



# Anton de Kom Universiteit van Suriname Bibliotheek

Universiteitscomplex, Leysweg 86, Paramaribo, Suriname, Postbus 9212  
Telefoon (597)464547, Fax (597)434211, E-mail: [adekbib@uvs.edu](mailto:adekbib@uvs.edu)

## APPROVAL

NAAM: ..... *Shalen Boeja* .....

verleent wel / ~~niet~~ aan de AdeKUS kosteloos de niet-exclusieve toestemming om haar / zijn Drs. / B.Sc. / M.Sc.  
afstudeerscriptie online beschikbaar te stellen aan gebruikers binnen en buiten de AdeKUS.

Plaats en datum, ..... *Paramaribo, 24-10-2025* .....

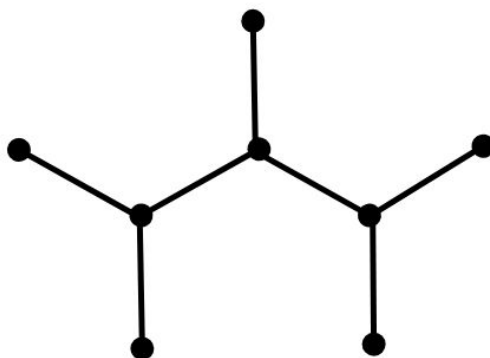
Handtekening ..... *Boeja S* .....



ANTON DE KOM UNIVERSITEIT VAN SURINAME  
FACULTEIT DER WIS- EN NATUURKUNDIGE WETENSCHAPPEN

---

## On the Number of Distinct Topologies in Steiner Minimal Trees with $n$ Terminals



Faculteit : FWNW  
STUDIERICHTING : WISKUNDE  
SAMENSTELLER : Shalen Boeja  
BEGELEIDER : PROF. DR. P.O. DE WET  
DATUM : October 2025

## Foreword

For my bachelor thesis, I had the privilege of studying Steiner trees, focusing specifically on the number of distinct hexagonal tree topologies that can be found for a given number of terminals. This study proved to be both challenging and highly educational. I express my sincere gratitude to Dr. P.O. de Wet, who guided and supervised me throughout this process, as well as to everyone who motivated and supported me, giving me the encouragement necessary to bring this work to completion.

### **Declaration of Own Work:**

I hereby declare that the submitted thesis is my own work.

Student Name: Shalen Boeja

Date: 2-10-2025

Signature:



## Abstract

This thesis explores minimal Steiner trees, which are spanning trees of some finite vertex set  $S$  in a metric space such that  $S \supseteq V$  and each vertex in  $S \setminus V$  has degree at least 3. These trees connect  $n$  terminal points using  $n-2$  Steiner points. Each Steiner point connects exactly three edges at  $120^\circ$  angles, while terminal points connect to only one edge. Considering these properties it is established that MSTs are hexagonal trees. The main goal is to find and count all the different topologies, these trees can have for a given number of terminals.

An interesting finding is that hexagonal trees can be isomorphic and still have different topologies. Because of this using isomorphism was insufficient for determining the exact number of distinct topologies. Instead, it was used to provide a lower bound.

For up to 8 terminal points, all the different tree topologies were counted manually using a special method. For more than 8 terminals, the problem becomes too complicated to solve by hand. Instead, a computer program was used to count trees based on isomorphism, giving a lower estimate.

This study helps us understand the variety and complexity of these hexagonal Steiner trees. The findings can be useful in areas such as the design of efficient networks, city planning, and other situations where connections need to be as simple and cost-effective as possible.

# Contents

- Foreword** **1**
- Abstract** **2**
- 1 Introduction** **4**
- 2 Theoretical Framework** **5**
  - 2.1 Historical background . . . . . 5
  - 2.2 Basic Concepts . . . . . 6
  - 2.3 Hexagonal Trees . . . . . 10
    - 2.3.1 Angle Condition at Steiner Points . . . . . 10
- 3 Counting trees** **14**
  - 3.1 Cayley’s Theorem . . . . . 15
  - 3.2 Full unrooted binary trees . . . . . 19
  - 3.3 Catalan numbers and future research . . . . . 20
- 4 Methodology** **22**
  - 4.1 Code Overview . . . . . 23
- 5 Results** **25**
- 6 Discussion & Conclusion** **30**
- References** **31**
- A Appendix** **33**
  - A.1 Python Code for Generating Lower Bound . . . . . 33
  - A.2 Code output for various  $n$  . . . . . 36

# 1 Introduction

In various fields such as network design, road planning, and circuit layout, it is often necessary to connect a set of points in the most efficient way possible. A solution to this is the *Minimum Spanning Tree* (MST), which connects all points with the minimal total edge length. However, the MST is limited to use only the given points as vertices. In many real-world scenarios, allowing additional points can lead to significantly shorter total connections. This gives rise to the *Steiner Minimal Tree* (SMT) problem.

The SMT problem asks: given a set of terminals in a metric space, what is the shortest possible network of edges connecting them, where additional Steiner points are allowed to minimize total length? Unlike MSTs, SMTs are much harder to compute and analyze. They are known to be NP-hard in general. And their structure is more constrained: in the Euclidean plane, for example, all Steiner points in a minimal solution have degree 3, and the edges meet at  $120^\circ$  angles.

This research focuses on the number of distinct topologies SMTs can have. Specifically, the main research question is as follows:

Given a number of terminals  $n$ , how many distinct topologies are possible for a Steiner Minimal Tree?

To explore this question, we consider full steiner trees, which we define to be hexagonal trees. The hexagonal structure arises when minimizing lengths and using the 120 degree angle condition. Thus, we also consider the subquestion:

Why are Steiner Minimal Trees considered hexagonal trees?

To answer the research question, a combination of theoretical and computational approaches are used. First, relevant definitions from graph theory and Steiner trees are introduced. A literature review is conducted to understand the known properties of SMTs. Then, a custom algorithm is developed in Python to generate all unique tree topologies that satisfy the conditions of minimal hexagonal trees.

The structure of this thesis is as follows:

- Chapter 2 presents the necessary background on graph theory, spanning trees, and Steiner trees and the geometric constraints that leads us to hexagonal tree structures.
- Chapter 3 discusses various ways of counting graphs that are relevant to our research.
- Chapter 4 describes the algorithm developed to generate and filter distinct topologies and the python code used for the lower bound of the generated topologies.
- Chapter 5 presents the results obtained for the first 10 values of  $n$ , and compares them to the lowerbound.
- Chapter 6 presents the discussion, conclusion and suggestions for future research.
- We then conclude the thesis, respectively, with a reference list and appendices.

## 2 Theoretical Framework

### 2.1 Historical background

The first known person who mentioned and analyzed the Euclidean Steiner Tree problem was the French Mathematician Joseph Diaz(1771-1859) in 1811. He published his own mathematics journal known as the *Annales de Gergonne*, which was focussed on projective and algebraic geometry. One of the problems published in this journal was the Fermat Torricelli problem, including a version that we now call the Steiner tree problem.

The Fermat Torricelli problem is described as follows, given three points in the plane, find a point such that the sum of its Euclidean distances to the three given points is minimal. This problem was solved by Evangelista Torricelli and was named the *The Fermat-Torricelli problem* (Fermat Point Explained, n.d.).

Following are a few relevant problems that were introduced in volume 1 of the *Annales de Gergonne*.

“An engineer wishes to establish a communication between three cities, not located in a straight line, by means of a network composed of three branches, leading at one end to the three cities, and meeting at the other end at a single point between these three cities. The question is, how can one locate the point of intersection of the three branches of the network, so that their total length is as small as possible?” (p.196).

— (Brazil, Graham, Thomas, Zachariasen, 2014)

This is the Fermat-Torricelli problem in an engineering perspective. In the footnote of this problem it is said that this problem can be generalized by asking the question: how can one determine a point on a plane, whose distances to a number of arbitrary points in this plane is minimal?. It’s also added that one can even extend to points positioned in any manner in the plane. Meaning the footnote broadens this problem to more than 3 given points, with only one intersection point.

Another relevant problem in this journal is the following:

A number of cities are located at known locations on a plane; the problem is to link them together by a system of canals whose total length is as small as possible.

— (Brazil, Graham, Thomas, Zachariasen, 2014)

According to the footnote in this journal, one can assume that the canal is curved. In the first problem the number of branches of the network must equal the number of cities to which they lead on the one hand, and it is required, on the other hand, that these branches meet in the network at a single point; here, on the contrary, this condition is not required, and one should not enforce it if it does not result in a minimum.

This is the first known statement of the Steiner Tree problem. The footnote here clearly allows for more than one single point where they meet. Which is a strong remark that differs this problem from the Fermat-Torricelli problem. When we read further we find the first abstract statement of the steiner tree problem, under a ten page article titled *Resolues(Resolved)*. This page adresses purely geometric solutions to 11 problems. The steiner tree problem is the eleventh one and is stated as:

”PROBLEM XI. Connect any number of given points by a system of lines whose total length is as small as possible.”

The next known mention of the Steiner tree problem was by Gallicus in 1819, in *The Mathematical Repository*, a journal of mathematical problems and essays published in England. Followed by Karl Bopp and Eduard Hoffmann from 1879 and 1890, respectively (Brazil, Graham, Thomas & Zachariasen, 2014).

After Hoffmann, interest in the Steiner Tree problem seemed to have vanished until the paper of Vojtěch Jarník and Miloš Kössler (Jarník and Kössler, 1934). They developed notable and modern treatment of the problem in the Euclidean plane but also for higher dimensions. One of their interesting contributions was showing that a minimum Steiner tree for  $n = 6$  or  $n = 13$  is similar to a minimum

spanning tree on the same set of vertices without Steiner points. The remaining set of vertices ( $n = 7, 8, 9, \dots, 12$ ) were completed by Du et al (1978) where it was shown that the same applies to these cases.

One of the most influential works on the Steiner tree problem is the book by Courant and Herbert Robbins *What is Mathematics?* (Courant and Robbins, 1941). In this book the problem is introduced in Chapter VII under Steiner's Problem. It is described as connecting three villages by a system of roads of minimal total length, while treating the Fermat-Torricelli problem as a special case. They highlight some key properties of minimal networks, such as the fact that edges meet at an angle of  $120^\circ$  in Steiner points and that solutions are not necessarily unique. Courant and Robbins credited the origin of the problem to Jakob Steiner, but later research suggests that his role in developing the Fermat-Torricelli problem was relatively minor. Much of the foundational work had already been done by earlier mathematicians. This shows how presentations of the Steiner problem, such as that by Courant and Robbins, were based on limited historical information (Brazil, Graham, Thomas & Zachariasen, 2014).

## 2.2 Basic Concepts

**Definition 2.1** (Graph). A graph  $G$  is an ordered pair  $(V, E)$ , where  $V$  is some set, and  $E$  is a set of 2-point subsets of  $V$ . The elements of the set  $V$  are called the vertices of the graph  $G$  and the elements of  $E$ , edges of  $G$  (Matoušek & Nešetřil, 2008).

To make it clear that a graph  $G$  is defined by its vertices  $V$  and edges  $E$ , we represent it as  $G = (V, E)$ . The vertex set of  $G$  is written as  $V(G)$ , while the edge set is denoted by  $E(G)$ .

Let  $G$  and  $G'$  be graphs. We say that  $G$  is a *subgraph* of  $G'$  if  $V(G) \subseteq V(G')$  and  $E(G) \subseteq E(G')$ . Consider the vertices  $v_0, v_1, v_2, \dots, v_t$  of  $G$  that are all distinct, and let  $e_i = \{v_{i-1}, v_i\} \in E(G)$  for each  $i = 1, 2, \dots, t$ . A *path* in  $G$  is then the following sequence

$$(v_0, e_1, v_1, \dots, e_t, v_t).$$

If this sequence ends at the starting vertex  $v_0$ , the path is called a *cycle*, represented as

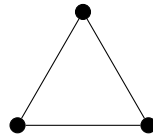
$$(v_0, e_1, v_1, \dots, e_t, v_t, v_0).$$

We say that a graph  $G$  is *connected* if for any two vertices in  $G$  there exists a path from  $x$  to  $y$ .

We introduce several types of specific graphs, as they will be referred to throughout the remainder of this thesis as their notation and terminology have become standard.

*The complete graph  $K_n$ :*

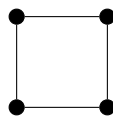
$$V = \{1, 2, 3, \dots, n\} \quad E = \binom{V}{2}$$



$K_3$

*The cycle  $C_n$ :*

$$V = \{1, 2, \dots, n\}, \quad E = \{\{i, i+1\} : i = 1, 2, \dots, n-1\} \cup \{\{1, n\}\}.$$



$C_4$

**Definition 2.2.** Two graphs  $G = (V, E)$  and  $G' = (V', E')$  are called *isomorphic* if there exists a bijection  $f : V \rightarrow V'$  such that

$$\{x, y\} \in E \quad \text{if and only if} \quad \{f(x), f(y)\} \in E'$$

holds for all  $x, y \in V, x \neq y$ . Such a function  $f$  is called an *isomorphism* of the graphs  $G$  and  $G'$ . The fact that  $G$  and  $G'$  are isomorphic is written as  $G \cong G'$ .

Two graphs  $G = (V, E)$  and  $G' = (V', E')$  are said to be *isomorphic* if there exists a bijection  $f : V \rightarrow V'$  such that

$$\{x, y\} \in E \quad \Leftrightarrow \quad \{f(x), f(y)\} \in E'$$

for all distinct vertices  $x, y \in V$ . The mapping  $f$  is then called an *isomorphism* between  $G$  and  $G'$ . When this is the case, we denote the relation by  $G \cong G'$ .

**Definition 2.3 (Tree).** A tree is a connected graph that does not have a cycle.

A graph  $G = (V, E)$  satisfies the following equivalent conditions:

- $G$  is a tree.
- Path uniqueness: For every two vertices  $x, y \in V$ , there exists exactly one path from  $x$  to  $y$ .
- Minimal connected graph: The graph  $G$  is connected, and deleting any of its edges gives rise to a disconnected graph.
- Maximal graph without cycles: The graph  $G$  contains no cycle, and any graph arising from  $G$  by adding an edge (i.e., a graph of the form  $G + e$ , where  $e$  is a new edge that is not in  $G$ ) already contains a cycle.
- Euler's formula:  $G$  is connected and  $|V| = |E| + 1$ .

Let  $G$  be a graph, and let  $v$  be a vertex of  $G$ . The number of edges of  $G$  that contain the vertex  $v$  is written as  $\deg_G(v)$ . The number  $\deg_G(v)$  is called the *degree* of  $v$  in the graph  $G$ . Let us label the vertices of  $G$  as  $v_1, v_2, \dots, v_n$  (in some arbitrarily chosen order). The sequence

$$(\deg_G(v_1), \deg_G(v_2), \dots, \deg_G(v_n))$$

is referred to as the *degree sequence* of the graph  $G$ , or as the *score* of  $G$ . By assigning different numberings to the vertices of the same graph, one generally obtains sequences that differ in the order of their terms. Therefore, we do not consider two scores distinct if one can be derived from the other by simply rearranging the order of the terms (Matoušek & Nešetřil, 2008).

In general graph theory, graphs are defined abstractly, but in this research we will be studying Steiner Trees in the Euclidean plane. Working in the Euclidean plane allows the usage of the Euclidean distance. Given that  $(x_1, y_1)$  and  $(x_2, y_2)$  are any two points in the plane, then the distance between them is defined as:

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

We will only have arcs which consist of a finite number of straight line segments. The length of such an arc is simply the sum of the lengths of the segments. An **arc** is a subset  $\alpha$  of the plane of the form  $\alpha = \gamma([0, 1]) = \{\gamma(x) : x \in [0, 1]\}$ , where  $\gamma : [0, 1] \rightarrow \mathbf{R}^2$  is an injective continuous map of the closed interval  $[0, 1]$  into the plane. The points  $\gamma(0)$  and  $\gamma(1)$  are called the *endpoints* of the **arc**  $\alpha$ .

**Definition 2.4 (Drawing).** By a drawing of a graph  $G = (V, E)$  we mean an assignment as follows: to every vertex  $v$  of the graph  $G$ , assign a point  $b(v)$  of the plane, and to every edge  $e = \{v, v'\} \in E$ , assign an arc  $\alpha(e)$  in the plane with endpoints  $b(v)$  and  $b(v')$ . We assume that the mapping  $b$  is injective (different vertices are assigned distinct points in the plane), and no point of the form  $b(v)$  lies on any of the arcs  $\alpha(e)$  unless it is an endpoint of that arc.

**Definition 2.5 (Topological graph).** A graph together with some drawing is referred to as a topological graph.

Terms of graph theory naturally also apply to topological graphs. Based on this concept, we call a topological graph a *tree* if its graph is a tree. A tree is a *spanning tree* of some finite set  $V$  in  $\mathbf{R}^2$  if it has  $V$  as its vertex set. Suppose that we have the set  $V = \{A, B, C\}$ , then a possible spanning tree is shown in Figure 1. Let the vertices  $A, B$  and  $C$  in Figure 1 be considered cities with coordinates

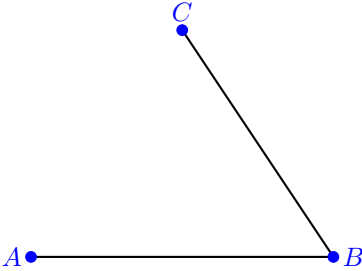


Figure 1

$A(0, 1), B(2, 1)$  and  $C(1, 3)$ . Then an interesting problem is how can we connect these cities with roads while keeping the length of these roads minimal? This is a common problem in network design, where the usual approach is to determine the minimum spanning tree(MST). The problem of finding a MST is defined as follows:

**Definition 2.6.** Given a graph  $G = (V, E)$  and edge weights  $w$ , find a subset of the edges  $E' \subseteq E$  such that: (i) the subgraph  $(V, E')$  is a spanning tree, and (ii) the sum of edge weights

$$w(E') = \sum_{e \in E'} w(e)$$

is minimized.

To solve this we simply use the Euclidean distances between the vertices for the weights. In our case, the distance for every pair of vertices is:

$$d(A, B) = \sqrt{(2 - 0)^2 + (1 - 1)^2} = 2$$

$$d(B, C) = \sqrt{(2 - 0)^2 + (1 - 1)^2} = \sqrt{5}$$

$$d(A, C) = \sqrt{(1 - 0)^2 + (3 - 1)^2} = \sqrt{5}$$

We take the shortest distances and that results in either A-B-C or B-A-C shown in Figure 2. The Total length  $\|T\|$  of the MST is thus  $2 + \sqrt{5}$ .

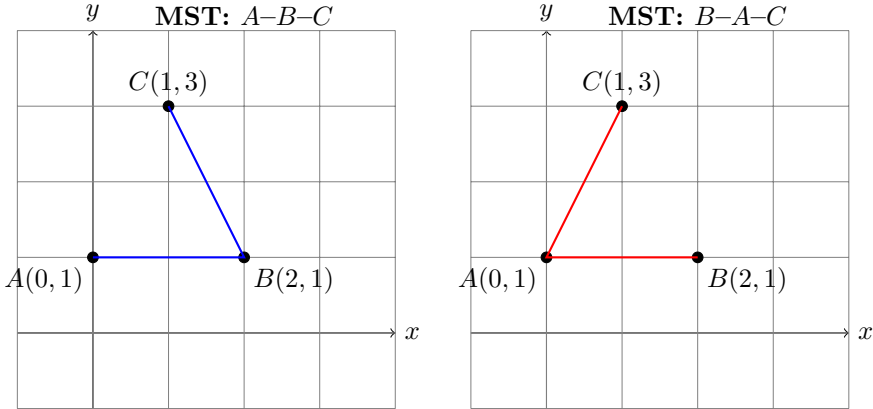


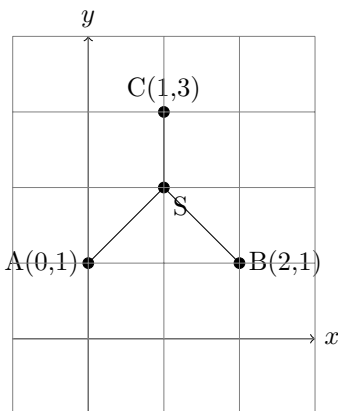
Figure 2

However, the MST is limited to using only the given vertices as connection points. This leads to the natural question: Can we reduce the total length more by adding extra points? This is how we

obtain the Steiner tree, which extends the idea of spanning trees by allowing the inclusion of extra vertices to obtain a shorter tree.

**Definition 2.7.** A *Steiner tree* of  $V$  is a spanning tree of some finite vertex set  $S$  in a metric space<sup>1</sup> such that  $S \supseteq V$  and each vertex in  $S \setminus V$  has degree at least 3. We call the vertices in  $V$  the terminals of the Steiner tree, and the vertices in  $S \setminus V$ , the Steiner points of the Steiner tree.

Using the same set from Figure 1, we can introduce a Steiner Point  $S$  to create a Steiner tree (Figure 3).



**Figure 3**

The length of each edge can be calculated as follows;

$$d(A, S) = \sqrt{(1-0)^2 + (2-1)^2} = \sqrt{2}$$

$$d(B, S) = \sqrt{(1-2)^2 + (2-1)^2} = \sqrt{2}$$

$$d(C, S) = \sqrt{(1-1)^2 + (2-3)^2} = 1$$

For this tree  $\|T\| = 1 + 2\sqrt{2}$

This example shows that introducing a Steiner point may reduce the total tree length compared to the minimum spanning tree(MST), confirming that the Steiner tree provides a shorter connection between the given vertices. It can also be shown that, for a given set of vertices in the Euclidean plane, there exists a shortest Steiner tree, called a *Steiner Minimal Tree (SMT)*.

Using Euler's formula a Steiner tree connecting  $n$  terminal points and  $m$  additional Steiner points forms a tree with  $n + m - 1$  edges. In such a tree, we can make two key observations:

1. Terminal points must have degree at least one, and we define Steiner points to have degree at least three.
2. For each graph  $G = (V, E)$ , we have

$$\sum_{v \in V} \deg_G(v) = 2|E|.$$

Applying these to our Steiner tree, which has  $n + m - 1$  edges, results in a total degree sum of  $2(n + m - 1)$ .

Combining both observations, we get a lower bound on the total degree:

$$2(n + m - 1) \geq n + 3m$$

---

<sup>1</sup>We will only use the Euclidean plane  $\mathbf{R}^2$ .

Solving this inequality leads to:

$$\begin{aligned}
 2(n + m - 1) &\geq n + 3m \\
 2n + 2m - 2 &\geq n + 3m \\
 2n - n + 2m - 3m &\geq 2 \\
 n - m &\geq 2 \\
 m &\leq n - 2
 \end{aligned}$$

which shows that the number of Steiner points is bounded above by  $n - 2$ . The possible number of graphs for a finite number of vertices is finite. So it follows there are only finitely many different graph structures for Steiner trees with  $n$  given terminals. Let us consider a particular Steiner tree with the terminals fixed in  $R^2$ , but the positions of the Steiner points are variable. The tree's edges are composed of straight line segments. Let us assume the Steiner points all lie within a closed ball containing  $V$ . The total length of such a tree is a continuous function of the positions of the Steiner points, which are restricted to a compact subset of  $R^{2m}$ . By the extreme value theorem, a continuous function defined on a compact set attains a minimum. In conclusion, among all possible graph structures and positions of Steiner points, there exists at least one structure that gives a tree of minimal total length. This confirms the existence of an SMT for any finite set of vertices in the Euclidean plane (De Wet, 2008).

## 2.3 Hexagonal Trees

A Steiner tree is a *full Steiner tree* if all terminals have degree 1. All terminals A, B and C in Figure 3 have degree 1, meaning this Steiner tree is a full Steiner tree. In contrast, a *Steiner tree that is not full*, is the union of edge disjoint full Steiner trees. We call the smaller trees that the union is made of full subtrees and the original tree is called the main tree. An example of this is shown in Figure 4.

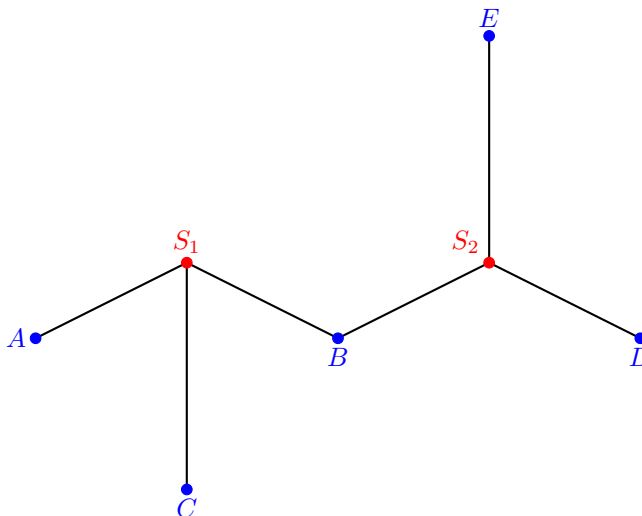


Figure 4

Each full SMT has the interesting attributes that the Steiner points all have degree 3 and that the three edges meet at  $120^\circ$  relative to each other. In what follows, it is shown why the angle between the three edges at a Steiner point must be  $120^\circ$ .

### 2.3.1 Angle Condition at Steiner Points

Consider two edges  $\alpha$  and  $\beta$  meeting at less than  $120^\circ$ , and draw  $AS$ ,  $BS$  and  $CS$  as shown in Figure 5, such that angles at  $S$  are  $120^\circ$  and such that  $\|AC\| = \|BC\|$ . It is sufficient to show that

$\|AS\| + \|BS\| + \|CS\|$  is shorter than  $\|AC\| + \|BC\|$ . Using the 30-60-90 triangle rule for  $ABC$  gives us:

$$\|AS\| = \|BS\| = 2\|SD\| \quad \text{and} \quad \|AD\| = \sqrt{3}\|SD\|$$

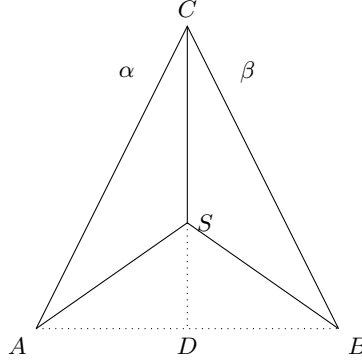


Figure 5

We note that

$$\begin{aligned} \|AS\| + \|BS\| + \|CS\| &= \|CS\| + 4\|SD\| \\ &= \sqrt{\|CS\|^2 + 8\|CS\|\|SD\| + 16\|SD\|^2}, \end{aligned}$$

but from

$$\begin{aligned} \|AC\| = \|BC\| &= \sqrt{(\|CS\| + \|SD\|)^2 + \|AD\|^2} \\ &= \sqrt{(\|CS\| + \|SD\|)^2 + 3\|SD\|^2}, \end{aligned}$$

we have that

$$\begin{aligned} \|AC\| + \|BC\| &= 2\sqrt{\|CS\|^2 + 2\|CS\|\|SD\| + 4\|SD\|^2} \\ &= \sqrt{4\|CS\|^2 + 8\|CS\|\|SD\| + 16\|SD\|^2}. \end{aligned}$$

The tree in which three edges meet at a point forming  $120^\circ$  angles results in a shorter total length than the tree where the angles are less than  $120^\circ$ . This result explains why every Steiner point in an SMT has degree three, and the edges form  $120^\circ$  angles in the Steiner point. This structure minimizes  $\|T\|$  (De Wet, 2008). Motivated by this, we introduce the following definition:

**Definition 2.8** (Hexagonal Tree). A *hexagonal tree (HT)* is a full Steiner tree with the following properties:

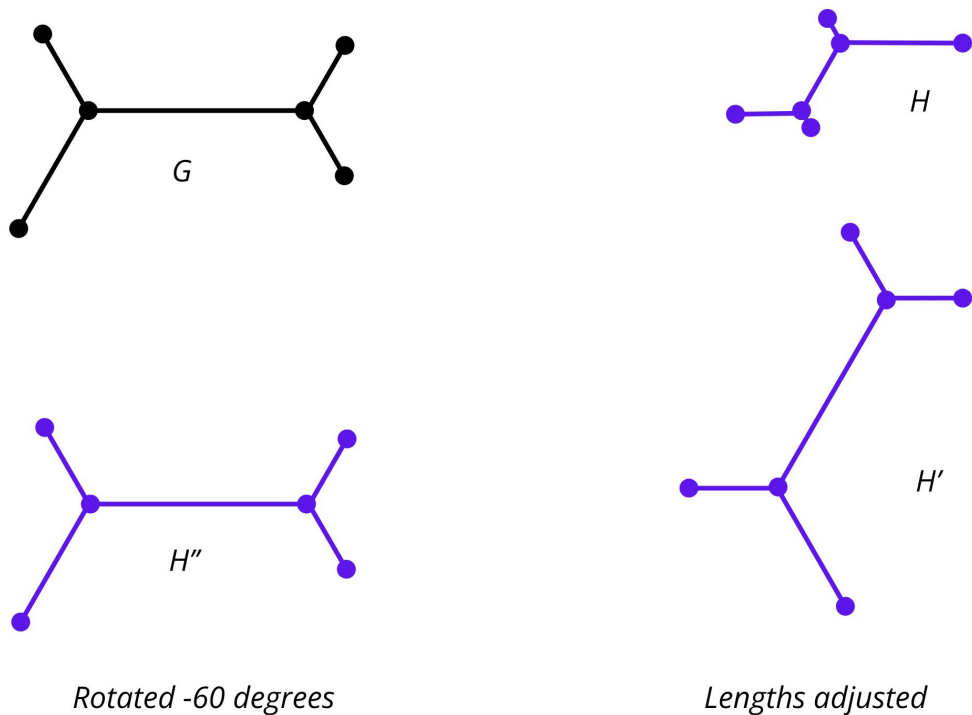
1. every Steiner point has degree 3;
2. all edges are straight line segments;
3. the three edges that are connected to the Steiner points meet at angles of  $120^\circ$  relative to each other.

Then we can make the following definition:

**Definition 2.9** (Congruent Hexagonal Trees). Two hexagonal trees  $G$  and  $H$  are said to be *congruent* if one can be transformed into the other by translation, rotation, and/or mirroring. If we were to put  $G$  on top of  $H$  after the transformation it would fit right on top of it.

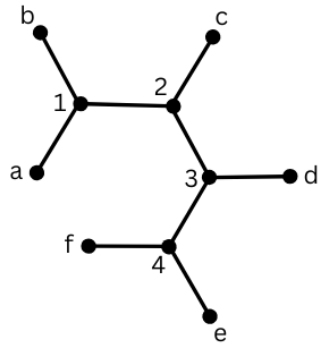
**Definition 2.10** (Same Topology). Two hexagonal trees  $G$  and  $H$  are said to have the *same topology* if there exists a way to change the edge lengths of  $G$  such that it becomes congruent to  $H$ .

For example in Figure 6. we can change the length of the edges of  $H$ , rotate it  $-60$  degrees, then we get a tree that is congruent to  $G$ . That is,  $G$  and  $H$  have the same topology.

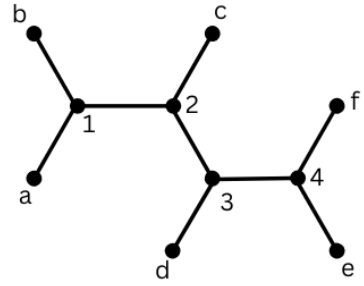


**Figure 6:** Two hexagonal trees with the same topology

It is important to note that if two hexagonal trees  $G$  and  $H$  have the same topology, then their graph structures are isomorphic. However, the converse is not necessarily true: two hexagonal trees with isomorphic graph structures do not always have the same topology. We illustrate this with an example. Let  $G$  and  $H$  be two hexagonal trees shown in figure 7. Both these trees have the vertex set  $V(G) = V(H) = \{1, 2, 3, 4, a, b, c, d, e, f\}$  and the edge set  $E(G) = E(H) = \{(1, a), (1, b), (1, 2), (2, c), (2, 3), (3, d), (3, 4), (4, e), (4, f)\}$ .



*G*



*H*

**Figure 7**

In conclusion they are isomorphic but they don't have the same topology.

### 3 Counting trees

In this section, we explore several ways of counting graphs. To introduce this, we begin with a fundamental proposition in combinatorics:

**Proposition 3.1.** Any  $n$ -element set  $X$  has exactly  $2^n$  subsets.

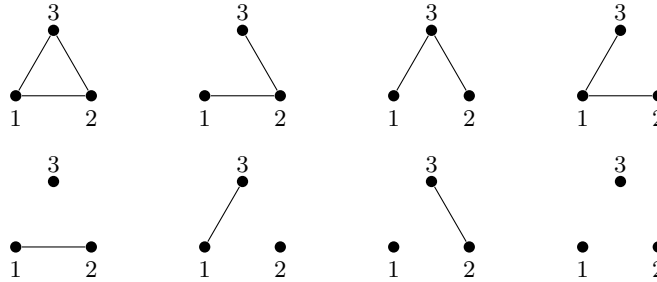
Let  $V = \{1, 2, \dots, n\}$  be a set of vertices. Each edge connects a pair of vertices and the set of all possible edges is the set of 2-element subsets of  $V$ , denoted  $\binom{V}{2}$ . There are  $\binom{n}{2}$  such pairs (Matoušek & Nešetřil, 2008). To define a graph with vertex set  $V$ , we choose a subset of those possible edges:

$$E \subseteq \binom{V}{2}.$$

By Proposition 3.1., the amount of distinct graphs on  $V$  is:

$$2^{\binom{n}{2}}$$

However, there are less than  $2^{\binom{n}{2}}$  pairwise nonisomorphic graphs with  $n$  vertices. For example, for  $V = \{1, 2, 3\}$ , there are  $8 = 2^{\binom{3}{2}}$  distinct graphs:



Out of these 8 possibilities, only 4 are nonisomorphic:



Figure 8

So we get the question: Given  $n$  vertices, how many pairwise nonisomorphic graphs are there for a general  $n$ ? Finding this number precisely is not easy, but we can at least get estimate by a simple trick. It is clear that the number of nonisomorphic graphs on  $n$  vertices cannot exceed the total number of different graphs on the set  $V$ , given by  $2^{\binom{n}{2}}$ . On the other hand, suppose we have a graph  $G$  with vertex set  $V$ . We may ask: how many different graphs  $G'$  on  $V$  are isomorphic to  $G$ ? As an example, consider the case where  $G$  is the graph on the vertex set  $\{1, 2, 3\}$  (Figure 9), there are 3 graphs that are isomorphic.

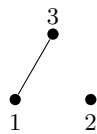


Figure 9

If  $G'$  is isomorphic to  $G$ , then there exists a bijection  $f : V \rightarrow V$  that is an isomorphism between  $G$  and  $G'$ . Since the possible bijections on a set of  $n$  vertices  $f : V \rightarrow V$  is  $n!$ ,  $G$  can be isomorphic to

at most  $n!$  distinct graphs on the set  $V$  (note that this may overcount some graphs). For our specific example with  $n = 3$  we had  $3! = 6$  bijections, but only 3 distinct graphs are actually isomorphic to  $G$ .

In other words, each equivalence class under the relation  $\cong$  on the set of all graphs with vertex set  $V$  consists of at most  $n!$  graphs. Therefore, the number of equivalence classes is at least

$$\frac{2^{\binom{n}{2}}}{n!}.$$

As a result, there exists a set of at least this many graphs on  $n$  vertices that are pairwise nonisomorphic (Matoušek Nešetřil, 2008).

### 3.1 Cayley's Theorem

In the specific case of trees, Cayley proved that there are  $n^{n-2}$  distinct trees on  $n$  vertices (Wilson, 2010). For example, consider Figure 10, which shows two different tree structures we can get for a tree on 4 vertices. We now count how many different trees there are for  $n = 4$  by counting how many different ways we can label the vertices with labels  $\{1, 2, 3, 4\}$ .

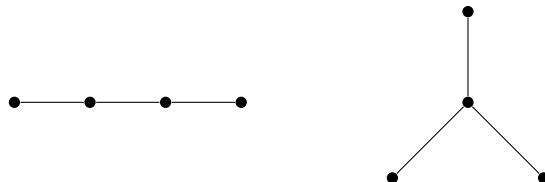


Figure 10

The first results in  $\frac{4!}{2} = 12$  different trees. And in the last tree, there is a central vertex that can be labeled in four different ways, contributing an additional 4 trees. Therefore, the total number of trees on four vertices is  $12 + 4 = 16$ . This small example is generalized by Cayley's theorem:

[Cayley's theorem] The number of trees we can have on  $n$  vertices equals:

$$n^{n-2}.$$

In other words, there are exactly  $n^{n-2}$  distinct spanning trees on  $n$  labeled vertices ((Matoušek Nešetřil, 2008)). To provide a proof for this theorem, we first look at a proposition that will be used in the proof.

**Proposition 3.2.** Let  $d_1, d_2, \dots, d_n$  be positive integers such that their sum is  $2n - 2$ . Then the number of spanning trees of the graph  $K_n$  in which the vertex  $i$  has degree exactly  $d_i$  for each  $i = 1, 2, \dots, n$  is given by

$$\frac{(n-2)!}{(d_1-1)!(d_2-1)! \cdots (d_n-1)!}.$$

**Proof.** By induction on  $n$ .

**Case  $n = 1$ :** The graph  $K_1$  consists of a single vertex and no edges. There is only one trivial spanning tree: the vertex itself.

**Case  $n = 2$ :** The graph  $K_2$  has two vertices connected by one edge. The unique spanning tree is the edge itself, with degree sequence  $(1, 1)$ . We check the formula:

$$\frac{(2-2)!}{(1-1)! \cdot (1-1)!} = \frac{0!}{0! \cdot 0!} = \frac{1}{1 \cdot 1} = 1$$

which matches the expected number of spanning trees. For  $n = 1, 2$ , the proposition holds trivially, so let  $n > 2$ .

**Case  $n - 1$ :** By the inductive hypothesis, for  $(n - 1)$  degree sequences we have

$$\frac{((n - 2) - 1)!}{(d_1 - 1)! \cdots (d_2 - 1)! \cdots (d_{n-1} - 1)!} = \frac{(n - 3)!}{(d_1 - 1)! \cdots (d_2 - 1)! \cdots (d_{n-1} - 1)!}$$

**Case  $n$ :**

In a tree with  $n$  vertices there are always  $n - 1$  edges. Applying the Handshaking Lemma, we get:

$$\sum_{v \in V} \deg_G(v) = 2|E| = 2(n - 1).$$

Therefore,

$$\sum d_i = 2n - 2 < 2n.$$

Because the sum of the  $d_i$  is less than  $2n$ , there must exist at least one  $i$  such that  $d_i = 1$ . For notational convenience, we assume that  $d_n = 1$  in the proof. We could take any other  $d_i = 1$  and it would give us the same results. Let  $T$  be the set of all spanning trees of  $K_n$  with a given degree sequence  $(d_1, d_2, \dots, d_n)$ , where each vertex  $i$  has degree  $d_i$ . We split  $T$  into  $n - 1$  subsets  $T_1, T_2, \dots, T_{n-1}$ . Then we define  $T_j$  to be the set that consists of all trees in which vertex  $n$  is connected to vertex  $j$ .

Now, take any tree from  $T_j$ . Removing vertex  $n$  together with its single edge yields a spanning tree of  $K_{n-1}$ . In this new tree, the degrees remain the same for all vertices except for vertex  $j$ , whose degree decreases by one: the new degree sequence becomes

$$(d_1, \dots, d_{j-1}, d_j - 1, d_{j+1}, \dots, d_{n-1}).$$

This construction defines a bijection between the trees in  $T_j$  and the set  $T'_j$ , consisting of all spanning trees of  $K_{n-1}$  with this adjusted degree sequence. Each distinct tree in  $T_j$  maps to a unique tree in  $T'_j$ , and each tree in  $T'_j$  can be transformed back into a tree in  $T_j$  by reattaching vertex  $n$  to vertex  $j$ . In other words the number of trees in  $T'_j$  equals the number of trees in  $T_j$ , because there's a bijection between them.

Continuing with the inductive hypothesis( $K_{n-1}$ ), we have

$$|T_j| = |T'_j| = \frac{(n - 3)!}{(d_1 - 1)! \cdots (d_{j-1} - 1)! ((d_j - 1) - 1)! (d_{j+1} - 1)! \cdots (d_{n-1} - 1)!}$$

$$|T_j| = |T'_j| = \frac{(n - 3)!}{(d_1 - 1)! \cdots (d_{j-1} - 1)! (d_j - 2)! (d_{j+1} - 1)! \cdots (d_{n-1} - 1)!}$$

To simplify the expression, we use the standard factorial identity:

$$(d_j - 1)! = (d_j - 1) \cdot (d_j - 2)! \implies (d_j - 2)! = \frac{(d_j - 1)!}{d_j - 1} \implies \frac{1}{(d_j - 2)!} = \frac{d_j - 1}{(d_j - 1)!}$$

This substitution allows us to write all denominator terms in the form  $(d_i - 1)!$ , including the term for  $d_j$ , whose degree was reduced by one.

Applying this identity to the denominator gives:

$$\frac{(n - 3)! (d_j - 1)}{(d_1 - 1)! \cdots (d_{j-1} - 1)! (d_j - 1)! (d_{j+1} - 1)! \cdots (d_{n-1} - 1)!} = \frac{(n - 3)! (d_j - 1)}{(d_1 - 1)! (d_2 - 1)! \cdots (d_{n-1} - 1)!}.$$

The formula remains valid when  $d_j = 1$ ; in this case, it evaluates to 0, which correctly reflects that no spanning tree exists in which vertex  $j$  has degree  $d_j - 1 = 0$ .

So now summing over all  $j = 1$  to  $j = n - 1$  we have:

$$\begin{aligned} |T| &= \sum_{j=1}^{n-1} |T_j| = \sum_{j=1}^{n-1} \frac{(n - 3)! (d_j - 1)}{(d_1 - 1)! \cdots (d_{n-1} - 1)!} \\ &= \frac{(n - 3)!}{(d_1 - 1)! \cdots (d_{n-1} - 1)!} \sum_{j=1}^{n-1} (d_j - 1) \end{aligned}$$

$$\sum_{i=1}^n d_i = 2(n-1) \Rightarrow \sum_{j=1}^{n-1} d_j = 2(n-1) - d_n = 2n-3$$

$$\sum_{j=1}^{n-1} (d_j - 1) = (2n-3) - (n-1) = n-2$$

$$|T| = \frac{(n-3)!}{(d_1-1)! \cdots (d_{n-1}-1)!} \cdot (n-2) = \frac{(n-2)!}{(d_1-1)! \cdots (d_{n-1}-1)!}$$

Since  $d_n = 1$ , we can multiply the denominator by the factor  $(d_n - 1)! = 0! = 1$  with no harm, and this finishes the inductive step.

$$|T| = \frac{(n-2)!}{(d_1-1)!(d_2-1)! \cdots (d_n-1)!}$$

□

To explain this visually, consider the complete graph  $K_4$ , and a spanning tree with score:

$$(d_1, d_2, d_3, d_4) = (2, 2, 1, 1)$$

This is valid because:

$$\sum d_i = 2(n-1) = 2(4-1) = 6$$

First we calculate the number of spanning trees by using the proposition:

$$\frac{(4-2)!}{(2-1)!(1-1)!(1-1)!(2-1)!} = \frac{2!}{(1!)(0!)(0!)(1!)} = 2$$

Then we follow by trying the method that was used for the proof.

$T$  is considered to be the set with all spanning trees with score  $(2,2,1,1)$ . For the sake of the proof we treat vertex 4 as special ( $d_n = 4$ ). For each  $j = 1, 2, 3$ ,  $T_j$  is the subset of  $T$  where vertex 4 is connected to vertex  $j$ . In this case, we can split  $T$  into  $n - 1$  groups with:

- $T_1$  consists of all trees of  $T$  where vertex 4 is connected to vertex 1.
- $T_2$  consists of all trees of  $T$  where vertex 4 is connected to vertex 2.
- $T_3$  consists of all trees of  $T$  where vertex 4 is connected to vertex 3.

Next we consider a tree from the group  $T_1$  (figure 11). All trees in  $T_1$  have an edge  $\{4, 1\}$ .

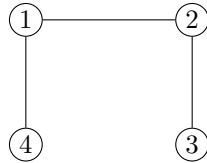


Figure 11

We now remove vertex 4 (a leaf), and reduce the degree of vertex 1 by 1. The new degree sequence on 3 vertices becomes:  $(d'_1, d_2, d_3) = (1, 2, 1)$ .

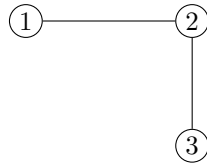


Figure 12

We do the same for  $T_2$  and  $T_3$  and get the new scores:

- For  $T_2$ : We remove the edge  $\{4, 2\}$ . Vertex 2 loses one connection, so the new score  $(d_1, d'_2, d_3) = (2, 1, 1)$ .
- For  $T_3$ : We remove the edge  $\{4, 3\}$ . Vertex 3 loses one connection, so the new score becomes  $(d_1, d_2, d'_3) = (2, 2, 0)$ , which is not a valid tree degree sequence because vertex 3 becomes disconnected.

For each reduced score, we count the number of trees using the known formula: Number of trees with degrees  $(d_1, d_2, d_3)$  is

$$|T| = \frac{(n-2)!}{(d_1-1)!(d_2-1)!(d_3-1)!}$$

For a tree on 3 vertices,  $n = 3$ , so  $(n-2)! = 1! = 1$ .

For  $T_1$ : The degrees are  $(1, 2, 1)$ , so:

$$|T_1| = \frac{1!}{0! \cdot 1! \cdot 0!} = 1.$$

For  $T_2$ : The degrees are  $(2, 1, 1)$ , so:

$$|T_2| = \frac{1!}{1! \cdot 0! \cdot 0!} = 1.$$

For  $T_3$ : The degrees are invalid, since vertex 3 has degree 0, so:

$$|T_3| = 0.$$

Adding all groups together results in:

$$|T| = |T_1| + |T_2| + |T_3| = 1 + 1 + 0 = 2.$$

This is equal to the results from the proposition. Now we can prove Cayley's theorem by using the multinomial theorem:

[multinomial theorem] For arbitrary real numbers  $x_1, x_2, \dots, x_n$  and any natural number  $m \geq 1$ , the following equality holds:

$$(x_1 + x_2 + \dots + x_n)^m = \sum_{\substack{k_1 + \dots + k_n = m \\ k_i \geq 0}} \frac{m!}{k_1! \dots k_n!} x_1^{k_1} \dots x_n^{k_n}.$$

We already know that  $\sum_{i=1}^n d_i = 2(n-1) = 2n-2$ . We will now count, for each such degree sequence, how many labeled trees of  $K_n$  have exactly these degrees. It is known that the number of labeled trees with vertex degrees  $d_1, \dots, d_n$  is given by:

$$\frac{(n-2)!}{(d_1-1)!(d_2-1)! \dots (d_n-1)!},$$

provided each  $d_i \geq 1$ . Therefore, the total number of spanning trees in  $K_n$  is obtained by summing up all possible scores of spanning trees:

$$\sum_{\substack{d_1 + \dots + d_n = 2n-2 \\ d_i \geq 1}} \frac{(n-2)!}{(d_1-1)!(d_2-1)! \dots (d_n-1)!}. \quad (1)$$

We then change the variables to match those in the multinomial theorem.

$$\text{Let } k_i = d_i - 1 \text{ for each } i = 1, \dots, n.$$

Then  $k_i \geq 0$ , and the condition  $\sum d_i = 2n - 2$  becomes:

$$\sum_{i=1}^n (k_i + 1) = 2n - 2 \Rightarrow \sum_{i=1}^n k_i = n - 2.$$

So the sum becomes:

$$\sum_{\substack{k_1 + \dots + k_n = n-2 \\ k_i \geq 0}} \frac{(n-2)!}{k_1! k_2! \dots k_n!}. \quad (2)$$

We proceed with the multinomial identity:

Let  $x_1 = x_2 = \dots = x_n = 1$ , and  $m = n - 2$ , then we get:

$$\sum_{\substack{k_1 + \dots + k_n = n-2 \\ k_i \geq 0}} \frac{(n-2)!}{k_1! \dots k_n!} = (1 + 1 + \dots + 1)^{n-2} \text{(Matoušek \& Nešetřil, 2008)}. \quad (3)$$

Since there are  $n$  terms in the sum inside the parentheses, we get:

$$(1 + 1 + \dots + 1)^{n-2} = n^{n-2}. \quad (4)$$

□

This theorem provides a fundamental result in tree enumeration. However, it is too general for our purposes. First, it counts different trees, whereas in our earlier chapters we clarified that our goal is to enumerate distinct topologies. That is, the trees do not have the same topology. Furthermore, Cayley's formula includes all types of trees, with no restriction on vertex degrees. In contrast, this study focuses specifically on *HT* topologies, where Steiner points must have degree 3, and terminal nodes have degree 1.

### 3.2 Full unrooted binary trees

We also consider the class of full unrooted binary trees.

**Definition 3.1.** A full unrooted binary tree is a tree in which each vertex has either one or three neighbors.

To see how many of these trees exist, we use the same approach that was used for the Cayley theorem.

Suppose that:

- $l$ : number of vertices with degree 1 (leaves)
- $m$ : number of vertices with degree 3 (internal nodes)

We have two constraints:

$$l + m = n \quad (5)$$

$$1 \cdot l + 3 \cdot m = 2(n - 1) \quad (6)$$

Subtracting (5) from (6):

$$(3m - m) = 2(n - 1) - n \Rightarrow 2m = n - 2 \Rightarrow m = \frac{n - 2}{2}$$

$$l = n - m = n - \frac{n - 2}{2} = \frac{n + 2}{2}$$

This implies that  $n$  must be even for  $l$  and  $m$  to be integers. Therefore, for odd  $n$ , there are no such trees: The amount of trees if  $n$  is odd is 0.

Then we count all valid scores. For even  $n$ , the number of valid degree sequences is equal to the number of ways to choose which  $m = \frac{n-2}{2}$  vertices will have degree 3:

$$\text{scores} = \binom{n}{\frac{n-2}{2}} = \binom{n}{\frac{n}{2} - 1}$$

For each valid score  $(d_1, \dots, d_n)$  with  $d_i \in \{1, 3\}$ , the number of labeled trees is given by:

$$\frac{(n-2)!}{\prod_{i=1}^n (d_i - 1)!}$$

For:

- $d_i = 1 \Rightarrow (d_i - 1)! = 0! = 1$
- $d_i = 3 \Rightarrow (d_i - 1)! = 2! = 2$

There are  $\frac{n+2}{2}$  leaves and  $\frac{n-2}{2}$  degree-3 vertices, so:

$$\prod_{i=1}^n (d_i - 1)! = 1^{\frac{n+2}{2}} \cdot 2^{\frac{n-2}{2}} = 2^{\frac{n}{2}-1}$$

Thus, the number of trees for one such degree sequence is as follows:

$$\frac{(n-2)!}{2^{\frac{n}{2}-1}}$$

By multiplying the number of degree sequences by the number of trees per sequence, we get:

$$\boxed{(n-2)! \cdot \binom{n}{\frac{n}{2} - 1} \cdot 2^{-(\frac{n}{2}-1)}}$$

This is the number of trees on  $n$  vertices in which all vertices have degree 1 or 3, for even  $n$  (Grinberg 2019). We refer to any such tree as  $T_{1,3}$ . The number of these trees is smaller than the number given by Cayley's theorem, which we can verify by comparing the first five values.

$n$	$n^{n-2}$	$T_{1,3}(n)$
1	1	0
2	1	1
3	3	0
4	16	4
5	125	0
6	1296	90

Table 1

### 3.3 Catalan numbers and future research

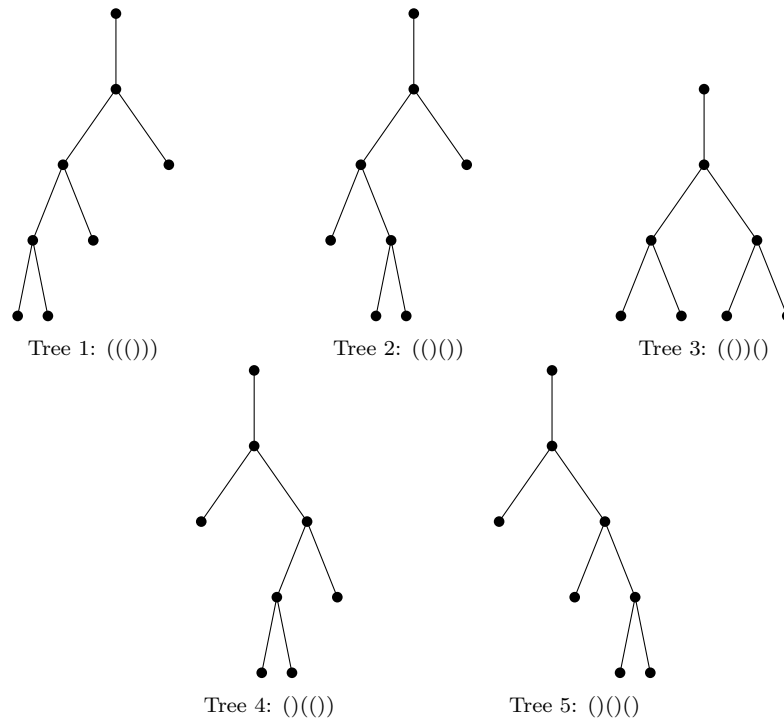
The Catalan numbers ( $C_n$ ) are named after the Belgian mathematician Eugène Charles Catalan (1814–1894) (Stanley 2015). In 1838, he established the standard formula to find the  $n$ -th Catalan number. The formula is:

$$C_n = \frac{(2n)!}{(n+1)!n!} = \binom{2n}{n} - \binom{2n}{n-1}, \quad \text{for } n \geq 0$$

They are a sequence of numbers that arise in many counting problems (Patrick Corn, Eddie The Head, Alex Li, et al., 2025). One of these problems is the numbers of ways to arrange  $n$  pairs of parentheses which are properly matched. Each one of these expressions can be represented as trees with  $n$  internal nodes and  $n+2$  leaves. For example for  $n=3$  there are 5 ways to arrange the parentheses:

- $((()))$
- $(())$
- $(())()$
- $()()$
- $()()()$

Each one of these corresponds to the following trees (Patrick Corn, Eddie The Head, Alex Li, et al., 2025).



**Figure 13:** Five tree shapes with an extra upward node at the root.

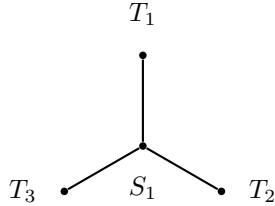
We can use this same method to describe the  $HT$  topologies and treat it as an upperbound, because the same  $HT$  topology can be described by different sequences. For example Tree 1 and Tree 4 are considered the same topology but they are described by different sequences. How these sequences of brackets can be used to further analyze different topologies seems like an interesting topic for future research, but falls outside of the scope of this thesis.

## 4 Methodology

To determine the number of *HT* topologies for a given number of terminals  $n$ , we introduce an algorithm referred to as the append algorithm.

Append Algorithm: We start with a base tree that has three terminals, labeled  $T_1, T_2, T_3$ , all connected to one Steiner point. Then we apply the **append step**: For each terminal  $T_i$  (where  $i = 1, 2, \dots, n$ ), we remove it and replace it with a new Steiner point connected to two new terminals. We repeat this for every  $T_i$  until the number of terminals is reached. After each round, we compare all the new topologies and keep only the **unique** ones. To understand this better we use the append algorithm on the base tree to get the amount of hexagonal tree topologies for  $n = 4$ .

We start with the base tree consisting of three terminals  $T_1, T_2, T_3$  connected to a Steiner point  $S_1$ .



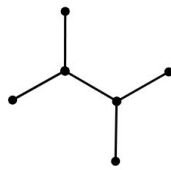
We apply the append step to terminal  $T_1$ :

- Remove  $T_1$
- Replace  $T_1$  with a Steiner point and two terminals.

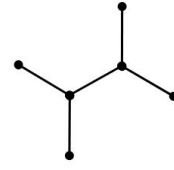
The new tree gets 4 terminals and 2 Steiner points. This is the first topology for  $n = 4$  terminals. This process is repeated for terminal  $T_2$ , then we compare this new tree to the first topology that we got for  $n = 4$ . They are the same topology, so we still only have one topology for  $n = 4$ . This process is repeated again for terminal  $T_3$  and it is clear that they all have the same topology.



Append at  $T_1$



Append at  $T_2$



Append at  $T_3$

**Figure 14**

So there's only one unique topology for  $n = 4$ . We manually use this algorithm up to  $n = 8$  to draw the unique topologies. For  $n \geq 9$  it becomes challenging to compare the results. To verify if the manually found number of topologies is correct, a lower bound is considered. This helps ensure that the count is not too low, since the lower bound represents the minimum number of valid trees that must exist.

To determine the lower bound, we use a python code (see Appendix A.1) that generates graphs with  $n$  terminals connected by  $n - 2$  Steiner points of degree 3, and filters out duplicates using graph isomorphism. However, as we have seen earlier, two hexagonal trees can be isomorphic and have different topologies. As a result, this program may filter out some distinct topologies that should be counted separately. Therefore, the output of this program can be safely treated as a lower bound on the total number of valid hexagonal tree topologies.

## 4.1 Code Overview

The Python implementation is designed around the properties of full Steiner trees:

- Terminals: degree 1
- Steiner points: degree 3
- Total number of Steiner points:  $n - 2$

To get unique trees, we use several functions, each of which are explained below.

`is_isomorphic(g1, g2)`

*Purpose:* Determines whether two graphs are isomorphic and ensures that isomorphic trees are counted only once.

`base_tree()`

*Purpose:* Returns the base case: the unique minimal Steiner tree topology for  $n = 3$ . It serves as a base for generating larger trees.

`expand_tree(tree, s_counter, t_counter)`

*Purpose:* Recursively grows the tree by adding one Steiner node and two terminals to each existing terminal. Each expansion replaces a terminal with a new Steiner point connected to two new terminals, while maintaining the properties of full Steiner trees.

`unique_trees(trees)`

*Purpose:* Filters out isomorphic trees from a list. We save only one representative per isomorphism class  $[G_i]$ , ensuring that each counted tree corresponds to a unique tree.

`relabel_nodes_consistently(G)`

*Purpose:* This function renames the nodes in a graph so that:

- all terminals are named:  $T_1, T_2, \dots, T_n$
- all Steiner nodes are named:  $s_1, s_2, \dots, s_k$

This makes the graph look clean and consistent.

`generate_topologies(n_terminals)`

*Purpose:* Main generator function that builds all unique Steiner trees for a given terminal count.

**Approach:**

1. Start with the base tree for  $n = 3$
2. Recursively apply `expand_tree` to grow trees
3. After each expansion step, filter the trees:
  - Keep only trees with the correct number of terminals
  - Remove isomorphic duplicates

`plot_graph(G, idx)`

*Purpose:* Visualizes a graph. *Styling:*

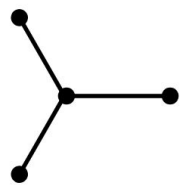
- Steiner nodes: red
- Terminals: sky blue

- Nodes labeled
- Title includes index for reference

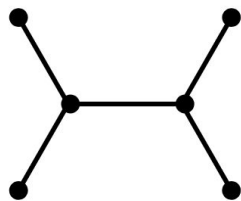
`__main__`

*Purpose:* Serves as the entry point for execution. Sets the desired number of terminals and displays the results.

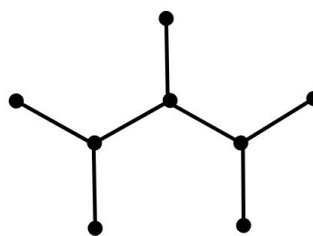
## 5 Results



$n = 3$

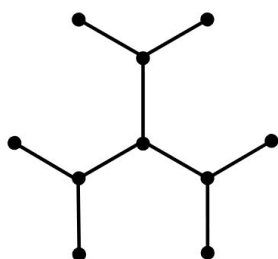


$n = 4$

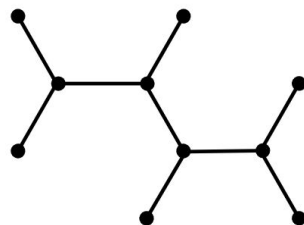


$n = 5$

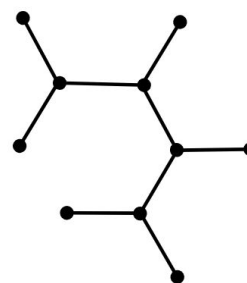
Figure 15



1.

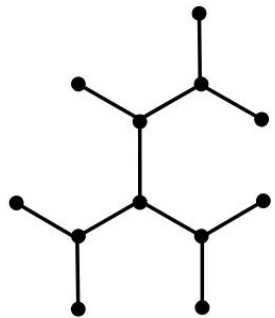


2.

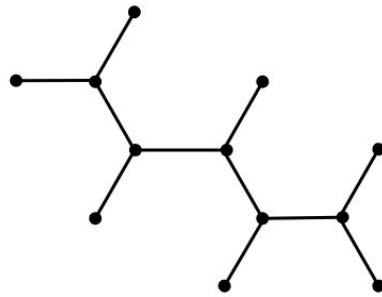


3.

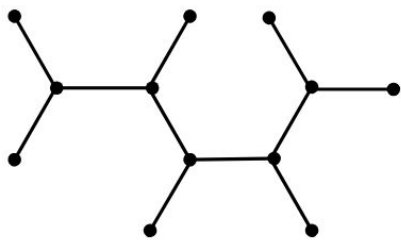
Figure 16:  $n = 6$



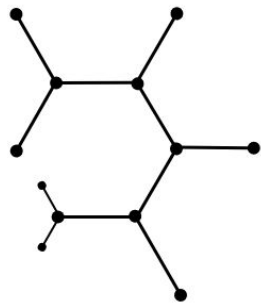
1.



2.

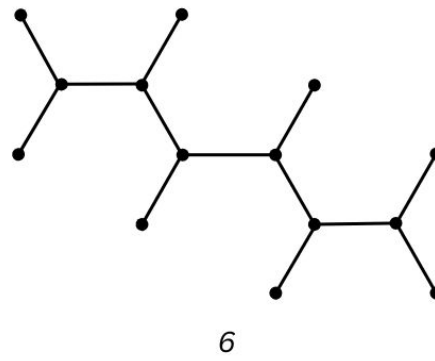
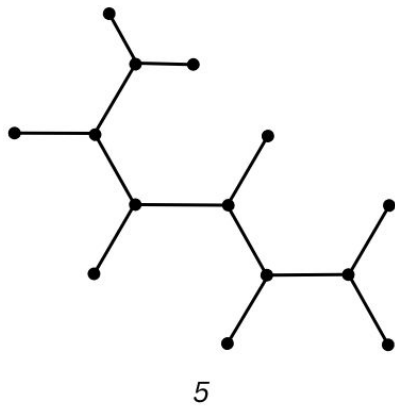
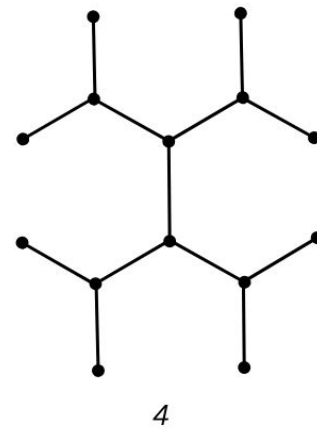
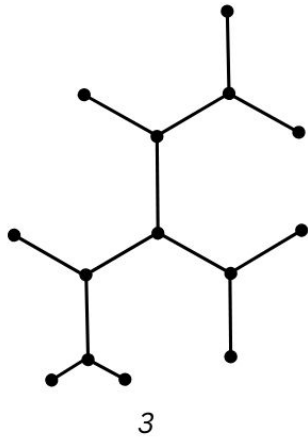
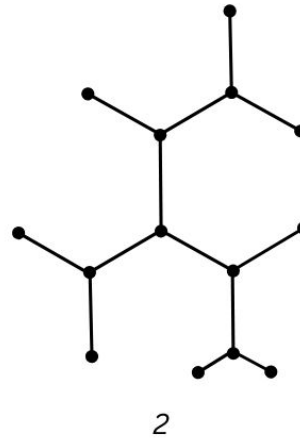
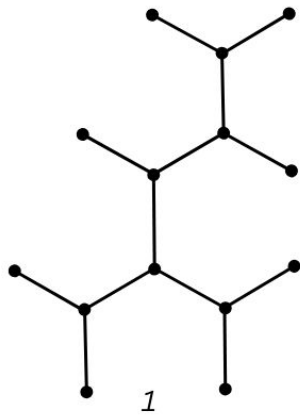


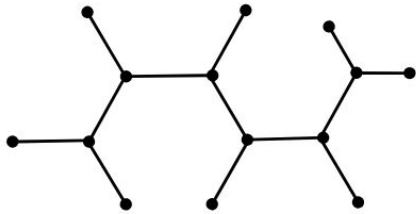
3.



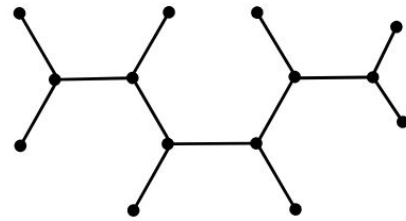
4.

Figure 17:  $n = 7$

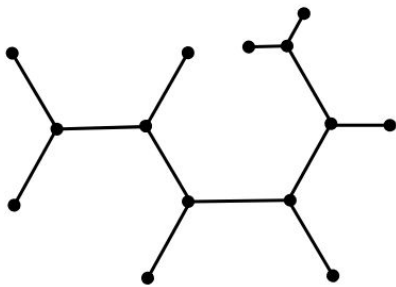




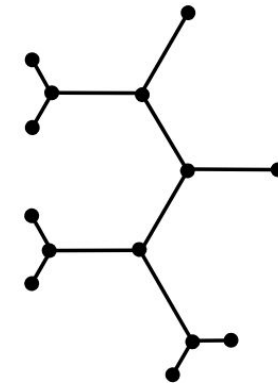
7



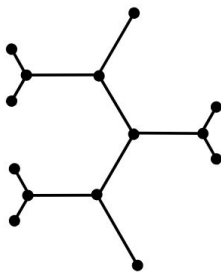
8



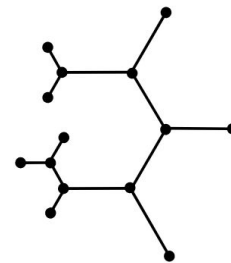
9



10



11



12

Figure 19:  $n = 8$

Following, we present the results of our manually generated  $HT$  topologies and the lower bound derived from the python program. Additionally, we provide visual representations of the python generated trees for small values of  $n$ (see Appendix A.2).

<b>Terminals <math>n</math></b>	<b>Steiner Points <math>n - 2</math></b>	<b>Lower Bound</b>	<b>Unique <i>HT</i> Topologies</b>
1	–	0	0
2	–	0	0
3	1	1	1
4	2	1	1
5	3	1	1
6	4	2	3
7	5	2	4
8	6	4	12
9	7	6	–
10	8	11	–

Table 2: Comparison of unique *HT* topologies and labeled  $T_{1,3}$  tree counts for small values of terminals  $n$ .

## 6 Discussion & Conclusion

This research focused on Steiner Minimal Trees (SMTs) and their structural properties. It revealed that SMTs are considered hexagonal trees when all terminals have degree 1 and the edges at each Steiner point meet at  $120^\circ$  angles. Different graph counting problems and their relevance to the study were examined; including Cayley's theorem, which counts all possible trees on  $n$  points, unrooted unlabeled binary trees, and Catalan numbers. Additionally, the Append Algorithm was developed to determine the number of distinct hexagonal tree topologies for a given number of terminals.

For small values of  $n$  (up to  $n = 8$ ), topologies were computed manually. As the number of terminals increased, manual computation became challenging. A Python program was also implemented to count isomorphic trees for a given  $n$ , providing a reliable lower bound, since trees with the same topology are always isomorphic.

The results indicate that the number of distinct topologies grows rapidly with  $n$ . Due to computational constraints, higher values of  $n$  were not explored. For future research, Catalan numbers, specifically sequences of parentheses, may offer a useful approach for identifying hexagonal tree structures and analyzing their differences.

## References

- [1] de Wet, P. Oloff. (2008). Geometric Steiner minimal trees (Doctoral dissertation, University of South Africa).
- [2] Brazil, M., Graham, R. L., Thomas, D. A., & Zachariasen, M. (2014). On the history of the Euclidean Steiner tree problem. *Archive for History of Exact Sciences*, 68(3), 327–354. <https://doi.org/10.1007/s00407-013-0127-z>
- [3] Fermat point explained. (n.d.). Retrieved June 15, 2025, from [https://everything.explained.today/Fermat\\_point/](https://everything.explained.today/Fermat_point/)
- [4] Gilbert S. (2015, August 25). CS4234: Optimization Algorithms – Lecture 3: Steiner Trees [Lecture notes]. National University of Singapore.
- [5] Courant, R., & Robbins, H. (1941). What is mathematics? (1st ed.). Oxford University Press.
- [6] Matoušek J. and Nešetřil J. (2nd ed.). (2018). Invitation to discrete mathematics. Oxford University Press.
- [7] Gergonne J.-D. (1816–1817). Proposed questions: Problems of geometry I. *Annales de mathématiques pures et appliquées*, 7, 196.
- [8] Grinberg D. (2019, January 26). Find the number of trees on  $2m$  given vertices in which all vertices have degree 1 or 3 [Answer]. Mathematics Stack Exchange. <https://math.stackexchange.com/a/3081257>
- [9] Bartlmae, S., Jünger P. J. & Langetepe E. (n.d.). NP-hardness and a PTAS for the Euclidean Steiner Line Problem. University of Bonn, Germany.
- [10] Stewart J. (2012). Calculus: Early transcendentals (7th ed.). Brooks/Cole, Cengage Learning.
- [11] Wilson R. J. (2010). Introduction to graph theory (4th ed.). Pearson Education.
- [12] Hwang F. K., & Weng J. F. (1986). Hexagonal coordinate systems and Steiner minimal trees.
- [13] Ras C., Swanepoel K., & Thomas D. A. (2017). Approximate Euclidean Steiner trees. *Journal of Optimization Theory and Applications*, 172(3), 845–873. <https://doi.org/10.1007/s10957-016-1036-5>
- [14] Weisstein, E. W. (n.d.). Trivalent tree. In MathWorld—A Wolfram Web Resource. Retrieved May 24, 2025, from <https://mathworld.wolfram.com/TrivalentTree.html>
- [15] Sloane, N. J. A. (Ed.). (n.d.). A000672: Number of 3-valent trees (= boron trees or binary trees) with  $n$  nodes. The On-Line Encyclopedia of Integer Sequences. Retrieved June 19, 2025, from <https://oeis.org/A000672>
- [16] Weisstein, E. W. (n.d.). Trivalent tree. MathWorld—A Wolfram Resource. Retrieved July 4, 2025, from <https://mathworld.wolfram.com/TrivalentTree.html>
- [17] Unrooted binary tree. (n.d.). Wikipedia. Retrieved July 30, 2025, from [https://en.wikipedia.org/wiki/Unrooted\\_binary\\_tree](https://en.wikipedia.org/wiki/Unrooted_binary_tree)
- [18] Sloane, N. J. A. (Ed.). (n.d.). OEIS. Retrieved July 30, 2025, from <https://oeis.org/A274056>
- [19] That's Maths. (2015, January 29). The Steiner minimal tree. <https://thatsmaths.com/2015/01/29/the-steiner-minimal-tree/>
- [20] Stanley, R. P. (2015). Catalan numbers. Cambridge University, England: Cambridge University Press. <https://doi.org/10.1017/CB09781139871495>

- [21] Catalan Numbers. (n.d.). Brilliant.org. Retrieved September 24, 2025, from <https://brilliant.org/wiki/catalan-numbers/>
- [22] Catalan number. (n.d.). Wikipedia. Retrieved September 24, 2025, from [https://en.wikipedia.org/wiki/Catalan\\_number](https://en.wikipedia.org/wiki/Catalan_number)

# A Appendix

## A.1 Python Code for Generating Lower Bound

The following Python code implements the algorithm described in the Methodology:

Listing 1: Python code used to compute the lower bound of hexagonal tree topologies

```
1  """
2  Author: Shalen Boeja
3
4  A Python program used to generate all unique minimal Steiner tree topologies
5  for a given number
6  of terminals, filtering duplicates via graph isomorphism to provide a lower
7  bound on the total
8  number of distinct topologies.
9  """
10
11 import networkx as nx
12 import matplotlib.pyplot as plt
13 from itertools import combinations
14
15 # --- Graph Isomorphism Checker ---
16 def is_isomorphic(g1, g2):
17     return nx.is_isomorphic(g1, g2)
18
19 # --- Base Tree for n=3 Terminals ---
20 def base_tree():
21     G = nx.Graph()
22     G.add_edges_from([("s1", "T1"), ("s1", "T2"), ("s1", "T3")])
23     return G
24
25 # --- Expand Tree by Attaching 1 Steiner + 2 Terminals to Each Terminal ---
26 def expand_tree(tree, s_counter, t_counter):
27     new_trees = []
28     terminals = [node for node in tree.nodes if tree.degree[node] == 1 and
29                 node.startswith("T")]
30     for term in terminals:
31         #checks the current
32         nodes in the tree and saves them
33         new_tree = tree.copy()
34         # creates a copy to
35         expand
36         neighbor = list(new_tree.neighbors(term))[0] # we save the neighbor
37         of the terminals so we can now where to expand after removing the
38         terminal
39         new_tree.remove_node(term)
40         # we remove the terminal
41         from the newtree
42
43         s_node = f"s{s_counter}"
44         # we create a new
45         steiner point with 2 terminals
46         t1_node = f"T{t_counter}"
47         t2_node = f"T{t_counter + 1}"
48
49         new_tree.add_edges_from([(neighbor, s_node), (s_node, t1_node), (
50             s_node, t2_node)]) #connects the neighbor we saved to the new
51         steiner mnode with terminals
52         new_trees.append(new_tree)
53         # saves the new tree in
54         new_trees
55
56     s_counter += 1
57     #increment the counter so the next nodes
58     names are different
```

```

40         t_counter += 2
41
42     return new_trees, s_counter, t_counter
43
44 # --- Prune Isomorphic Trees ---
45 def unique_trees(trees):
46     unique = []
47     for t in trees:           #we go through every tree in the list
48         if not any(is_isomorphic(t, u) for u in unique):    # we compare t to
49             unique.append(t)                                # all u's in unique
50             unique.append(t)                                # if t is not
51             unique.append(t)                                # isomorphic to any u in unique then we add t to unique
52     return unique
53
54 # --- Relabel Graph Nodes Consistently ---
55 def relabel_nodes_consistently(G):           #his defines a function that
56     takes in a graph G
57     terminals = sorted([n for n in G.nodes if G.degree[n] == 1], key=lambda x:
58         x) #finds all terminals in G and sorts them
59     steiners = sorted([n for n in G.nodes if G.degree[n] == 3], key=lambda x:
60         x)
61     mapping = {}           #We create an empty dictionary to store the new names. We
62     'll use this to tell NetworkX which old names should map to which new
63     names.
64
65     for i, t in enumerate(terminals, 1): #we go through every terminal one by
66         one
67         mapping[t] = f"T{i}"           #This builds up a dictionary that says:
68         rename this to that.
69     for i, s in enumerate(steiners, 1):
70         mapping[s] = f"s{i}"
71
72     return nx.relabel_nodes(G, mapping)     #This uses a built-in NetworkX
73     function that applies the renaming. It returns a new graph where all
74     the nodes have nice clean labels.
75
76 # --- Generate Unique Topologies with Exactly n Terminals ---
77 def generate_topologies(n_terminals): #This function generates all unique
78     minimal Steiner tree topologies for a given number of terminals.
79     if n_terminals == 1: #If someone asks for 1 terminal, just return a graph
80         with one node: T1. Theres no Steiner point just a dot.
81         G = nx.Graph()
82         G.add_node("T1")
83         return [G]
84     elif n_terminals == 2: #If someone asks for 2 terminals, just return a
85         graph with one node: T1, T2
86         G = nx.Graph()
87         G.add_edges_from([("T1", "T2")])
88         return [G]
89     elif n_terminals == 3:
90         return [base_tree()] ##If someone asks for 3 terminals, give the base
91         tree
92
93 trees = [base_tree()]

```

```

83     s_counter = 2 # since s1 is already used in base tree
84     t_counter = 4 # since T1, T2, T3 are used in base tree
85
86     while True: #We loop until we have a tree with exactly n_terminals
87         terminals.
88         next_gen = [] #We collect all expanded trees in next_gen.
89         for t in trees:
90             expanded, s_counter, t_counter = expand_tree(t, s_counter,
91                 t_counter)
92             next_gen.extend(expanded) #We expand all its terminals using the
93                 expand_tree function. That gives us new trees with 1 more
94                 terminal per expansion
95
96         unique = unique_trees(next_gen) #We remove duplicate topologies using
97             unique_trees
98
99         # Check if any tree has desired number of terminals
100        if unique:
101            filtered = [t for t in unique if sum(1 for node in t.nodes if node
102                .startswith("T") and t.degree[node] == 1) == n_terminals] #We
103                keep only trees that exactly match the terminal count we want.
104            if filtered:
105                return [relabel_nodes_consistently(t) for t in filtered] #If
106                we found trees with the correct number of terminals: We
107                clean up the node names using relabel_nodes_consistently()
108                . Then return them
109
110        trees = unique
111
112    # --- Plotting Helper ---
113    def plot_graph(G, idx):
114        pos = nx.spring_layout(G, seed=idx)
115        degrees = dict(G.degree())
116        colors = ["red" if node.startswith("s") else "skyblue" for node in G.nodes
117            ()]
118        plt.figure(figsize=(4, 4))
119        nx.draw(G, pos, with_labels=True, node_color=colors, node_size=600)
120        plt.title(f"SMT Topology #{idx + 1}")
121        plt.axis('off')
122        plt.show()
123
124    # --- Main Execution ---
125    if __name__ == '__main__':
126        n = 10 # target number of terminals
127        topologies = generate_topologies(n)
128        print(f"Number of unique SMT topologies for {n} terminals: {len(topologies
129            )}")
130
131        for idx, tree in enumerate(topologies):
132            plot_graph(tree, idx)

```

A.2 Code output for various  $n$

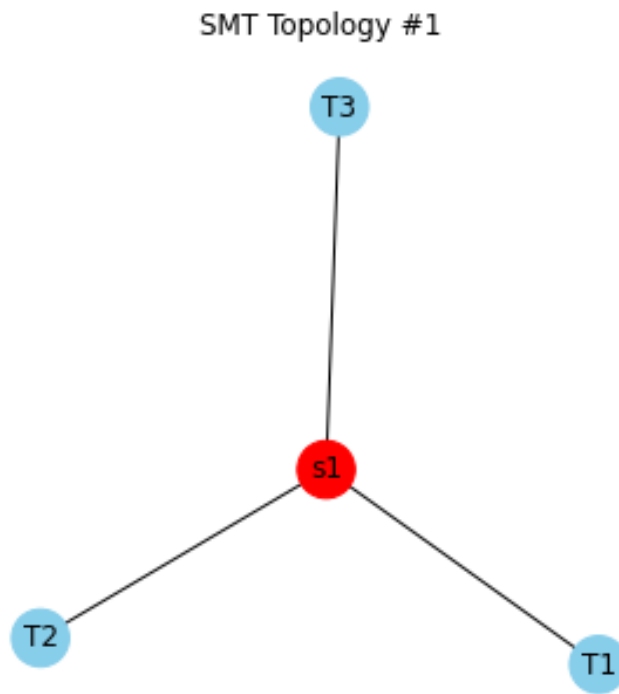


Figure 20:  $n = 3$

SMT Topology #1

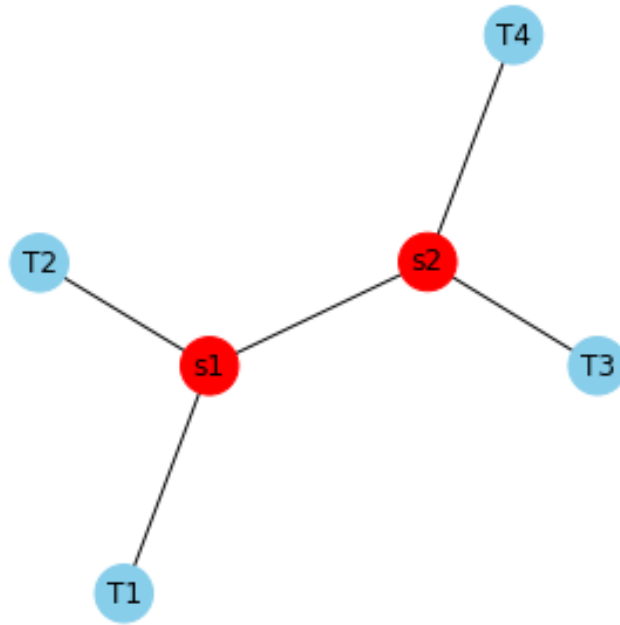


Figure 21:  $n = 4$

SMT Topology #1

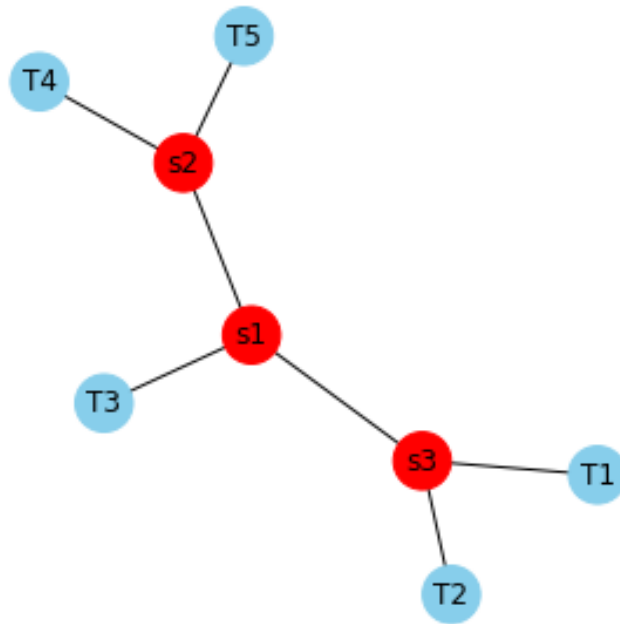


Figure 22:  $n = 5$

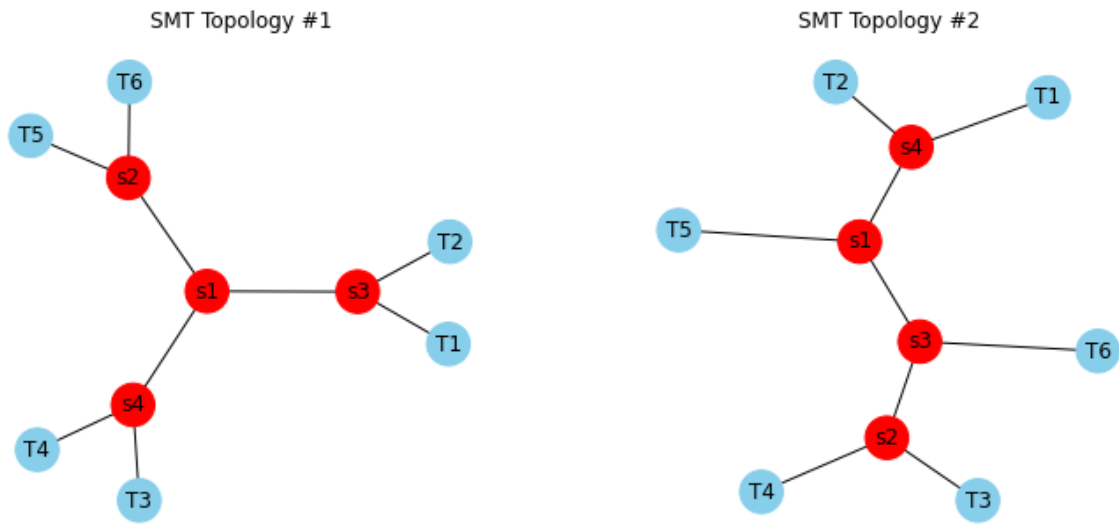


Figure 23:  $b n = 6$

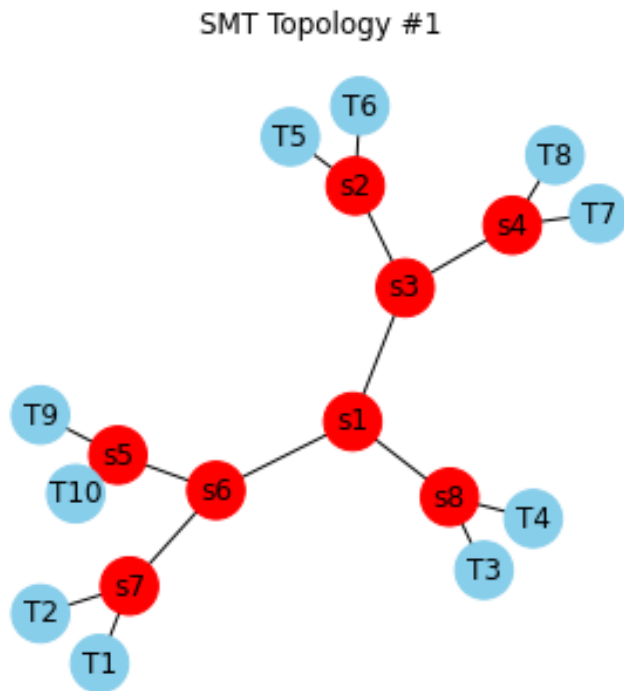


Figure 24: One of the computed unique topologies for  $n = 10$