(T_i)$. If $a \in S$, it is a terminal with the sole neighbour being the Steiner point u attached to it. Continue this for new points in $V(T_i)$ until there are no new points in T_i. The resulting terminal-labelled graph T_i is a full Steiner configuration as all the terminals have a degree of one.

Now, each edge $e \in E(T)$ can be a part of exactly one full Steiner component, so the full Steiner decomposition exists and it is unique. □

Consequently, the EUCLIDEAN STEINER problem can be solved if there is a routine to find the minimal Steiner tree for a full Steiner configuration, given that we know all the possible Steiner configurations.

2.2.1 The Melzak algorithm

In this section, we show a finite-time method to find a minimal Steiner tree H for a given full Steiner configuration T, if such a tree exists.

Theorem 1. An exact method exists to find a minimal Steiner tree for a given full Steiner configuration T.

Proof. We introduce the Melzak algorithm, which finds the minimal Steiner tree for a given full Steiner configuration $T = T_0$. A step-by-step example can be found in the appendix, see chapter A. The algorithm consists of two stages: the merging stage and the reconstruction stage. In the merging stage, two terminals adjacent to the same Steiner point are substituted with a new point in the Steiner configuration T, reducing the size. In the reconstruction stage, the Steiner points are located. Note that the locations of the terminals are known in all Steiner configurations T_i in the merging stage.

Merging stage
Consider two terminals of a configuration T *siblings* if they are adjacent to the same Steiner point s. Let a and b be siblings in T_0 and locate the point e_1 forming an equilateral triangle with them, as shown in Figure 2.6. There are two possible choices for e_i in each merging step. Substitute points a, b and s with e in T_0 so that e is connected to the third neighbour of s and is treated as terminal in the modified Steiner configuration T_1. The number of both terminals and Steiner points in T_1 are one less than in the original T_0, and so $|N_1| - |S_1| = (|N| - 1) - (|S| - 1) = |N| - |S| = 2$. Thus, T_i remains a full Steiner configuration after each merging step. The merging steps are repeated until the Steiner configuration T_{n-2} consists of

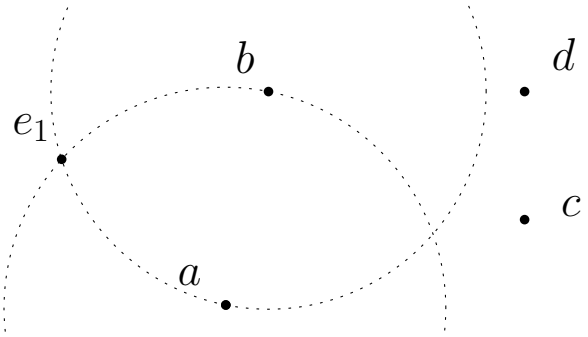


Figure 2.6. Locating the equilateral point e_1.

no Steiner points and two terminals, denoted by e_x and e_y.

Reconstruction stage
Start the reconstruction stage by connecting e_x and e_y with a segment, constructing a minimal Steiner tree for e_x and e_y.

Consider the second-to-last Steiner configuration T_{n-3} of the merging stage. Either e_x or e_y substituted two terminals of T_{n-3} in the last merging step. Denote substituted terminals as a and b, and the Steiner point as s. Now, we can restore the points a and b, locations of which are already known. The location of the Steiner point s can be found using the result of EUCLIDEAN STEINER problem for a set of three terminals as in Figure 2.5. The Steiner point might not exist if the intersection of segment e_xe_y and circle abe is not on the shorter arc ab. If s can be found, the resulting network H_{n-3} is the minimal Steiner tree for T_{n-3}.

We continue reconstructing the minimal Steiner tree for T_{i-1} by using the minimal Steiner tree for T_i, until we get to the original Steiner configuration T_0. If Steiner point does not exist at some reconstruction step, the algorithm stops, as the minimal Steiner tree does not exist for the full Steiner configuration T with the equilateral points selected in the merging stage.

If the reconstruction stage completes, the resulting tree is the minimal Steiner tree for configuration T.

□

2.2.2 Choosing the correct equilateral point

As previously shown, there are two choices for each equilateral point in the merging stage, thus there are 2^n possibilities for each full Steiner configuration. We assume that geometric operations can be completed in constant time. Then, by choosing the correct equilateral point, we can

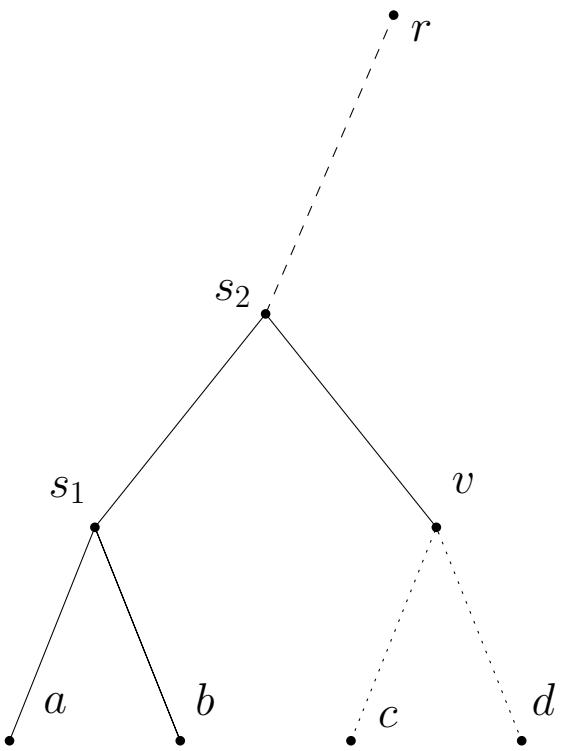


Figure 2.7. Full Steiner configuration T_0 with the root r. Vertices c and d might not exist.

reduce the time of the Melzak algorithm to linear for a given full Steiner configuration. A method for this was found by Hwang and Smith [7].

Theorem 2. An exact method exists to find a minimal Steiner tree for a given full Steiner configuration with a complexity of $\mathcal{O}(n)$.

Proof. We can brute force the cases $n = 3$ and $n = 4$, so we assume $n \geq 5$.

First, we choose an arbitrary terminal r from the full Steiner configuration T_0. Consider r as the root of T_0 and calculate the depth of other vertices corresponding to r. Let a be the deepest terminal. The locations of the corresponding points on the Euclidean plane are irrelevant to this method.

As we have $n \geq 5$, the depth of a is at least three. Now, denote the Steiner point adjacent to a as s_1 and the Steiner point adjacent to s_1 as s_2, as shown in configuration Figure 2.7. Steiner points are of degree three, so the vertex b exists and is a terminal as the terminal a is the deepest. Similarly, there is a vertex v outside of the path ra attached to the vertex s_2. If v is a terminal, the correct equilateral point e_{ab} is at the opposite side of the line ab than v as shown in Figure 2.8. If v is a Steiner point, then its children c and d are terminals, as the terminal a is the deepest. If c and d are on the same side of the line ab, the equilateral point e_{ab} is on the opposite side of the line ab as c, as shown in Figure 2.9. Then we continue as in the merging stage, substituting a, b and s_1 by e_{ab} in the

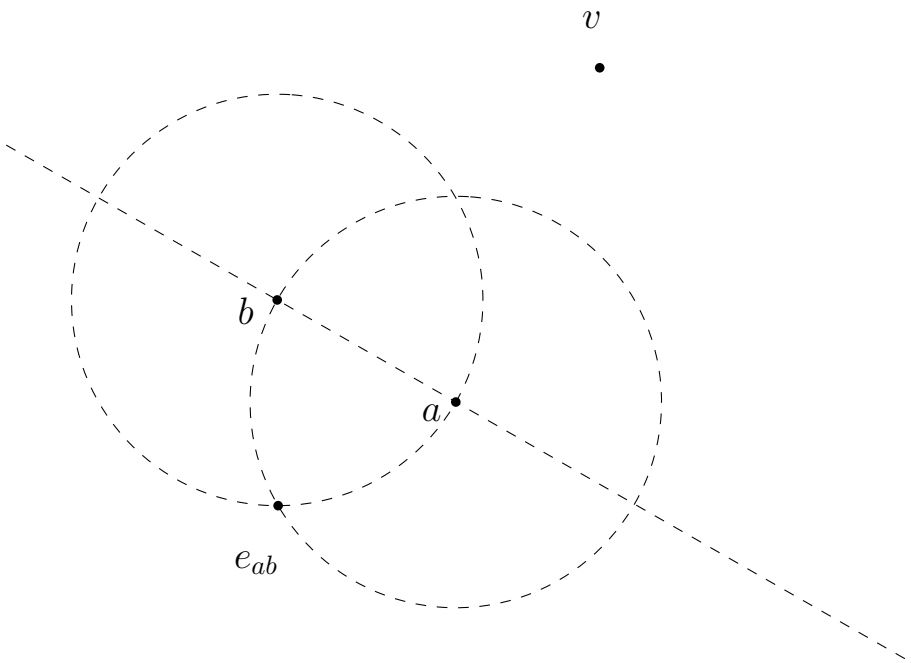


Figure 2.8. Locating the equilateral point e_{ab} when v is a terminal.

Steiner configuration T. If c and d are on the same side of the line ab, we can try to locate e_{cd}. As previously, if a and b are on the same side of line cd, the equilateral point e_{cd} is on the opposite side of the line cd as a.

If neither statement holds as the segments ab and cd intersect, there is no minimal Steiner tree for Steiner configuration T, and the algorithm stops with no solutions found.

The above is repeated for the same r with different a until only four terminals are left, after which the solution is found by brute force. The correctness and time complexity of $\mathcal{O}(n)$ is verified by Hwang, but the proof is out of scope for this thesis. □

Thus, the minimal Steiner tree for a full Steiner configuration can be found in linear time, if one exists. To solve the EUCLIDEAN STEINER problem, we need to consider all possible Steiner configurations, use the improved Melzak algorithm and choose the shortest among them.

2.2.3 The number of Steiner configurations

We present the findings of F. Hwang [7] to determine the total number of Steiner configurations.

First, we denote the number of full Steiner configurations as $f(n)$, where $n \geq 2$ is the number of terminals, as usual. Clearly, $f(2) = 1$, as the only possible Steiner configuration is just an edge connecting the two terminals. When $n > 2$, all terminals are attached to a Steiner point in a full Steiner

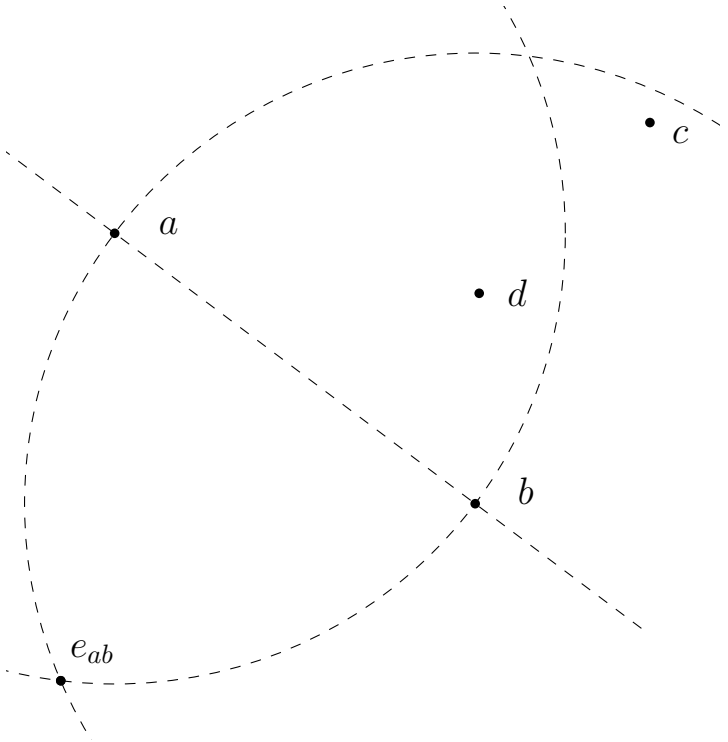


Figure 2.9. Locating the equilateral point e_{ab} when v is a Steiner point attached to terminals c and d.

configuration. Consider a full Steiner configuration with $n+1$ terminals. Now, the terminal p_{n+1} can be removed with the Steiner point it is attached to, resulting in a full Steiner configuration with n terminals. Thus, every full Steiner configuration with $n+1$ terminals can be constructed from a full Steiner configuration with n terminals by substituting one edge with a Steiner point and the new terminal. As there are $2n-3$ edges in a full Steiner configuration, we get:

$$f(n+1) = (2n-3)f(n) \tag{2.3}$$

With the condition $f(2) = 1$, Hwang solved the equation to be

$$f(n) = \frac{(2n-4)!}{2^{(n-2)}(n-2)!} \tag{2.4}$$

Then, let $F(n,k)$ be the number of Steiner configurations with n terminals and k Steiner points such that no terminal has degree three. This can be obtained from f by selecting $k+2$ terminals and a full Steiner configuration on it. Then we can add the rest of the $n-k-2$ terminals to the edges of the Steiner configuration, similarly to the construction of $f(n+1)$. There are $2k+1$ edges for the first choice, $2k+2$ for the second, until $n+k-2$ choices for the last. There are $\binom{n}{k+2}$ ways to select the first terminals, $f(k+2)$ full Steiner trees for each and $\frac{(n+k-2)!}{(2k)!}$ ways to select edges for the remaining

terminals. This gives us the value for $F(n,k)$ as

$$F(n,k) = \binom{n}{k+2} f(k+2) \frac{(n+k-2)!}{(2k)!} \tag{2.5}$$

Consider a Steiner configuration with n_3 terminals having a degree of three. These can be found using equation eq. (2.5) by labeling n_3 terminals as Steiner points. There is $\binom{n}{n_3}$ ways to select the terminals and $F(n - n_3, k + n_3)$ Steiner configurations for them. As Steiner configuration is a tree, there are $n + k - 1$ edges in it. The total degree of nodes is $2n + 2k - 2$ and as Steiner points are of degree three, the total degree of terminals is $2n - k - 2$. Then, as each terminals have a degree of at least one, the number of terminals with degree three is at most $\lfloor (n - k - 2)/2 \rfloor$. Unlike Steiner points, the order of terminals matters for the Steiner configuration, and can be distributed among Steiner points in $(k + n_3)!/k!$ ways. Thus, the number for Steiner configurations for n terminals, k Steiner points and n_3 terminals with degree of three is

$$\binom{n}{n_3} F(n - n_3, k + n_3) \frac{(k + n_3)!}{k!} \tag{2.6}$$

Denote $F(n)$ as the total number of Steiner configurations n terminals. We sum over the possible numbers of Steiner points and terminals of degree three and by using equation eq. (2.6) we obtain:

$$F(n) = \sum_{k=0}^{(n-2)} \sum_{n_3=0}^{\lfloor (n-k-2)/2 \rfloor} \binom{n}{n_3} F(n - n_3, k + n_3) \frac{(k + n_3)!}{k!} \tag{2.7}$$

In [7] Hwang calculated the values from two to eight for functions $f(n)$ and $F(n)$ as shown in Table 2.1. Hwang also noted that although $f(n)$ is

n	2	3	4	5	6	7	8
$f(n)$	1	1	3	15	105	945	10395
$F(n)$	1	4	31	360	5625	110880	2643795

Table 2.1. The number of full Steiner configurations and Steiner configurations for n terminals.

much smaller than $F(n)$, it still grows superexponentially. The function $f(n)$ can be approximated using Stirling's formula [11], as $f(n+1) = (2n-3)f(n)$ acts as factorial multiplied by 2^n. Thus, we have $f(n) \approx 2^n n! \approx 2^n \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$, in which the term n^n dominates the growth. Thus, we have $f(n) = \mathcal{O}(2^n \sqrt{2\pi n} \left(\frac{n}{e}\right)^n) \leq \mathcal{O}(n^n)$. The number of Steiner configurations is even greater.

3. Graph Steiner Tree

Let G be a connected undirected graph, and K be a subset of vertices $K \subseteq V(G)$ called terminals. The GRAPH STEINER asks for a connected subgraph H of G with minimum weight $\sum_{e \in E(H)} w(e)$ satisfying $K \subseteq V(H)$ called the *minimum Steiner tree*.

Although easier to solve than its Euclidean counterpart, the graph version is quite complex compared to similar problems such as the minimum spanning tree. The essence of the complexity is finding the most suitable subset of $V(H) \subset V(G)$. In a naive approach, we find a minimal spanning tree for each possible subset of $V(G)$ containing K and select the one with the smallest value. The minimal spanning tree can be solved in $\mathcal{O}(|E(G)| \log |V|)$ [3]. By denoting the number of vertices as $|V(G)| = n$, the number of terminals as $|K| = k$, and the number of non-terminal vertices of a tree as m, the complexity of solving the GRAPH STEINER is:

$$\sum_{m=0}^{n-k} \binom{n-k}{m} \cdot \mathcal{O}(|E(G)| \log(m+k)) \leq \mathcal{O}(2^n(|E(G)| \log n)). \tag{3.1}$$

In practical problems, the ratio of vertices that are terminals is usually small, especially if the graph is used to approximate the Euclidean Steiner problem with discrete locations for Steiner points. In this section, we represent an algorithm that can solve the Steiner problem in graphs in polynomial time concerning the total number of vertices n.

Definition 11. Given an undirected and connected graph G with positive edge weights and a set of terminals $K \subseteq V(G)$, a *Steiner Tree* is a connected subgraph H of G satisfying $K \subseteq V(H)$.

In graphs, we denote the number of terminals as $|K| = k$ and number of vertices as $n = |V(G)|$.

Definition 12. Given the graph G and set of terminals K, STEINER TREE

problem in graphs asks for the minimum Steiner Tree, that is the Steiner tree H of G with minimal cost $w(H) = \sum_{e \in E(H)} w(e)$.

Lemma 7. The minimum Steiner Tree H is a tree.

Proof. The Steiner tree is connected by definition. Assume there is a cycle in the graph. Now, at least one edge can be removed from the cycle so that H would stay connected. Because edge weights are positive, there exists a subgraph containing all vertices of H with a smaller total weight. Therefore, the minimum Steiner Tree must be a tree. □

Theorem 3. The Steiner problem in graphs can be solved a fixed-parameter tractable (FPT) algorithm with a running time of $3^k n^{\mathcal{O}(1)}$ [4].

Proof. The algorithm follows the outline presented by Cygan et al. [4, p. 131].

First, the graph G is preprocessed for the algorithm. We want to ensure that terminals are of degree one, so that the shortest paths don't include them. If $k = 1$, the corresponding minimum Steiner tree is a graph with a single vertex $v \in K$ and no edges. Therefore, assume $k > 1$. A Steiner tree for K in G exists only if the terminals K belong to the same connected component of G. If a Steiner Tree exists, the algorithm is applied to the connected component G_i of G with $K \in V(G_i)$. Without loss of generality, denote $G_i = G$ and assume G to be connected. This assures that the Steiner tree with the minimum weight exists.

Next, the graph is processed to ensure that terminals have degree 1 and that they are not neighbours of each other. This is achieved by creating a vertex t' for each terminal $t \in K$ connected with an edge tt' with a fixed weight w_0. Denote the set of new vertices as K' and the modified graph as G'. Each vertex $t' \in K'$ is of degree 1 in G' with a sole neighbour $t \in K,\ t \notin K'$.

Lemma 8. The minimum Steiner tree H for K in G corresponds one-to-one to the minimum Steiner tree H' for K' in G'.

Proof. For a Steiner Tree H for K in G, there exists a Steiner tree H'_i for K' in G', so that $V(H') = V(H) \cup K'$ and $E(H') = E(H) \cup \{tt'\}$, where $\{tt'\}$ is a set of edges connecting new terminals $t' \in K'$ to corresponding original terminals $t \in K$.

Respectively, for each Steiner tree H' for K' in G', there exists a Steiner tree H, so that $V(H) = V(H') \setminus K'$ and $E(H) = E(H') \setminus \{tt'\}$. Therefore, a

bijection exists between Steiner Trees for K in G and Steiner Trees for K' in G'.

Now, the total weight is $w(H') = w(H) + kw_0$, where the term kw_0 is independent of the selection of H. Thus, the minimum Steiner Tree for K' in G' contains the minimal Steiner Tree for K in G. □

As of Lemma 8, finding the minimum Steiner Tree for K' in G' is sufficient.

We define a dynamic programming table T as follows: for every nonempty subset $D \subseteq K'$ and every vertex $v \in V(G) = V(G') \setminus K'$, the value of entry $T[D, v]$ is the weight of the minimum Steiner tree for $D \cup \{v\}$ in G'.

Lemma 9. The weight of the minimum Steiner Tree for K' in G' is

$$\min_{v \in V(G)} T[K', v]. \tag{3.2}$$

Proof. As each terminal $t' \in K'$ is of degree 1 in G' with sole neighbour $t \notin K'$, a Steiner Tree for K' in G' contains at least one vertex $v \in V(G)$. Therefore, the minimal Steiner tree for K' in G' is the minimal Steiner tree for $K' \cup \{v\}$ in G' for some vertex $v \in V(G)$, so the weight of the minimum Steiner tree is among entries $T[K', v],\ v \in V(G)$. Additionally, every entry $T[K', v]$ is a weight for some Steiner Tree for K' in G', so the minimal Steiner tree is of weight $\min_{v \in G} T[K', v]$. □

The first step is to compute entries with $|D| = 1$, where the minimum Steiner tree is just the shortest path between the vertices v and $t' \in D$ in G. Intuitively, the minimal Steiner Tree for $D \cup \{v\}$ in G' may have a subset $D_i \subset D$ contained in a subtree H_i so that $D_i \subset V(H_i)$ and H_i is attached to the rest of the tree in some vertex u.

Lemma 10. For subsets with $D \subseteq K'$ with $|D| \geq 2$, the value of entry $T[D, v]$ can be calculated as follows:

$$T[D, v] = \min_{\substack{\emptyset \neq D' \subsetneq D, \\ u \in V(G)}} \big\{ T[D', u] + T[D \setminus D', u] + d(v, u) \big\} \tag{3.3}$$

Proof. It can be shown that the minimal Steiner Tree for $D' \cup (D \setminus D') \cup \{v\}$ in G' has the weight $\min_{u \in V(G)} \{T[D', u] + T[D \setminus D', u] + d(v, u)\}$.

First, it is shown that $T[D, v] \leq T[D', u] + T[D \setminus D', u] + d(u, v)$ for some $D' \subset D$ and $u \in V(G)$. Denote H_1 as the minimal Steiner Tree for $D' \cup \{u\}$ in G' and H_2 as the minimal Steiner Tree for $D \setminus D' \cup \{u\}$ in G', with weights $w(H_1) = T[D', u]$ and $w(H_2) = T[D \setminus D', u]$. Let P be the shortest

path from u to v, with a length of $w(P) = \sum_{e \in E(P)} w(e)$. Now the subgraph $H = H_1 \cup H_2 \cup P$ is a connected subgraph of G' with $D \subset V(H)$, i.e. H is a Steiner Tree for D in G'. It has a total weight of

$$w(H) = w(H_1 \cup H_2 \cup P) \leq w(H_1) + w(H_2) + w(P) = T[D', u] + T[D \setminus D', u] + d(u, v) \ . \tag{3.4}$$

The above holds for arbitrary choices of D' and u, thus

$$T[D, v] \leq \min_{\substack{\emptyset \neq D' \subset D, \\ u \in V(G)}} \{T[D', u] + T[D \setminus D', u] + d(u, v)\} \ . \tag{3.5}$$

Then, it suffices to show that $T[D, v] \geq T[D', u] + T[D \setminus D', u] + d(u, v)$ for some $D' \subsetneq D$ and $u \in V(G)$. Let H be the minimal Steiner Tree for $D \cup \{v\}$ in G'. Consider v as the root of the tree H. Let u be a vertex with the shortest distance to v with at least two children, potentially $u = v$. Such a vertex must exist since $|D| \geq 2$ and every terminal $t' \in D$ is of degree 1, and moreover, $u \notin K'$. Select an arbitrary child u_0 of u and decompose H into three parts as shown in Figure 3.1:

1. P is the path between u and v.
2. H_1 is a subtree of H rooted at u_0 with the edge uu_0.
3. H_2 is the rest of the tree H, that is an entire tree rooted at u, except the descendants of u_0 and the edge uu_0.

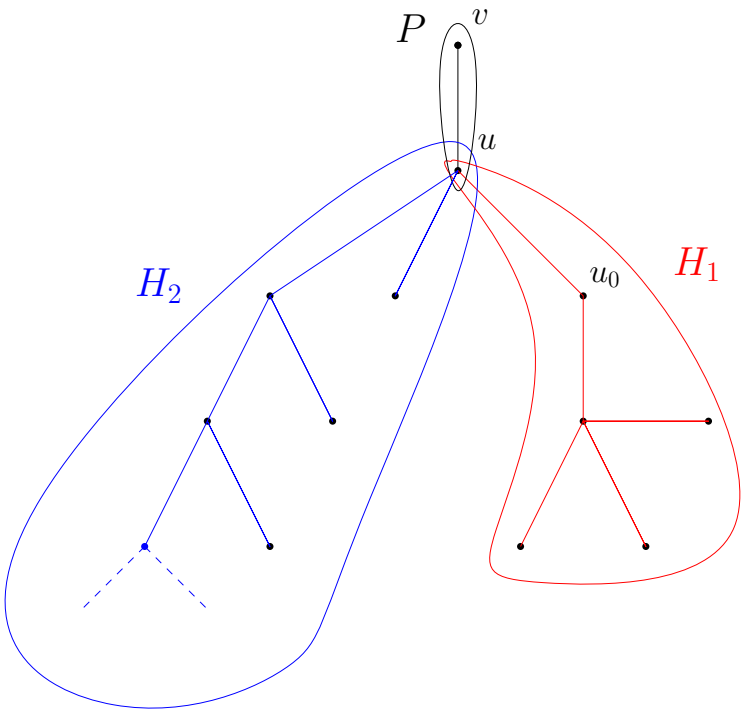


Figure 3.1. Partition of H.

As the graph H is a tree by Lemma 7, the parts are only joined in vertex

u. Let $D' = V(H_1) \cap K'$ and $D'' = V(H_2) \cap K'$. As all terminals $t' \in D$ are of degree 1, it follows $D'' = D \setminus D'$. The tree H is minimal, therefore:

1. $D' \neq \emptyset \neq D \setminus D'$, as otherwise u would have unnecessary children.
2. The weights are $w(H_1) = T[D', u]$ and $w(H_2) = T[D \setminus D', u]$.
3. The weight $w(P)$ is the shortest distance between u and v, i.e. $d(u, v)$.

As a consequence,

$$w(H) = T[D', u] + T[D \setminus D', u] + d(u, v) \geq \min_{\substack{\emptyset \neq D' \subset D, \\ u \in V(G)}} \{T[D', u] + T[D \setminus D', u] + d(u, v)\} . \tag{3.6}$$

Combining the results of eq. (3.5) and eq. (3.6) proves the lemma.

□

As stated in Lemma 9, the minimal Steiner Tree for K' in G' has a weight of $\min_{v \in V(G)} T[K', v]$, so we have the tools to solve the GRAPH STEINER TREE.

Lemma 11. The algorithm shown has a running time of $3^k n^{\mathcal{O}(1)}$.

Proof. The distance between two points can be calculated in polynomial time, for example by using Dijkstra's Algorithm [4, p. 132] which has a complexity of $\mathcal{O}(n^2)$. Calculating all the distances would then take

$$\sum_{v \in V(G)} \sum_{u \in V(G)} \mathcal{O}(n^2) = n^2 \mathcal{O}(n^2) \leq n^{\mathcal{O}(1)}$$

time, which is polynomial.

For each entry of the table $T[D, v]$, we have to consider all non-empty partitions of D with all possible roots $u \in V(G)$. We can assume that accessing the distance between vertices is a constant-time operation. Then denoting size of subset $D' \subsetneq D$ as i, the time-complexity of acquiring a value for entry $T[D, v]$ is:

$$\begin{aligned} \frac{1}{2} \sum_{i=1}^{|D|-1} \sum_{u \in V(G)} \binom{|D|}{i} &= \frac{n}{2} \sum_{i=0}^{|D|} \binom{|D|}{i} - \frac{n}{2} \binom{|D|}{0} - \frac{n}{2} \binom{|D|}{|D|} \\ &= \frac{n 2^{|D|}}{2} - \frac{n}{2} - \frac{n}{2} = n(2^{|D|-1} - 1) \leq \mathcal{O}(n 2^{|D|}) \end{aligned} \tag{3.7}$$

The total number of entries in the table T is the number of nonempty subsets $D \subseteq K'$ multiplied by all vertices $v \in V(G)$. Denote the size of a

subset D as d to obtain the complexity to fill the table as follows:

$$\begin{aligned}
&\sum_{d=1}^{k} \sum_{v \in V(G)} \binom{k}{d} \frac{1}{2} \sum_{i=1}^{d-1} \sum_{u \in V(G)} \binom{d}{i} \\
\leq &\sum_{d=1}^{k} \sum_{v \in V(G)} \binom{k}{d} (n 2^d) \\
= &\sum_{v \in V(G)} \binom{k}{d} 2^d n^2 \\
\leq &\sum_{d=1}^{k} \binom{k}{d} 2^d n^{\mathcal{O}(1)}
\end{aligned} \tag{3.8}$$

The binomial theorem states

$$(x+y)^a = \sum_{j=0}^{a} \binom{a}{j} x^{(a-j)} y^j \ . \tag{3.9}$$

Denote $x = 1$, $y = 2$, $a = k$ and $j = d$ to obtain

$$(1+2)^k = \sum_{d=0}^{k} \binom{|K|}{d} 1^{(k-d)} 2^d = \sum_{d=0}^{k} \binom{k}{d} 2^d \, . \tag{3.10}$$

Combining eq. (3.8) and eq. (3.10), the complexity can be calculated as

$$\sum_{d=1}^{k} \binom{k}{d} 2^d n^{\mathcal{O}(1)} \leq 3^k n^{\mathcal{O}(1)} \tag{3.11}$$

□

This concludes the proof of Theorem 3. □

4. Conclusion

STEINER TREE refers to a family of optimisation problems, where the goal is to find the shortest network connecting a given set of n number of *terminals* K. We have presented methods to solve two formulations of this problem: the Euclidean Steiner Tree and Graph Steiner Tree.

For the Euclidean Steiner Tree, we have written the Melzak's algorithm using modern terminology and geometric constructs. With a slight alteration, the complexity is reduced to linear with respect to the number of terminals for a given Steiner configuration. However, the superexponential nature causes the total complexity of the method to be of order $n^{\mathcal{O}(n)}$.

In contrast, the FPT algorithm presented for the Graph Steiner Tree has a complexity that is polynomial with respect to the number of total points and only of order 3^k. While this is still exponential, it is a lot faster. Additionally, this is a huge improvement compared to the naive approach of $\mathcal{O}(2^n)$.

When the Euclidean Steiner Tree is approximated, the infinite number of possible Steiner point locations is usually made discrete with some divisions of the Euclidean plane. When graph algorithms are used in those settings, the number of possible Steiner points is usually much larger than the number of terminals, thus, the algorithm having a polynomial complexity with respect to the total number of vertices in a graph is highly beneficial.

Simple examples of the two described algorithms are constructed to illustrate their performance, and these are presented in the Appendix.

References

[1] Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, September 1998.

[2] Sayan Bandyapadhyay, William Lochet, Daniel Lokshtanov, Saket Saurabh, and Jie Xue. *Euclidean Bottleneck Steiner Tree is Fixed-Parameter Tractable*, pages 699–711.

[3] Jiayu Chen. The analysis and application of prim algorithm, kruskal algorithm, boruvka algorithm. *Applied and Computational Engineering*, 19:84–89, 10 2023.

[4] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015.

[5] Fedor V. Fomin, Daniel Lokshtanov, Sudeshna Kolay, Fahad Panolan, and Saket Saurabh. Subexponential algorithms for rectilinear steiner tree and arborescence problems. *ACM Trans. Algorithms*, 16(2), March 2020.

[6] M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing steiner minimal trees. *SIAM Journal on Applied Mathematics*, 32(4):835–859, 1977.

[7] F. K. Hwang and Dana S. Richards. Steiner tree problems. *Networks*, 22(1):55–89, 1992.

[8] Sándor Kisfaludi-Bak, Jesper Nederlof, and Karol Węgrzycki. A gap-eth-tight approximation scheme for euclidean tsp. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 351–362, 2022.

[9] Moyukh Laha and Raja Datta. A steiner tree based efficient network infrastructure design in 5g urban vehicular networks. *Computer Communications*, 201:59–71, 2023.

[10] Doreen A. Thomas Marcus Brazil, Ronald L. Graham and Martin Zachariasen. On the history of the euclidean steiner tree problem. *Archive for History of Exact Sciences*, 68(3):327–354, 2014.

[11] Peter R. Mercer. *Stirling's Formula*, pages 123–132. Springer Nature Switzerland, Cham, 2023.

[12] Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999.

[13] Chintan D. Patel, Tarun Kumar Tailor, Sunil Kumar Shukla, Samyak Shah, and Swapnil N. Jani. Steiner tree-based design of communication infrastructure with co-optimizing the pmu placement for economical design of wams. *IEEE Transactions on Instrumentation and Measurement*, 71:1–11, 2022.

[14] J. Scott Provan. Convexity and the steiner tree problem. *Networks*, 18(1):55–72, 1988.

[15] Hao Tang, Genggeng Liu, Xiaohua Chen, and Naixue Xiong. A survey on steiner tree construction and global routing for vlsi design. *IEEE Access*, 8:68593–68622, 2020.

A. Example of Melzak FST algorithm

Given four points $N = \{a, b, c, d\}$ as shown in Figure A.1, let us use the Melzak algorithm to find minimal Steiner tree associated Steiner configuration T so that $V(T) = \{a, b, c, d, s_1, s_2\}$ and $E(T) = \{as_1, bs_1, s_1s_2, cs_2, ds_2\}$, with terminals $N = \{a, b, c, d\}$ and Steiner points $S = \{s_1, s_2\}$. This is a full Steiner configuration as $|N| - |S| = 2$. In each merging step, both the number of Steiner points and the number of terminals are reduced by one, so the associated configuration T_i is always a full Steiner configuration.

A.1 Merging stage

There are only four points in N, so choosing the equilateral points is straight-forward and no backtracking is needed. Select siblings a and b and the corresponding Steiner point s_1. Now we can locate the equilateral point e_1 as shown in Figure A.2. We substitute the original a, b and s_1 with e_1 in associated Steiner configuration T_1, so we have $V(T_1) = \{e_1, c, d, s_2\}$ and $E(T_1) = \{e_1s_2, cs_2, ds_2\}$ with $N_1 = \{e_1, c, d\}$ and $S_1 = \{s_2\}$.

Then we select points e_1, c and the corresponding Steiner point s_2. Again, the 3-point algorithm is used to find the equilateral point e_2 as shown in Figure A.3. Now we again substitute the original points with e_2 so that

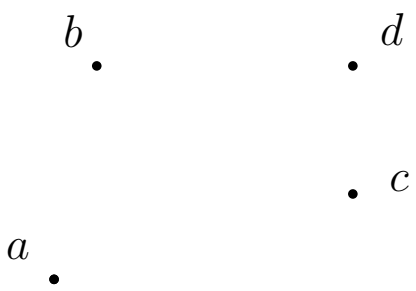


Figure A.1. The points N.

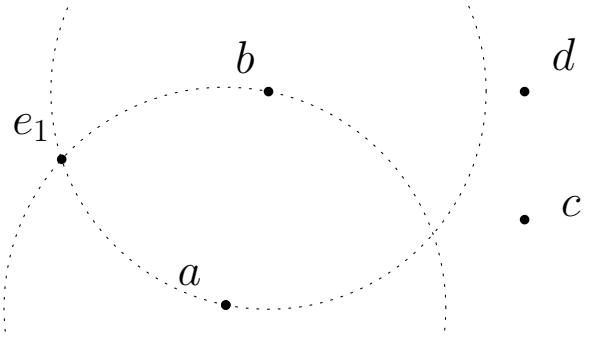


Figure A.2. Locating the equilateral point e_1.

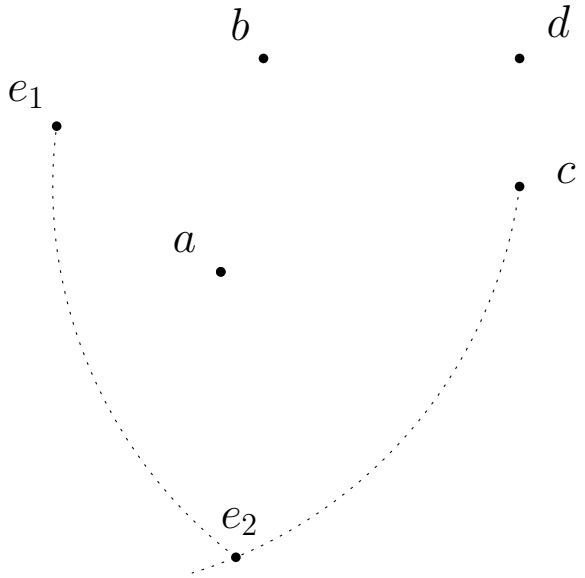


Figure A.3. Locating he equilateral point e_2.

we have $V(T_2) = \{e_2, d\}$ and $E(T_2) = \{e_2 d\}$. The summary of the merging stage is shown in Table A.1.

T_i	$V(T_i)$	N_i	S_i	$E(T_i)$
$T = T_0$	$\{a, b, c, d, s_1, s_2\}$	$\{a, b, c, d\}$	$\{s_1, s_2\}$	$\{as_1, bs_1, s_1s_2, cs_2, ds_2\}$
T_1	$\{e_1, c, d, s_2\}$	$\{e_1, c, d\}$	$\{s_2\}$	$\{e_1s_2, cs_2, ds_2\}$
T_2	$\{e_2, d2\}$	$\{e_2, d2\}$	$\emptyset$	$\{e_2d\}$

Table A.1. Evolution of the Steiner configuration in the merging stage

A.2 Reconstruction stage

Finally, the minimal Steiner tree for points e_2 and d is just the segment between them, as in Figure A.4.

We start reconstruction by using Steiner configuration T_1, by substituting the point e_2 with a Steiner point s_2 and terminals c and e_1, as shown in Figure A.5. We draw a circle around the points e_2, e_1 and c. The point s_2 is located in the intersection of the segment e_2d and the circle. Notably, if

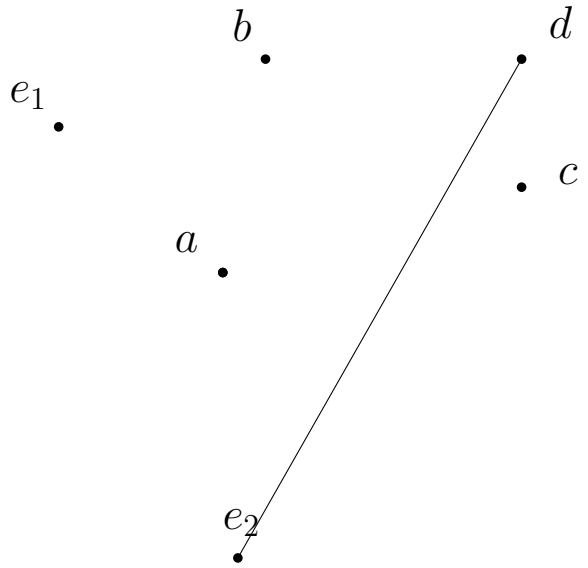


Figure A.4. The Steiner tree for $\{e_2, d\}$.

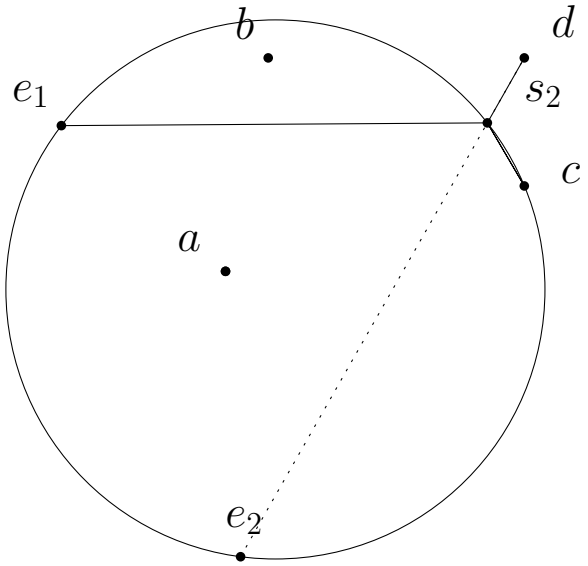


Figure A.5. Locating the Steiner point s_2.

the point s_2 were not in the arc ce_2, then the minimal Steiner tree would not exist for the given Steiner configuration T. Then, we substitute the segment e_2d with segments ds_2, s_2c and s_2e_1.

Then, the point e_1 is substituted by a, b and s_1, as shown in Figure A.6, and we proceed similarly to the previous step. We have now found the minimal Steiner tree for N with full Steiner configuration T, as shown in Figure A.7.

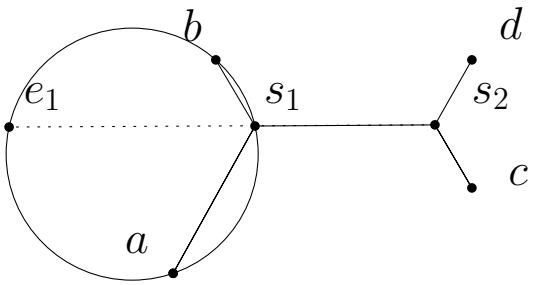


Figure A.6. Locating the Steiner point s_1.

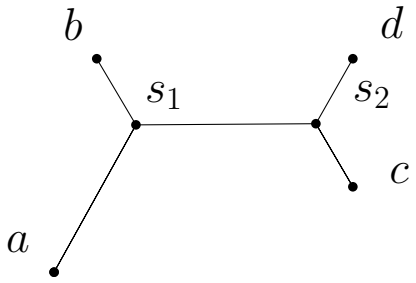


Figure A.7. The minimal Steiner tree for N and Steiner configuration T.

B. Example of Graph algorithm

For this example, utilise the dynamic algorithm to find the minimum cost Steiner tree for the points $K = \{b, c, d\}$ in the following graph G:

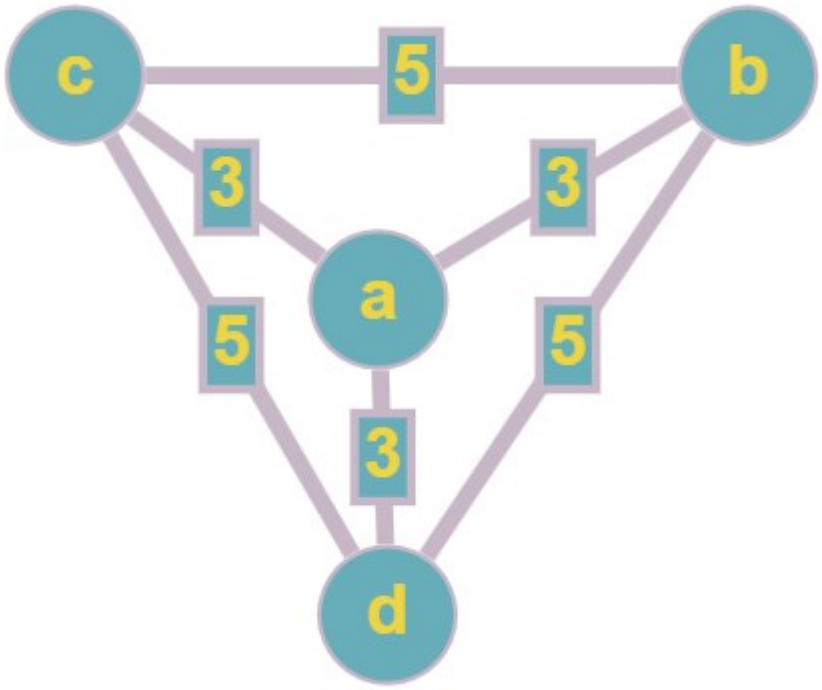


Figure B.1. An example graph G.

First process the graph, so that each $t \in K$ has a terminal $t' \in K'$ attached to it, resulting in a graph G' as in Figure B.2.

As stated in Lemma 8, solving the minimum Steiner tree for this graph also solves the original graph.

Start filling the table with the entries $T[D, v]$, where $|D| = 1$, as in Table B.1.

Then for each entry $T[D, v]$, where $|D| = 2$, calculate the following:

$$\min_{u \in V(G') \setminus K', \emptyset \neq D' \subset D} \{T(D', u) + T(D \setminus D', u) + dist(v, u)\} \tag{2.1}$$

For example, take entry $T[\{b', c'\}, d]$ as in Figure B.3. There is only one

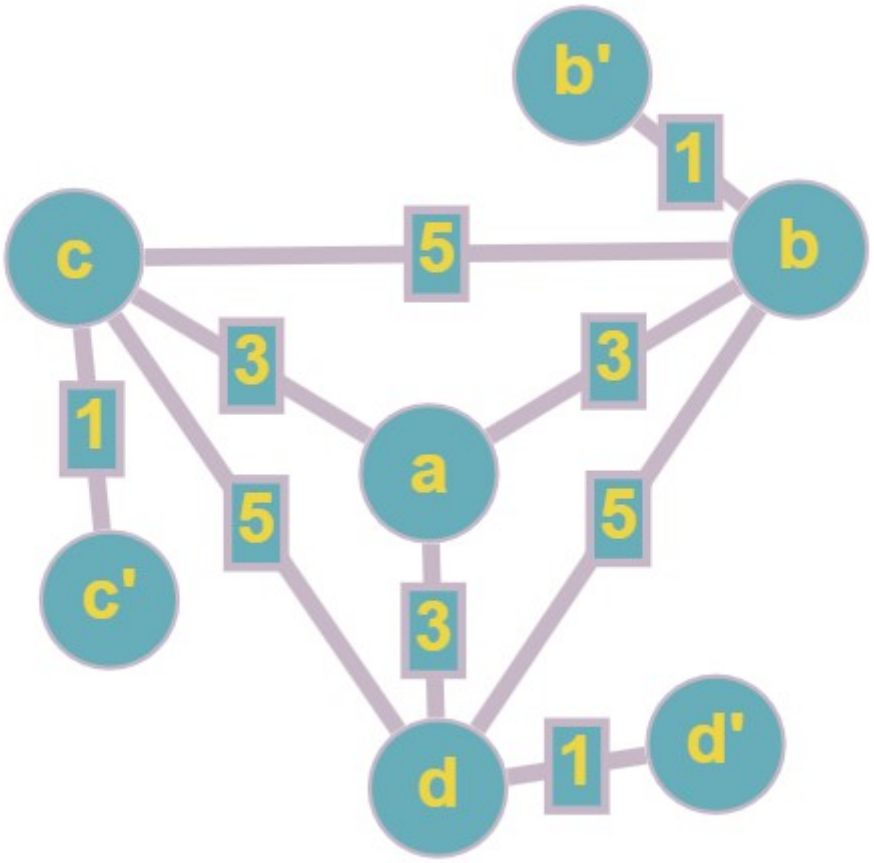


Figure B.2. Graph G' with terminal set K'.

	a	b	c	d
$\{b'\}$	4	1	6	6
$\{c'\}$	4	6	1	6
$\{d'\}$	4	6	6	1

Table B.1. The initial step.

way to partition D into non-empty parts, so there are four possibilities, one for each vertex $v \in V(G)$:

$$T(\{b'\}, a) + T(\{c'\}, a) + \textbf{dist}(a, d) = 4 + 4 + 3 = 11$$
$$T(\{b'\}, b) + T(\{c'\}, b) + \textbf{dist}(b, d) = 1 + 6 + 5 = 12$$
$$T(\{b'\}, c) + T(\{c'\}, c) + \textbf{dist}(c, d) = 6 + 1 + 5 = 12$$
$$T(\{b'\}, d) + T(\{c'\}, d) + \textbf{dist}(d, d) = 6 + 6 + 0 = 12$$

So in this instance, we have $T[\{b', c'\}, d] = 11$. Calculate the rest of the entries to obtain Table B.2.

For the final step, there are three partitions for set $\{b', c', d'\}$: $\{b', c'\}$, $\{b', d'\}$, and $\{c', d'\}$. Thus, there are $3 \times 4 = 12$ expressions per table entry

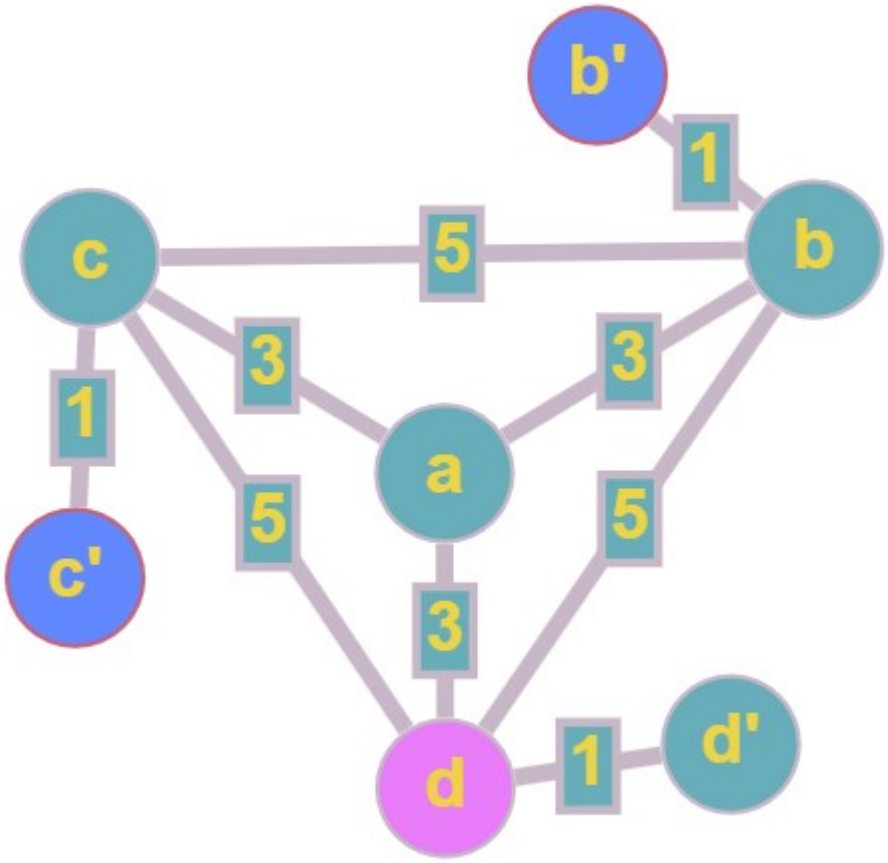


Figure B.3. Evaluation of entry $[T, v]$ with $T = \{b', c'\}$ and $v = d$.

	a	b	c	d
$\{b'\}$	4	1	6	6
$\{c'\}$	4	6	1	6
$\{d'\}$	4	6	6	1
$\{b', c'\}$	8	7	7	11
$\{b', d'\}$	8	7	11	7
$\{c', d'\}$	8	11	7	7
$\{b', c', d'\}$				

Table B.2. Table T after second step.

to consider. For $T[\{b', c', d'\}, a]$, we have:

$$T(\{b', c'\}, a) + T(\{d'\}, a) + \mathbf{dist}(a, a) = 8 + 4 + 0 = 12$$
$$T(\{b', c'\}, b) + T(\{d'\}, b) + \mathbf{dist}(b, a) = 7 + 6 + 3 = 16$$
$$T(\{b', c'\}, c) + T(\{d'\}, c) + \mathbf{dist}(c, a) = 7 + 6 + 3 = 16$$
$$T(\{b', c'\}, d) + T(\{d'\}, d) + \mathbf{dist}(d, a) = 11 + 1 + 3 = 15$$
$$T(\{b', d'\}, a) + T(\{c'\}, a) + \mathbf{dist}(a, a) = 8 + 4 + 0 = 12$$
$$T(\{b', d'\}, b) + T(\{c'\}, b) + \mathbf{dist}(b, a) = 7 + 6 + 3 = 16$$
$$T(\{b', d'\}, c) + T(\{c'\}, c) + \mathbf{dist}(c, a) = 11 + 1 + 3 = 15$$
$$T(\{b', d'\}, d) + T(\{c'\}, d) + \mathbf{dist}(d, a) = 7 + 6 + 3 = 16$$
$$T(\{c', d'\}, a) + T(\{b'\}, a) + \mathbf{dist}(a, a) = 8 + 4 + 0 = 12$$
$$T(\{c', d'\}, b) + T(\{b'\}, b) + \mathbf{dist}(b, a) = 11 + 1 + 3 = 15$$
$$T(\{c', d'\}, c) + T(\{b'\}, c) + \mathbf{dist}(c, a) = 7 + 6 + 3 = 16$$
$$T(\{c', d'\}, d) + T(\{b'\}, d) + \mathbf{dist}(d, a) = 7 + 6 + 3 = 16$$

And thus, the value for the entry $T[\{b', c', d'\}, a]$ is 12. For entry $T[\{b', c', d'\}, B]$ we have:

$$T(\{b', c'\}, a) + T(\{d'\}, a) + \textbf{dist}(a, b) = 8 + 4 + 3 = 15$$
$$T(\{b', c'\}, b) + T(\{d'\}, b) + \textbf{dist}(b, b) = 7 + 6 + 0 = 13$$
$$T(\{b', c'\}, c) + T(\{d'\}, c) + \textbf{dist}(c, b) = 7 + 6 + 5 = 18$$
$$T(\{b', c'\}, d) + T(\{d'\}, d) + \textbf{dist}(d, b) = 11 + 1 + 5 = 17$$
$$T(\{b', d'\}, a) + T(\{c'\}, a) + \textbf{dist}(a, b) = 8 + 4 + 3 = 15$$
$$T(\{b', d'\}, b) + T(\{c'\}, b) + \textbf{dist}(b, b) = 7 + 6 + 0 = 13$$
$$T(\{b', d'\}, c) + T(\{c'\}, c) + \textbf{dist}(c, b) = 12 + 1 + 5 = 18$$
$$T(\{b', d'\}, d) + T(\{c'\}, d) + \textbf{dist}(d, b) = 7 + 6 + 5 = 18$$
$$T(\{c', d'\}, a) + T(\{b'\}, a) + \textbf{dist}(a, b) = 8 + 4 + 3 = 15$$
$$T(\{c', d'\}, b) + T(\{b'\}, b) + \textbf{dist}(b, b) = 11 + 1 + 0 = 12$$
$$T(\{c', d'\}, c) + T(\{b'\}, c) + \textbf{dist}(c, b) = 7 + 6 + 5 = 18$$
$$T(\{c', d'\}, d) + T(\{b'\}, d) + \textbf{dist}(d, b) = 7 + 6 + 5 = 18$$

So, the final table T is following:

	a	b	c	d
$\{b'\}$	4	1	6	6
$\{c'\}$	4	6	1	6
$\{d'\}$	4	6	6	1
$\{b', c'\}$	8	7	7	12
$\{b', d'\}$	8	7	12	7
$\{c', d'\}$	8	12	7	7
$\{b', c', d'\}$	12	12	12	12

Table B.3. Final minimal tree.

In the table above, the minimum Steiner tree for the modified graph G' is 12. The Steiner tree for the original graph G weights $12 - 1 \cdot |K| = 9$. The final table has the same values for all vertices $V(G)$, as every vertex is included in the final Steiner tree.