NP-hard Networking Problems

Exact and Approximate Algorithms

Dissertation zur Erlangung des Grades
des Doktors der Naturwissenschaften (Dr. rer. nat.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

vorgelegt von

Rouven Naujoks

Saarbrücken
2008

Tag des Kolloquiums: 22. Dezember 2008

Dekan der Naturwissenschaftlich-Technischen Fakultät I:
Professor Dr. Joachim Weickert

Berichterstatter:
Professor Dr. Kurt Mehlhorn, Max-Planck-Institut für Informatik, Saarbrücken
Priv.-Doz. Dr. Ernst Althaus, Max-Planck-Institut für Informatik, Saarbrücken
Professor Dr. Stefan Funke, Universität Greifswald

Mitglieder des Prüfungsausschusses:
Professor Dr. Joachim Weickert (Vorsitzender)
Professor Dr. Kurt Mehlhorn
Priv.-Doz. Dr. Ernst Althaus
Dr. Stefan Canzar

meiner Großmutter

Acknowledgements

First of all, I would like to thank my supervisor, Ernst Althaus, for his help, his support and his collaboration. I would also like to thank Ernst Althaus, Stefan Funke and Kurt Mehlhorn for agreeing to coreferee this thesis and Stefan Funke, Sören Laue and Zvi Lotker for the cooperation in various results of this thesis.

Special thanks go to my family and to all my friends.

Zusammenfassung

In verschiedenen wissenschaftlichen Disziplinen, wie der Biologie, der Lingustik und dem Entwurf kabelloser Kommunikationsnetzwerke, wird man mit der Konstruktion von Verbindungsnetzwerken über einer gegebenen Menge von Objekten konfrontiert. Diese Netzwerke sollen bestimmte Eigenschaften erfüllen und gleichzeitig eine gegebene Kostenfunktion minimieren. In dieser Arbeit werden NP-schwere Netzwerkprobleme dieser Art behandelt. Die Arbeit untergliedert sich in zwei Teile. Im ersten Teil beschäftigen wir uns hauptsächlich mit dem sogenannten Steinerbaumproblem in der Hamming-Metrik. Die Berechnung solcher Bäume hat sich als eines der Hauptwerkzeuge in der Rekonstruktion abstammungsgeschichtlicher Beziehungen zwischen Spezien herausgestellt. Wir geben einen neuen, exakten Algorithmus, welcher der Branch-and-Bound-Methode von Hendy und Penny deutlich überlegen ist. Diese galt in den letzten 25 Jahren als die schnellste Methode zur Berechnung solcher Bäume. Des Weiteren stellen wir ein erweitertes Modell vor, welches die Fälle behandelt, in denen die abstammungsgeschichtlichen Beziehungen bestmöglich durch eine nicht baumartige Struktur beschrieben wird. Im zweiten Teil beschäftigen wir uns mit verschiedenen Problemen, wie sie bei dem Entwurf kabelloser Ad-hoc-Netzwerke auftreten: Unter denjenigen Kommunikationsstrukturen, die bestimmte Kommunikationsarten zulassen, versucht man diejenige zu finden, welche die Stromaufnahme des Netzwerkes minimiert. Wir zeigen, wie für diese Probleme approximative Lösungen gefunden werden können.

Abstract

An important class of problems that occur in different fields of research such as biology, linguistics or in the design of wireless communication networks, deal with the problem of finding an interconnection of a given set of objects. Additionaly, these networks should satisfy certain properties and minimize a certain cost function. In this thesis, we discuss such NP-hard networking problems in two parts. First, we mainly deal with the so-called Steiner minimum tree problem in Hamming metric. The computation of such trees has become a key tool for the reconstruction of the ancestral relationships of species. We give a new exact algorithm that clearly outperforms the branch and bound based method of Hendy and Penny which has been considered to be the fastest for the last 25 years. Further, we propose an extended model to cope with the case in which the ancestral relationships are best described by a non-tree structure. Finally, we deal with several problems occurring in the design of wireless ad-hoc networks: While minimizing the total power consumption of a wireless communication network, one wants to establish a messaging structure such that certain communication tasks can be performed. We show how approximate solutions can be found for these problems.

Contents

1 The Steiner Minimum Tree in Hamming Metric Problem **11**

1.1 Our Contribution . . . 12
1.2 Related Work . . . 13
1.3 Preliminaries . . . 15
1.3.1 Fitch's Algorithm . . . 23
1.4 Pruning Algorithm . . . 24
1.5 Pruning Tests . . . 26
1.5.1 Edge Replacement Tests . . . 26
1.5.2 Topology Replacement Tests . . . 41
1.5.3 Preprocessing Techniques . . . 47
1.5.4 Implementation Issues . . . 48
Cascading Pruning Tests . . . 48
Amortized Fitch Range Costs . . . 48
Topology Replacement Tests . . . 49
1.6 Lower Bounds . . . 49
1.6.1 Lower Bounds by Minors . . . 49
1.6.2 Lower Bounds by Dimension Partitioning . . . 50
One-Dimensional Steiner Minimum Trees . . . 53
Two-Dimensional Steiner Minimum Trees . . . 53
Three-Dimensional Steiner Minimum Trees . . . 54
k-Dimensional Steiner Minimum Trees . . . 57
1.6.3 Implementation Issues . . . 58
Cascading Lower Bounds . . . 58
Maximum Weight Matchings . . . 58
Computing the Subproblems . . . 59
Computing MST Lower Bounds . . . 60
Inner Preprocessing . . . 61
1.7 Experiments . . . 61
1.7.1 Running Times . . . 61
1.7.2 Lower Bounds . . . 63
1.7.3 Preprocessing . . . 63
1.8 Conclusion . . . 63
1.9 An Extension to Recombination Networks . . . 72

1.9.1 Model . . . 72
1.9.2 Related Work . . . 73
1.9.3 The Algorithm . . . 74
Preliminaries . . . 74
Evaluation of a Recombination Network . . . 74
Enumeration Process . . . 75
Pruning the Search Space . . . 77
Recombination Phase . . . 78
1.9.4 Experiments . . . 79
Fixed Recombination Scenarios . . . 79
Results . . . 80
1.9.5 Conclusion . . . 81

2 Problems in Wireless Network Design **83**
2.1 Introduction . . . 83
2.2 Preliminaries . . . 84
2.3 The k-Station Network/k-Disk Coverage Problem . . . 87
2.3.1 Our Contribution . . . 87
2.3.2 Related Work . . . 88
2.3.3 A Small Coreset For k-Disk Cover . . . 88
2.3.4 Algorithms . . . 91
Discrete Version . . . 91
Non-Discrete Version . . . 91
2.3.5 k-Disk Cover With Few Outliers . . . 92
2.4 The k-Hop Multicast Problem . . . 93
2.4.1 Our Contribution . . . 94
2.4.2 Related Work . . . 94
2.4.3 Preliminaries . . . 95
2.4.4 A Small Coreset For k-hop Multicast . . . 95
2.4.5 Solution Via a Naive Algorithm . . . 97
2.5 The k-Set Broadcast Problem . . . 97
2.5.1 Our Contribution . . . 98
2.5.2 Related Work . . . 99
2.5.3 Preliminaries . . . 100
2.5.4 Algorithms . . . 100
A Naive, Brute-Force Algorithm . . . 100
Small Coreset of the Network Topology . . . 101
Faster $O(1)$-Approximations . . . 102
2.6 TSP Under Squared Euclidean Distance . . . 105
2.6.1 Our Contribution . . . 106
2.6.2 Related Work . . . 106
2.6.3 Why Euclidean TSP Does Not Work . . . 106
2.6.4 A 6-Approximation Algorithm . . . 107
2.7 Conclusion . . . 108

Bibliography **109**

List of Figures

1.1 Three points A, B and C and the corresponding Torricelli point P. . . . 11
1.2 The first two levels of the recursion tree of Hendy and Penny's algorithm. 14
1.3 A concatenation of two rooted topologies. 20
1.4 An example of Fitch's algorithm on one character terminals. 23
1.5 An example of the propagated Fitch range function on one character terminals. 29
1.6 An example of the restricted Fitch algorithm on one character terminals. 32
1.7 An example of the non-optimality of the $\|\cdot\|_m$ bound. 37
1.8 An example of the MST-substitute property. 45
1.9 An example demonstrating the performance of the MST-substitute test. 46
1.10 An example of the modified counting sort algorithm. 59
1.11 Proof illustration for Lemma 1.11. 61
1.12 The input trees A and B for the Seq-Gen data. 64
1.13 The three different cases occurring in the lower bound pruning step. . . 77
1.14 Pruning by minors. 79

2.1 Points, a range assignment and the induced communication graph. . . . 85
2.2 Covering input points by 3 discs. 87
2.3 Covering input points by 3, ignoring 3 outliers. 92
2.4 A 4-hop multicast communication graph. 94
2.5 A 4 sender broadcast communication graph. 99
2.6 Proof illustration for the constant factor approximation algorithm. . . . 104
2.7 An optimal energy-minimal tour for points on a line 106
2.8 Tree T and its child trees $T_1, T_2, \ldots, T_k$ 106

List of Tables

1.1 The effect of the preprocessing methods. 65
1.2 A running time comparison - generated data. 66
1.3 A running time comparison - real life data. 67
1.4 A lower bound comparison. 70
1.5 The comparison of Recco and Recomb. 81
1.6 The reliability of Recomb and maximum parsimony I. 81
1.7 The reliability of Recomb and maximum parsimony II. 81

Chapter 1

The Steiner Minimum Tree in Hamming Metric Problem

The Steiner minimum tree problem has a long history starting in the middle of the 17th century. Pierre de Fermat (1601-1665) stated the following problem: Given three points A, B and C in the Euclidean plane, find a fourth point P such that the sum of the lengths of the line segments $\overline{PA}, \overline{PB}, \overline{PC}$ is minimized. The problem was approached from different points of view by Vincenzo Viviani (1622-1703) and Evangelista Torricelli (1608-1647) - to who's honor the point P is also referred to as the Torricelli-point P (see Figure 1.1). In this context, Bonaventura Cavalieri (1598-1647) should not remain unmentioned as it was he who found an easy way to construct P by the circumcircles of the isosceles triangles attached to $\overline{PA}, \overline{PB}$ and $\overline{PC}$.

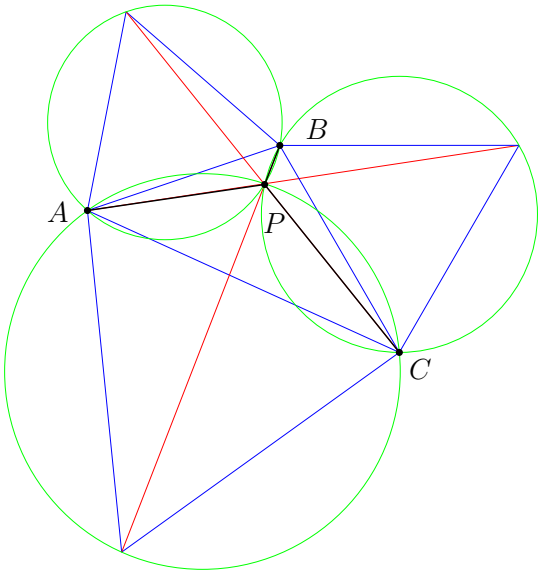


Figure 1.1: Three points A, B and C and the corresponding Torricelli point P.

The generalized Fermat problem reads as follows: Given a set N of points in the

Euclidean plane, find a set S of additional points and line segments with endpoints in N or in S, such that for all pairs of points in N there is a path of line segments connecting them, while minimizing the sum of the lengths of the line segments. In mathematics, it is sometimes quite unclear why things like theorems or problems are named as they are. It is not different in the case of the Fermat problem which today is referred to as the Steiner problem even though Jakob Steiner (1796-1863) did not contribute much to the problem as Schreiber notes in [Sch86].

The problem can be generalized to the Steiner tree problem in graphs: Given a weighted graph $G = (S \cup N, E, w)$ with nonnegative edge weights where the nodes are partitioned into the set of terminal nodes N and the set of Steiner nodes S, find a tree of minimal total length spanning the nodes in N. The core of this problem lies in its restriction to instances in which the edge cost function w satisfies the triangle inequality, called the metric Steiner tree problem in graphs, as there is an approximation factor preserving reduction from the unconstrained version of the problem to the metric one (see [Vaz03]).

Unfortunately, most variants of the Steiner tree problem are known to be NP-hard. In fact, one of them was among the 21 problems for which Richard Karp showed in his seminal work in 1972 [Kar72] NP-completeness. While most of these variants do not even admit a polynomial time approximation scheme, constant factor approximation algorithms for the Steiner tree problem in graphs are known (see for example [Meh88], [GR00]).

In recent years, a lot of work has been invested in the development of algorithms that solve the Steiner tree problem exactly, without guaranteeing polynomial running times but rather exhibiting good performance in practice. Most of these algorithms focus on special settings, as does the approach of Warme, Winter and Zachariasen [WWZ00] for computing exact solutions for the problem in the Euclidean plane or another approach by Zachariasen [Zac99] for solving the rectilinear Steiner minimum tree problem in the plane. An algorithm for solving instances for the Steiner tree in graphs problem was given by Plozin [Pol03].

The version we consider in this thesis is the Steiner tree problem in Hamming metric where the metric space consists of strings over some alphabet Σ. The Hamming metric between two strings is simply the number of character-wise differences of the strings. Like the other versions mentioned above, this problem is also known to be NP-hard [FG82]. Computing such trees is well studied and has applications in several fields of science such as computational linguistics and computational biology. In both fields, the strings represent sets of characteristics of given objects of which one wants to reconstruct an ancestral relationship. For evaluating the implementation of the algorithm that we will present in this thesis, we focus on the biological context in which the strings represent, for example, parts of the genome of a species. In this context the problem is also known as the maximum parsimony problem and the trees under consideration are called phylogenetic trees.

1.1 Our Contribution

Among all methods for computing Steiner minimum trees optimally, algorithms using variations of a branch and bound method found by Hendy and Penny [HP82] have been

the fastest for more than 25 years. We will describe a new pruning based approach which is superior to previous methods, discuss its implementation and demonstrate, by means of artificial and real-world biological data sets, its performance compared to free and commercial software tools. Part of this work was published in [AN06].

1.2 Related Work

We now discuss related work in more detail. As mentioned before, Warme et al. have shown in [WWZ00] how one can compute solutions for the Steiner tree problem in the Euclidean plane. Since our algorithm conceptually uses some ideas of their approach, let us now outline their algorithm. For a Steiner tree T, we call the maximal subtrees, whose internal vertices are Steiner points, the full components of T. We can subdivide the computation of a Steiner tree into two phases. First, a superset X of the set of the full components of T is computed. Afterwards, the minimum cost subset of X yielding a tree is determined. The second part is typically solved by an integer linear programming approach. One can observe that in the case of the Euclidean plane, the size of the full components is very small (typically less than 10, even in instances with 10.000 terminals). Although there are instances where the algorithm will create an exponential number of candidate full components, experiments indicate a number of full components in random instances that is roughly linear. Unfortunately, in Hamming metric, a Steiner minimum tree consists typically of only a small number of – if not only one – full components. Since the efficiency of their approach heavily depends on partitioning the instance into many small full components, one cannot apply this technique to our problem setting. But as we will explain later, our algorithm will also use this two phase technique of first constructing a set of small trees and then combining these small trees into full trees.

In graphs, the most successful algorithm for solving the Steiner minimum tree problem is based, among other techniques, on successfully pruning parts of the graph, that is, discarding graphs that contain substructures that lead to provable non-optimal trees. All known techniques are described in the PhD thesis of Polzin [Pol03]. We will adopt some ideas of these pruning techniques in this work.

There are many papers on computational methods for solving the maximum parsimony problem. As mentioned before, almost all papers describing exact methods for solving the maximum parsimony problem, describe variants of the branch and bound algorithm of Hendy and Penny [HP82] (for example [Fel04, Swo03, KTND07]). For an extensive list of the available software, see [WEBa].

Before we start explaining Hendy and Penny's algorithm, note that without loss of generality, one can assume that in an optimal Steiner tree all inner nodes have degree 3 and that the leaves of the tree are exactly the points to be spanned (justified by Theorem 1.1). If not mentioned otherwise, we will assume that the trees under consideration satisfy this property. The algorithm starts with a tree T with 3 terminals as leaves. Now it recursively increases the tree by removing each edge $e = \{u, v\}$ in T, by adding a Steiner point p and by connecting u, v and a so far not spanned terminal w to p (see Figure 1.2). If the cost of the current tree $\mathcal{T}$ plus a lower bound for adding the remaining points to it is larger than the best known Steiner tree, we backtrack the recursion, knowing that any tree constructed by increasing the current one has a cost

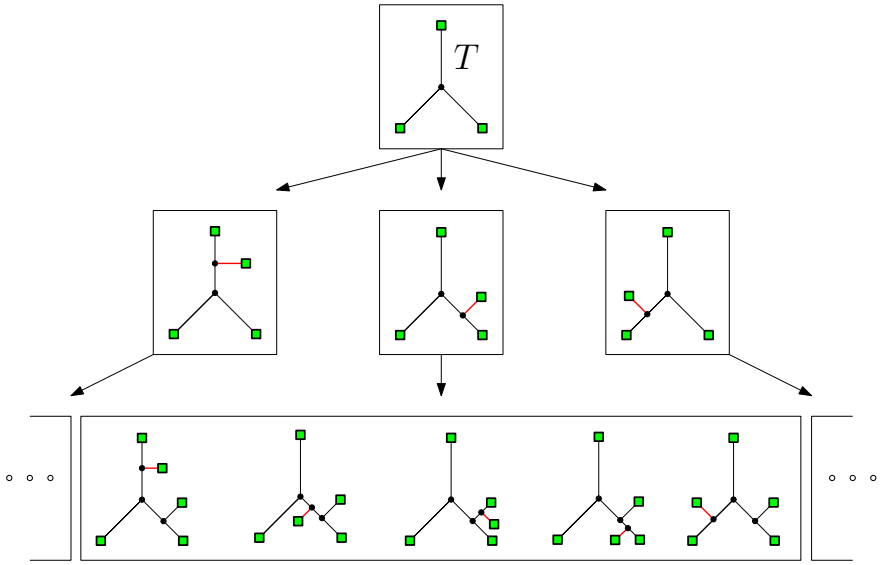


Figure 1.2: The first two levels of the recursion tree of Hendy and Penny's algorithm. The green boxes indicate spanned points, the black discs Steiner points. Red edges represent newly inserted edges.

strictly larger than the cost of an optimal one.

The probably main disadvantage of this approach lies in the bad quality of the lower bounds that one can compute. Note that in each step of the algorithm, a new terminal is inserted at an arbitrary position. Thus, intuitively speaking, we have no structure that we could exploit in finding good lower bounds.

But how can one determine the cost of such a tree? The cost of a tree, which is defined as the sum of the cost of its edges, depends on how we label the Steiner nodes, that is, on the strings that we assign to them. Fitch has shown in [Fit71] that one can label the inner nodes of a given tree optimally in time $O(n \cdot d \cdot |\Sigma|)$, where d denotes the length of the strings, such that the cost of the tree is minimized. We will see in section 1.3 how Fitch's algorithm works.

Holland et al. [HHPM05] attempt to prove the optimality of parsimony trees by computing tight lower bounds that hopefully match upper bounds. The authors use so-called one and two-column discrepancy bounds (see Section 1.6) for subproblems with the aim of combining these bounds to a lower bound for the complete problem instance by a simple relaxation of an integer linear program, as defined in Section 1.6.

Beside the maximum parsimony problem, the second prominent way of reconstructing a phylogenetic tree is the maximum likelihood tree (which is not part of this thesis) where the assumption is made that the characters are pairwise independent and that the branching follows a Markov process. The optimal tree is the most likely tree under this process. Even though both methods are currently used by biologists, there is no clear answer to the question of which method provides a more "realistic" reconstruction of a phylogeny. We refer to [Ste05] for a more detailed discussion of this problem.

1.3 Preliminaries

First we fix some notation: Let $s \in \Sigma^d$ be a string of length d over the alphabet Σ. We denote by s_i the i-th character of the string. We also say that a string $s \in \Sigma^d$ is a point in the space Σ^d.

Given two functions $\mu_1 \colon X_1 \to Y$ and $\mu_2 \colon X_2 \to Y$ where $X_1 \cap X_2 = \emptyset$, we define

$$\mu_1 \odot \mu_2 \colon X_1 \cup X_2 \to Y; v \mapsto \begin{cases} \mu_1(v) & \text{if } u \in X_1 \\ \mu_2(v) & \text{if } u \in X_2 \end{cases}$$

For a set X we define $\mathcal{P}_n(X) := \{Y \subseteq X \mid |Y| = n\}$ to be the set of all subsets of X of cardinality n. Furthermore, we write $\mathcal{P}(X) := \{Y \subseteq X\}$ for the power set of X, that is, the set of all subsets of X.

Given a function $\mu \colon X \to Y$ and a set $\mathcal{X} \subseteq X$, we define the *image* of μ on $\mathcal{X}$ to be

$$\mathrm{Im}_{\mathcal{X}}(\mu) := \{\ \mu(x) \mid x \in \mathcal{X}\ \}$$

For $\mathrm{Im}_X(\mu)$ we also write $\mathrm{Im}(\mu)$.

Given a function $\mu \colon X \to Y$ and a set $\mathcal{X} \subseteq X$, we define the *restriction* of μ on $\mathcal{X}$ to be the function $\mu|_{\mathcal{X}} \colon \mathcal{X} \to Y;\ u \mapsto \mu(u)$.

Given a function $\mu \colon X \to Y^d$. For $i \in \{1, \ldots, d\}$ we define

$$\mu_i \colon X \to Y;\ x \mapsto \mu(x)_i$$

For a multiset M we denote by $\mathrm{m}_A(x)$ the multiplicity of x in A.

Given a rooted tree $\mathcal{T} = (V, E)$, we denote by $\mathrm{root}(\mathcal{T})$ the root of $\mathcal{T}$. If the last edge on the unique path from $\mathrm{root}(\mathcal{T})$ to a node x is (y, x) then $\mathrm{parent}(x) := y$ is the *parent* of x, and x is a *child* of y. If two nodes x and y are children of the same node, x is called a *sibling* of y. A node with no children is a *leaf*. We denote by $\mathrm{leaves}(\mathcal{T})$ the set of leaves of $\mathcal{T}$. A non-leaf node is an *internal* node. For a node $v \in V$, any node u on the unique path from $\mathrm{root}(\mathcal{T})$ to v is called an *ancestor* of v. If u is an ancestor of v then we call v a successor of u. For the set of ancestors of a node u we write $\mathrm{anc}(u)$ and for the set of successors of u, $\mathrm{succ}(u)$.

Given a graph $G = (V, E)$, we denote by $u \rightsquigarrow v$ a path in G from a node $u \in V$ to a node $v \in V$.

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ then we define

$$(G_1 \cup G_2) := (V_1 \cup V_2, E_1 \cup E_2)$$

Definition 1.1 (Hamming distance) *Given two strings $s^1, s^2 \in \Sigma^d$, the value*

$$\|s^1, s^2\| := |\{\ i \in \{1, \ldots, d\} \mid s^1_i \neq s^2_i\ \}|$$

is called the Hamming distance *of the strings s^1 and s^2.*

Lemma 1.1 (Hamming metric) $(\Sigma^d, \|\cdot\|)$ *is a metric space.*

Proof: Let $x, y, z \in \Sigma^d$. Obviously $\|x, y\| = \|y, x\|$. Note that $\|x, y\| = \sum_{i=1}^{d} \|x_i, y_i\|$. Since $\|x_i, y_i\| \geq 0$ we have on the one hand

$$\|x, y\| = 0 \Leftrightarrow \forall i \in \{1, \ldots, d\} : \|x_i, y_i\| = 0 \Leftrightarrow x = y$$

On the other hand we can show by a simple case distinction that $\forall i \in \{1, \ldots, d\} : \|x_i, z_i\| \leq \|x_i, y_i\| + \|y_i, z_i\|$ which directly implies $\|x, z\| \leq \|x, y\| + \|y, z\|$. □

Definition 1.2 (topology) *For our purpose, a* topology *is a pair* $\mathcal{T} = ((V, E), \mu)$ *where* (V, E) *is an undirected tree and* μ *is a function* $\mu\colon T' \to \Sigma^d$ *for a set* $T' \subseteq V$*. We say that* $\mathcal{T}$ *is* fully labeled *if* $T' = V$*, otherwise we call* $\mathcal{T}$ partially labeled*.*

Definition 1.3 (span/size of a topology) *Given a topology* $\mathcal{T} = ((V, E), \mu)$*, we call* $\text{span}(\mathcal{T}) := \text{Im}(\mu)$ *the* span *of* $\mathcal{T}$ *and* $|\text{span}(\mathcal{T})|$ *the* size *of* $\mathcal{T}$*.*

Definition 1.4 (subtopology) *Given two topologies* $\mathcal{T} = ((V, E), \mu\colon T \to \Sigma^d)$ *and* $\mathcal{S} = ((V_s, E_s), \mu_s\colon T_s \to \Sigma^d)$*. We call* $\mathcal{S}$ *a* subtopology *of* $\mathcal{T}$ *if there exists a injective function* $\sigma\colon V_s \to V$ *such that*

- $\forall\ \{u, v\} \in E_s : \{\sigma(u), \sigma(v)\} \in E$
- $\forall\ u \in T_s : \mu(u) = \mu(\sigma(u))$

We call such a function σ *a* subtopology mapping*.*

Note that in the above definition (V_s, E_s) must be connected, since $\mathcal{S}$ is a topology.

Definition 1.5 (cost of a fully labeled topology) *Given a fully labeled topology* $\mathcal{T} = ((V, E), \mu)$*, we associate with* E *the* edge cost function

$$\text{cost}\colon E \to \mathbb{R}_{\geq 0};\ \{u, v\} \mapsto \|\mu(u), \mu(v)\|$$

The cost *of* $\mathcal{T}$ *is defined as* $\text{cost}(\mathcal{T}) := \sum_{e \in E} \text{cost}(e)$*.*

Definition 1.6 (cost of a partially labeled topology) *Let* $\mathcal{T} = ((V, E), \mu\colon T' \to \Sigma^d)$ *be a partially labeled topology, then* $\text{cost}(\mathcal{T})$ *is defined as*

$$\min_{\overline{\mu}\ :\ V \setminus T' \to \Sigma^d} \text{cost}\left(((V, E), \mu \odot \overline{\mu})\right)$$

We call

$$\operatorname*{argmin}_{\overline{\mu}\ :\ V \setminus T' \to \Sigma^d} \text{cost}\left(((V, E), \mu \odot \overline{\mu})\right)$$

a complemental labeling *of* $\mathcal{T}$ *and the function* $\mu \odot \overline{\mu}$ *a* complemented labeling *of* $\mathcal{T}$*.*

Given a topology $\mathcal{T} = ((V, E), \mu_{\mathcal{T}})$. For a labeling $\mu\colon V' \to \Sigma^d$ with $V' \subseteq V$ we denote by $\text{cost}[\mu](\mathcal{T})$ the cost of the topology $((V, E), \mu)$.

Definition 1.7 (Steiner tree in Hamming metric) *Given a set* $T = \{t_1, t_2, \ldots, t_n\} \subseteq \Sigma^d$ *of* terminals*, a* Steiner tree in Hamming metric ST over T *is a fully labeled topology* $((Z \cup S, E), \mu)$ *with* $Z = \{z_1, z_2, \ldots, z_n\}$ *and* $\mu(z_i) = t_i$ *for* $i \in \{1, \ldots, n\}$*. We call* Z *the set of* terminal nodes *and* S *the set of* Steiner nodes *of* ST*.*

Given a Steiner tree in Hamming metric ST, we denote by snodes(ST) the set of Steiner nodes of ST.

Definition 1.8 (Steiner minimum tree in Hamming metric) *Given a set* $T \subseteq \Sigma^d$*, a* Steiner minimum tree in Hamming metric *over* T *(or short:* SMT(T)*), is a Steiner tree in Hamming metric over* T *of minimal cost.*

The task of finding such a tree is called the *Steiner minimum tree in Hamming metric problem.*

Definition 1.9 (column of a problem instance) *Given an instance of the Steiner minimum tree in Hamming metric problem* $T \subseteq \Sigma^d$*, we call the multiset*

$$C^i(T) := \{t_i \mid t \in T\}$$

the i*-th* column *of* T*.*

Definition 1.10 (optimal topology) *Suppose we are given an instance* $T \subseteq \Sigma^d$ *of the Steiner minimum tree problem in Hamming metric. We call a topology* $\mathcal{T}$ *with* $T \subseteq \text{span}(\mathcal{T})$ *optimal with respect to* T *if* $\text{cost}(\mathcal{T})$ *equals the cost of a Steiner minimum tree over* T*.*

Definition 1.11 (partial/full topology) *Given an instance of the Steiner tree problem with the terminal set* $T \subseteq \Sigma^d$*. A topology* $\mathcal{T}$ *is called* partial *with respect to* T *if* $\text{span}(\mathcal{T}) \subseteq T$ *and* full *with respect to* T *if* $\text{span}(\mathcal{T}) = T$*.*

Definition 1.12 (tree in standard form) *Given a tree* $\mathcal{T} = (V, E)$ *and a set* $V' \subseteq V$*. We say that* $\mathcal{T}$ *is in standard form with respect to* V' *if*

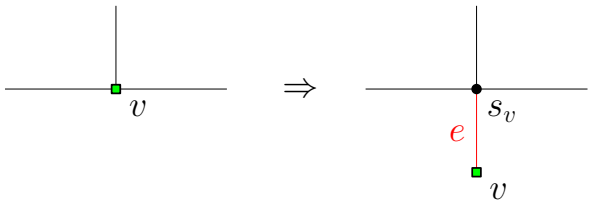
1. $\forall\, v \in V' : \deg(v) = 1$
2. $\forall\, v \in V \setminus V' : \deg(v) = 3$

The following theorem allows us to consider only Steiner trees in standard form for the remainder of this chapter.

Theorem 1.1 (structural lemma) *Given a set* $T = \{t_1, t_2, \ldots, t_n\} \subseteq \Sigma^d$*. Then there is a Steiner minimum tree* $ST = ((V, E), \mu)$ *over* T *with terminal nodes* Z *such that* (V, E) *is in standard form with respect to* Z*.*

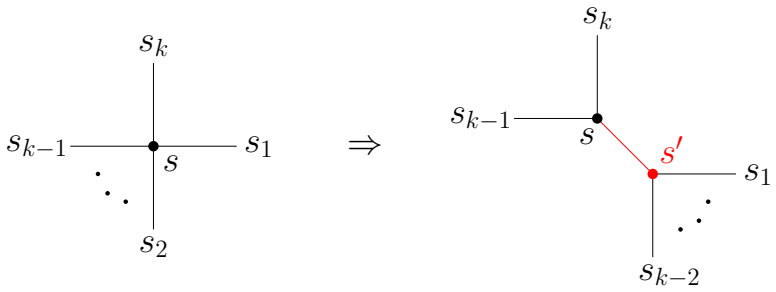
Proof: Suppose we are given a Steiner minimum tree ST'. We will transform ST' into a Steiner tree ST satisfying both conditions without increasing the cost of the tree in the following two steps:

- First consider a node $v \in Z$ that violates constraint 1. Since (V, E) is connected, $\deg(v) > 1$. The idea is to add a new Steiner node s_v, to remove the edges $(\{u_i, v\})_{i \in \{1,\dots,\deg(v)\}}$ incident to v and to add the edges $(\{u_i, s_v\})_{i=1..\deg(v)}$ and $e := \{v, s_v\}$. Furthermore we set $\mu(s_v) := \mu(v)$. Note that for each added edge an edge with the same cost was removed and that $\text{cost}(e) = 0$. Thus we have now $deg(v) = 1$ without increasing the cost of the topology. Furthermore note that ST is still a valid Steiner tree.



  We repeat this procedure until for all $v \in Z$ we have $\deg(v) = 1$.

- Consider now a Steiner node s violating constraint 2. Assume for now that $k := \deg(s) > 3$. We add another Steiner node s' with $\mu(s') := \mu(s)$. Let $s_1, \dots, s_k$ be the adjacent nodes of s. We now remove the edges $(\{s, s_i\})_{i=1..k-2}$. Then we add the edges $\{s', s_i\}_{i=1..k-2}$ and the edge $e = \{s, s'\}$. Note that $\text{cost}(e) = 0$ and that for each other added edge there was an edge that has been removed before with the same cost. Thus we do not increase the cost of the tree by performing this operation but now we have $\deg(s) = 3$ and $\deg(s') < \deg(s)$.



  Now let $\deg(s) = 2$. Then we can simply remove s and reconnect the two adjacent nodes via a new edge e. Since $\|\cdot\|$ is a metric (see lemma 1.1), we do not increase the cost of the tree by doing so. If $\deg(s) = 1$, s and its incident edge can be deleted without violating the validity of the Steiner tree. Repeatedly applying the second case, all Steiner nodes have degree 3.

□

Definition 1.13 (range/range point) *We call a subset of* $\mathcal{P}(\Sigma)$ *a* range *and* $r \in \mathcal{P}(\Sigma)^d$ *a* range point. *For a point* p *and a range point* r *we write*

$$p \in r \quad \textit{if} \quad p_i \in r_i \; \forall \, i \in \{1, \dots, d\}$$

Sometimes we will slightly abuse the notation by considering a point $p \in \Sigma^d$ as a range point by implicitly converting p into $(\{p_1\}, \{p_2\}, \ldots, \{p_d\}) \in \mathcal{P}(\Sigma)^d$ and vice versa.

We use the following notation: $\|x, y\|_{\min}$ denotes the minimal distance between two range points x and y, that is

$$\|x, y\|_{\min} := \min\{\ \|p_x, p_y\| \mid p_x \in x,\ p_y \in y\ \}$$

Similarly, we define

$$\|x, y\|_{\max} := \max\{\ \|p_x, p_y\| \mid p_x \in x,\ p_y \in y\ \}$$

Note that $\|x, y\|_{\min}$ respectively $\|x, y\|_{\max}$ can be computed using $O(d)$ basic set operations since

$$\|x, y\|_{\min} = \{\ i \in \{1, \ldots, d\} \mid x_i \cap y_i = \emptyset\ \}$$

respectively

$$\|x, y\|_{\max} = \{\ i \in \{1, \ldots, d\} \mid \neg(x_i = y_i\ \wedge\ |x_i| = |y_i| = 1)\ \}$$

which can be seen as follows: the case $\|x, y\|_{\min}$ is trivially true, so let us consider the last equality. Let $\|x_i, y_i\|_{\max} = 1$, then (without loss of generality) $\exists\ a \in x_i : a \notin y_i$. Now assume that $x_i = y_i\ \wedge\ |x_i| = |y_i| = 1$ holds. Thus both $x_i = y_i = \{b\}$ for some element b. But this is a contradiction to the fact that there is an $a \in x_i : a \notin y_i$. Now let us assume that $\neg(x_i = y_i\ \wedge\ |x_i| = |y_i| = 1)$ holds, which is equivalent to $x_i \neq y_i \vee |x_i| = 1 \vee |y_i| = 1$. If $x_i \neq y_i$ then clearly there must be an element in one of the sets that is not contained in the other and thus $\|x_i, y_i\| = 1$. Let without loss of generality $|x_i| \neq 1$. Since $|x_i| > 0$, x_i contains at least two distinct elements, let us say b_1 and b_2. Thus for any element $c \in y_i$ either $b_1 \neq c$ or $b_2 \neq c$. Thus $\|x_i, y_i\| = 1$

Definition 1.14 (rooted topology) *A topology $\mathcal{T} = ((V, E), \mu)$ with a specific root node $\mathrm{root}(\mathcal{T}) \in V$ is called a* rooted topology. *We also write for $\mathcal{T}$: $((V, E), \mu, \mathrm{root}(\mathcal{T}))$.*

Definition 1.15 (induced rooted topology) *Given a topology $((V, E), \mu)$ and an edge $e_r = \{u, v\} \in E$, the rooted topology $((V \cup \{r\}, E'), \mu)$ with $r \notin V$,*

$$E' = E \cup \{\{r, u\}, \{r, v\}\} \setminus \{e_r\}$$

and with root r is called the rooted topology induced *by e_r.*

In the next section it will be necessary to undo the operation of inducing a rooted topology for a certain type of rooted topologies. For this purpose we define

Definition 1.16 (de-rooted topology) *Given a rooted topology $\mathcal{T} = ((V, E), \mu\colon T' \to \Sigma^d, r)$ with $\deg(r) = 2$ and $r \notin T'$. We denote by $\overline{\mathcal{T}}$ the unrooted topology*

$$(\ (V \setminus \{r\},\ E \setminus \{\{r, c_2\}, \{r, c_1\}\}) \cup \{\{c_1, c_2\}\}),\ \ \mu\ )$$

where c_1 and c_2 are the two adjacent nodes of r.

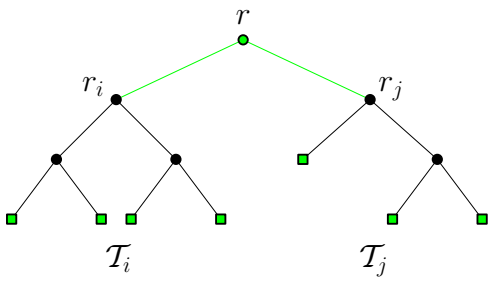


Figure 1.3: A concatenation of two rooted topologies $\mathcal{T}_i$ and $\mathcal{T}_j$ with roots r_i and r_j. The green edges indicate added edges.

Definition 1.17 (concatenation of rooted topologies) *Given two rooted topologies* $\mathcal{T}_1 = ((V_1, E_1), \mu_1 \colon T_1 \to \Sigma^d, r_1)$ *and* $\mathcal{T}_2 = ((V_2, E_2), \mu_2 \colon T_2 \to \Sigma^d, r_2)$ *with* $T_1 \cap T_2 = \emptyset$*. The concatenation* $\mathcal{T}_1 \cdot \mathcal{T}_2$ *of these topologies is a rooted topology*

$$((V, E), \mu_1 \odot \mu_2, r)$$

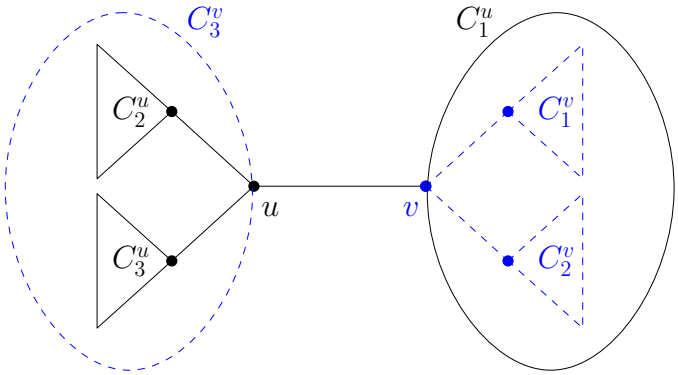
with

$$V = V_1 \cup V_2 \cup \{\, r \,\},$$
$$E = E_1 \cup E_2 \cup \{\{r, r_1\}, \{r, r_2\}\},$$

(see Figure 1.3).

Theorem 1.2 (tree dissection) *Given a tree* $\mathcal{T} = (V, E)$ *with* $|V| \geq 4$ *such that* (V, E) *is in standard form with respect to its leaves. Then there exists an inner node* $v \in V$ *such that removing the node and its incident edges from* $\mathcal{T}$ *decomposes* $\mathcal{T}$ *in* 3 *connected components such that none of them spans more than* $\lfloor l/2 \rfloor$ *leaves of* $\mathcal{T}$*, where* $l := |\text{leaves}(\mathcal{T})|$

Proof: Given an inner node $u \in V$ (note that such a node must exist because $|V| \geq 3$). Removing u from V and its adjacent edges from E decomposes $\mathcal{T}$ into the three connected components $C_1^u = (V_1, E_1)$, $C_2^u = (V_2, E_2)$ and $C_3^u = (V_3, E_3)$. Without loss of generality let $|V_1| \geq |V_2| \geq |V_3|$. If $|V_1| < \lfloor l/2 \rfloor$ the claim holds trivially – so let us assume otherwise. Consider now the inner node $v \in V_1$ with $\{u, v\} \in E$. Note that if such a node does not exist the component C_1^u would consist of just one leaf. Since C_1^u was the biggest of these components, all other components would also contain just one leaf. Thus the claim would trivially hold. If such a node v exists, removing v and its incident edges from $\mathcal{T}$ again decomposes $\mathcal{T}$ into the three connected components C_1^v, C_2^v and C_3^v as shown in the following figure.



We know that $\text{span}(C_3^v) = \text{span}(C_2^u) \cup \text{span}(C_3^u)$ and thus $|\text{span}(C_3^v)| < \lfloor l/2 \rfloor$ since $|\text{span}(C_1^u)| > \lfloor l/2 \rfloor$. Let without loss of generality C_1^v be the largest of the three components (C_i^v). Since v was an inner node $\text{span}(C_1^v) \subsetneq \text{span}(C_1^u)$. Now let us recall what we have: u and v induced partitions of the leaves of $\mathcal{T}$ into

$$(L_1^s, L_1^b)_1 := (\text{span}(C_2^u) \cup \text{span}(C_3^u), \text{span}(C_1^u))$$

and into

$$(L_2^s, L_2^b)_2 := (\text{span}(C_2^v) \cup \text{span}(C_3^v), \text{span}(C_1^v))$$

with $|L_1^s| < \lfloor l/2 \rfloor$, $|L_2^s| < \lfloor l/2 \rfloor$ and $L_2^b \subsetneq L_1^b$. Repeating the procedure until a node is found for which the claim holds, yields a sequence $(L_i^s, L_i^b)_{i \in \mathbb{N}}$ with $|L_i^s| < \lfloor l/2 \rfloor$ and $L_i^b \subsetneq L_j^b$ for $i > j$. Since $|L_i^b|$ is monotonically decreasing, the sequence is finite, thus proving the existence of a node u so that the claim holds. □

Definition 1.18 (topology dissection) *Given a topology*

$$\mathcal{T} = ((V, E), \mu \colon T' \to \Sigma^d)$$

and a node $u \in V$, removing u from the tree $\mathcal{B} := (V, E)$ decomposes $\mathcal{B}$ into $\deg(u)$ *many connected components $\mathcal{B}_i := (V_i, E_i)$. We also say, removing u from $\mathcal{T}$ decomposes $\mathcal{T}$ into* $\deg(u)$ *many components $(\mathcal{T}_i)_{i \in \{1,\dots,\deg(u)\}}$, namely*

$$\mathcal{T}_i = (\mathcal{B}_i, \mu_i \colon T_i' \to \Sigma^d) \quad \textit{with} \quad \mu_i \colon T_i' \cap V \to \Sigma^d;\ u \mapsto \mu(u)$$

A well known property of an optimal Steiner tree is described by the so called bottleneck-Steiner-distances. Let us now recapitulate two important results in Lemma 1.2 and in Theorem 1.3.

Definition 1.19 (bottleneck-Steiner-distances) *Given a set $T \subset \Sigma^d$ of terminals and an arbitrary minimum spanning tree $M = (T, E_M)$ over T. The* bottleneck-Steiner-distance *function (short:* bnsd*-function) is given by*

$$\text{bnsd} \colon T \times T \to \mathbb{N};$$

$$(u, v) \mapsto \begin{cases} \max\{\ \|e\| \mid e \in E_M \textit{ is in the path } u \rightsquigarrow v \textit{ in } M\ \}, & \textit{if } u \neq v \\ 0, & \textit{if } u = v \end{cases}$$

In the above definition we have chosen an arbitrary minimum spanning tree for the definition of the bottleneck-Steiner-distance function. We will now show that the bnsd-function is independent of the choice of M.

Lemma 1.2 *The bottleneck-Steiner-distance function is independent of the choice of the minimum spanning tree.*

Proof: Given two minimum spanning trees $M_1 = (T, E_M^1)$ and $M_2 = (T, E_M^2)$ and the corresponding bottleneck-Steiner-distance functions $\text{bnsd}_1 : T \times T \to \mathbb{N}$ and $\text{bnsd}_2 : T \times T \to \mathbb{N}$. Assume that the functions are not equal, that is there exist $u, v \in T$ such that without loss of generality $\text{bnsd}_1(u, v) > \text{bnsd}_2(u, v)$. Let $e \in E_M^1$ be the edge on the path $u \rightsquigarrow v$ in M_1 with highest cost. Removing e from E_M^1 decomposes M_1 into two connected components C_1 and C_2. Now consider the path $p := u \rightsquigarrow v$ in M_2. Let $\{u', v'\} \in p$ be an edge with $u' \in C_1$ and $v' \in C_2$. Note that such an edge must exist, since $u \neq v$. Adding $\{u', v'\}$ to E_M^1 reconnects C_1 and C_2 again but at a lower cost since $\text{bnsd}_1(u, v) > \text{bnsd}_2(u, v)$, thus showing that the new tree is shorter than M_2 which is a contradiction to the assumption that M_2 was a minimum spanning tree. □

Definition 1.20 *Given a set $T = \{ t_1, t_2, \ldots, t_n \} \subset \Sigma^d$, we call a topology $\mathcal{T} = ((V, E), \mu : T' \to \Sigma^d)$ with $T' = \{ t'_1, t'_2, \ldots, t'_n \}$ and $\mu(t'_i) = t_i$ for all $i \in \{1, \ldots, n\}$ a* topology over T.

Theorem 1.3 (bottleneck-Steiner-distance) *Let $T \subset \Sigma^d$ be a set of terminals, let $ST = ((V, E), \mu)$ be a Steiner minimum tree over T with terminal nodes Z and let (V, E) be in standard form with respect to Z. For an edge $e \in E$ let $\mathcal{C}_1^e$ and $\mathcal{C}_2^e$ denote the two connected components in which ST is decomposed when removing e from ST. Then*

$$\forall e \in E : \forall u \in \text{leaves}(\mathcal{C}_1^e), \forall v \in \text{leaves}(\mathcal{C}_2^e) : \|e\| \leq \text{bnsd}(\mu(u), \mu(v))$$

Proof: Let $M = (Z, E_M)$ be a minimum spanning tree over Z with edge costs $\|\mu(u), \mu(v)\|$ for $\{u, v\} \in E_M$. Assume that there is an edge $e \in E$ violating the claim. Thus there exist nodes $u \in \text{leaves}(\mathcal{C}_1^e)$ and $v \in \text{leaves}(\mathcal{C}_2^e)$ such that $\|e\| > \text{bnsd}(\mu(u), \mu(v))$. Consider now the path $u \rightsquigarrow v$ in M. Since $u \in C_1^e$ and $v \in C_2^e$ there must be an edge (t_1, t_2) with $t_1 \in \text{leaves}(C_1^e)$ and $t_2 \in \text{leaves}(C_2^e)$. Now consider the Steiner tree $ST' = ((V, E \setminus \{e\} \cup \{\{t_1, t_2\}\}), \mu)$. We have

$$\begin{aligned}
\text{cost}\, ST' &= \sum_{e \in E \setminus \{e\} \cup \{\{t_1, t_2\}\}} \|e\| \\
&= \sum_{e \in E} \|e\| - \|e\| + \|\{t_1, t_2\}\| \\
&\overset{\text{def. of bnsd}}{\leq} \text{cost}(ST) - \|e\| + \text{bnsd}(\mu(u), \mu(v)) \\
&\overset{\text{by assumption}}{<} \text{cost}(ST)
\end{aligned}$$

which shows that ST was not a Steiner minimum tree thus proving the theorem. □

In the following we will only consider Steiner trees in standard form as stated in Theorem 1.1 unless stated otherwise.

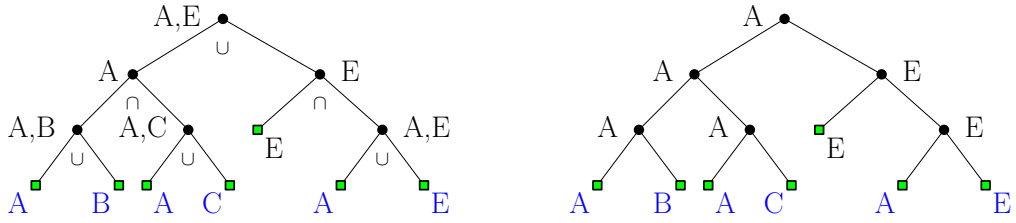


Figure 1.4: An example of Fitch's algorithm on one character terminals (blue characters). The left topology shows the ranges of the inner nodes after the first phase of the algorithm, the right one the state after the second phase when choosing the character A for the root range.

1.3.1 Fitch's Algorithm

Given a topology $\mathcal{T} = ((V, E), \mu_T \colon T' \to \Sigma^d)$ over $T = \{ t_1, t_2, \ldots, t_n \} \subset \Sigma^d$, how can one find a complemental labeling $\overline{\mu} \colon V \setminus T' \to \Sigma^d$ for $\mathcal{T}$? As already mentioned in the introduction, Fitch has shown in [Fit71] how this can be done in $O(n \cdot d \cdot |\Sigma|)$ time. We now sketch Fitch's algorithm (see Figure 1.4 for an example):

Definition 1.21 (Fitch range function) *Given a rooted topology*

$$\mathcal{T} = ((V, E),\ \mu_T \colon T' \to \Sigma^d,\ r)$$

over the terminal set T, we define the Fitch range function $\mu_F[\mathcal{T}] \colon V \to \mathcal{P}(\Sigma)^d$ *of $\mathcal{T}$ as follows: if $u \in T'$, we set $\mu_F[\mathcal{T}](u) := \mu_T(u)$ – otherwise we set*

$$\mu_F[\mathcal{T}](u) := \begin{cases} \mu_F[\mathcal{T}](u_1)_i \cup \mu_F[\mathcal{T}](u_2)_i, & \text{if } \mu_F[\mathcal{T}](u_1)_i \cap \mu_F[\mathcal{T}](u_2)_i = \emptyset \\ \mu_F[\mathcal{T}](u_1)_i \cap \mu_F[\mathcal{T}](u_2)_i, & \text{if } \mu_F[\mathcal{T}](u_1)_i \cap \mu_F[\mathcal{T}](u_2)_i \neq \emptyset \end{cases}$$

for all $i \in \{1, \ldots, d\}$ where u_1 and u_2 are the children of u.

The algorithm starts by picking an arbitrary edge $e \in E$ and constructs the rooted topology $\mathcal{T}_r = (((V \cup \{r\}, E), \mu_T \colon T' \to \Sigma^d, r)$ induced by e. Subsequently, the algorithm works in two phases:

- In the first phase the Fitch range function $\mu_F[\mathcal{T}_r]$ is computed recursively in a bottom up manner starting from the leaves of $\mathcal{T}_r$.
- In the second phase a complemented labeling μ for $\mathcal{T}_r$ is constructed recursively starting at the root node r. Let u be the current node in this traversal. We distinguish between two cases: if $u = r$ or $\mu(\text{parent}(u))_i \notin \mu_F[\mathcal{T}_r](u)_i$ we set $\mu(u)_i := x_i$ for an arbitrary $x_i \in \mu_F[\mathcal{T}_r](u)_i$. If $u \neq r$ and $\mu(\text{parent}(u))_i \in \mu_F[\mathcal{T}_r](u)_i$ we set $\mu(u)_i := \mu(\text{parent}(u))_i$. If u is not a leaf we recurse on the child nodes of u.

The main insight here is the following: Let – without loss of generality – all strings have length one and instead of range points we only consider ranges to simplify the

discussion. Consider now an inner node $u \in V \cup \{r\}$. There must be an optimal Steiner tree in which the labeling of u is chosen in such a way that the cost of its subtrees $\mathcal{T}_1$ and $\mathcal{T}_2$ is optimal. To see this assume otherwise. Then the cost for $\mathcal{T}_1$ and for $\mathcal{T}_2$ is at least by 1 larger than the cost for optimal subtrees. On the other hand, choosing u in an way such that the cost of the edge incident to its parent is minimized saves at most a cost of 1. Let $\mu_F[\mathcal{T}_r](u_1)$ and $\mu_F[\mathcal{T}_r](u_2)$ be the ranges of the two children nodes u_1 and u_2 of u. If $\mu_F[\mathcal{T}_r](u_1) \cap \mu_F[\mathcal{T}_r](u_2) \neq \emptyset$, we can select any letter of this intersection to construct a tree whose cost is the sum of the costs of the two subtrees. Hence $\mu_F[\mathcal{T}_r](u)$ becomes $\mu_F[\mathcal{T}_r](u_1) \cap \mu_F[\mathcal{T}_r](u_2)$ in this case. If $\mu_F[\mathcal{T}_r](u_1) \cap \mu_F[\mathcal{T}_r](u_2) = \emptyset$, the children of u will definitely have different letters. Hence the minimal cost subtree will be attained by one element in $\mu_F[\mathcal{T}_r](u_1) \cup \mu_F[\mathcal{T}_r](u_2)$ and with a cost one more than the sum of the cost of the subtrees.

The running time of Fitch's algorithm is obviously bounded by $O(n \cdot d \cdot |\Sigma|)$ since only a constant number of tree traversal is carried out and in each node of these traversals $O(d)$ basic set operations on sets of size $O(\Sigma)$ are performed.

Definition 1.22 (instantiation of the Fitch range function) *Given a rooted topology $\mathcal{T} = ((V, E), \mu, r)$ over some terminal set T, then we call a complemented labeling $\mu\colon V \to \Sigma^d$ obtained by the second phase of Fitch's algorithm, an* instantiation of $\mu_F[\mathcal{T}]$. *We denote by $\mathcal{L}(\mathcal{T})$ the set of all instantiations of $\mu_F[\mathcal{T}]$.*

1.4 Pruning Algorithm

In the following we assume that we are given an instance $T = \{t_1, t_2, \ldots, t_n\} \subseteq \Sigma^d$ of n strings of length d over the alphabet Σ of the Steiner minimum tree problem in Hamming metric. We will call a topology partial or full always with respect to the terminal set T. Furthermore we call a topology $\mathcal{T}$ optimal if it is optimal with respect to T. We also assume that $n > 3$ to avoid technical difficulties - note that for $n \leq 3$ the problem is trivial to solve.

The pruning algorithm we propose is conceptually quite simple: We start with the set X of all pairwise disjoint partial topologies spanning exactly one terminal in T, namely

$$X := \{ \left((\{v_i\}, \emptyset),\ \mu\colon \{v_i\} \to \{t_i\}; v_i \mapsto t_i, v_i \right) \mid i \in \{1, \ldots, n\} \}$$

Then we consecutively construct larger partial rooted topologies by combining smaller topologies contained in X and add them to X. When all full topologies are constructed we evaluate their cost using Fitch's algorithm and determine among them the one with smallest cost.

For the enumeration process we use Theorem 1.1 and Theorem 1.2 to cut down the search space considerably: Let $ST = ((V, E), \mu)$ be a Steiner minimum tree ST, then Theorem 1.1 allows us to assume without loss of generality that ST is given in standard form. Theorem 1.2 states that there must be a Steiner node (note that the Steiner nodes are exactly the inner nodes of ST, since (V, E) is in standard form) $u \in V$ such

that removing u from ST decomposes ST into three connected components C_1, C_2 and C_3 with $\text{size}(C_i) \leq \lfloor n/2 \rfloor$ for $i \in \{1, \ldots, 3\}$. Now consider the rooted topology ST_r induced from ST by some edge $\{u, v\}$ incident with u – let without loss of generality $v \in C_1$. Note that ST_r is still optimal. Then $ST_r = (C_1 \cdot C_2) \cdot C_3$, that is for ST there is an induced rooted topology ST_r that is composed (using the concatenation operation) by 3 rooted topologies of size not bigger than $\lfloor n/2 \rfloor$. Since ST was an arbitrary Steiner minimum tree in standard form this result is true for all such trees. This results justifies the following modifications, yielding a two phase enumeration algorithm:

Again, the first Phase of the algorithm starts with the set X of rooted partial topologies spanning exactly one terminal as defined above. Then all partial topologies of size i are constructed by concatenating all partial rooted topologies $\mathcal{T}_k, \mathcal{T}_l \in X$ with

- $\text{size}(\mathcal{T}_k) + \text{size}(\mathcal{T}_l) = i$ and
- $\text{span}(\mathcal{T}_k) \cap \text{span}(\mathcal{T}_l) = \emptyset$

and added to X. We continue until all partial topologies up to a size of $\lfloor n/2 \rfloor$ are constructed. In the second phase, for all triples $(\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3) \in \mathcal{P}_3(X)$ with $\text{span}(\mathcal{T}_1) \cup \text{span}(\mathcal{T}_2) \cup \text{span}(\mathcal{T}_3) = T$ and $\text{size}(\mathcal{T}_1) + \text{size}(\mathcal{T}_2) + \text{size}(\mathcal{T}_3) = n$ the full topologies $(\mathcal{T}_1 \cdot \mathcal{T}_2) \cdot \mathcal{T}_3$ are constructed and evaluated using Fitch's algorithm. Then, one with minimal cost is reported.

To cut down the search space even more, in the first phase of the algorithm, before adding the topology $\mathcal{T} := \mathcal{T}_k \cdot \mathcal{T}_l$ to X we run several pruning tests as described in the next section. The idea of these pruning tests is to proof that $\mathcal{T}$ cannot be a subtopology of an optimal topology. If it is so, we can clearly discard $\mathcal{T}$ from X.

Algorithm 1 pruning algorithm

$X := \{((\{v_i\}, \emptyset), \mu\colon \{v_i\} \to \{t_i\}; v_i \mapsto t_i, v_i) \mid i \in \{1, \ldots, n\}\}$
for $i = 2 \ldots \lfloor n/2 \rfloor$ **do**
  for all $\mathcal{T}_k,\ \mathcal{T}_l \in X$ **do**
    if $\text{span}(\mathcal{T}_k) \cap \text{span}(\mathcal{T}_l) = \emptyset$ and
      $\text{size}(\mathcal{T}_k) + \text{size}(\mathcal{T}_l) = i$ **then**
      if $\neg$ prunable($\mathcal{T}_k \cdot \mathcal{T}_l$) **then**
        $X = X \cup \{\mathcal{T}_k \cdot \mathcal{T}_l\}$
      end if
    end if
  end for
end for
for all $\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\} \in \mathcal{P}_3(X)$ **do**
  if $(\text{span}(\mathcal{T}_i))_{i \in \{1,\ldots,3\}}$ are pairwise disjoint **then**
    construct $\mathcal{T} := (\mathcal{T}_1 \cdot \mathcal{T}_2) \cdot \mathcal{T}_3$
    determine $\text{cost}(\mathcal{T})$ using Fitch's algorithm
  end if
end for

For constructing the set $\mathcal{FT}$ of full topologies we sort the non-pruned partial topologies lexicographically: First we sort them by the spanned terminals and then by increasing cost. This allows a fast construction of $\mathcal{FT}$, and guarantees that all full topologies in $\mathcal{FT}$ are counted exactly once. As stated before, a full topology is assembled by three partial topologies. Now assume that you have already built the topology $\mathcal{T}_1 \cdot \mathcal{T}_2$ using $\mathcal{T}_1, \mathcal{T}_2 \in X$. Then we have to concatenate $\mathcal{T}_1 \cdot \mathcal{T}_2$ with all $\mathcal{T}_3 \in X$ such that $\text{span}(\mathcal{T}_1 \cdot \mathcal{T}_2 \cdot \mathcal{T}_3) = T$. Therefore, we perform a binary search on the sorted list for the first topology $\mathcal{T}_3$ such that $\text{span}(\mathcal{T}_3) = T \setminus \text{span}(\mathcal{T}_1 \cdot \mathcal{T}_2)$. Then we iterate over this list creating the full topology for each element until either a topology does not span the correct terminals anymore or $|\mathcal{T}_1 \cdot \mathcal{T}_2| + |\mathcal{T}_3|$ is greater than an upper bound on the cost of an optimal solution, like the length of the minimal full topology found so far or a bound computed by a heuristic.

1.5 Pruning Tests

As mentioned in the previous section, before adding a newly created topology $\mathcal{T} = ((V_\mathcal{T}, E_\mathcal{T}), \mu_\mathcal{T})$ to the set X we test if $\mathcal{T}$ can be a subtopology of a Steiner minimum tree $ST = ((V, E), \mu)$. In this section we will derive several tests of that kind. Recall that we only consider Steiner trees in standard form as stated in Theorem 1.1.

We will use the following observation that helps to simplify many of the subsequent proofs. Consider a fully labeled topology $\mathcal{T} = ((V, E), \mu)$. Due to the properties of the Hamming metric, the cost of $\mathcal{T}$ is given by

$$\begin{aligned}
\text{cost}(\mathcal{T}) &= \sum_{\{u,v\} \in E} \|\mu(u), \mu(v)\| \\
&= \sum_{\{u,v\} \in E} \sum_{i=1}^{d} \|\mu(u)_i, \mu(v)_i\| \\
&= \sum_{i=1}^{d} \sum_{\{u,v\} \in E} \|\mu(u)_i, \mu(v)_i\| \\
&= \sum_{i=1}^{d} \text{cost}[\mu_i](\mathcal{T})
\end{aligned}$$

That is, when determining the cost of a fixed topology we can consider the characters of the associated strings over Σ^d independently.

1.5.1 Edge Replacement Tests

The idea of the pruning tests that we will derive in this section is a generalization of the bottleneck Steiner test that we have seen before in Theorem 1.3. Let $ST = ((V, E), \mu)$ be a Steiner tree. Removing edges from ST, decomposes the tree in several connected components. If we are able to reconnect these components such that we get a cheaper network over the terminal nodes of ST than ST is, we know that ST was not optimal.

Essentially, this is what is done in Theorem 1.3 for one edge $e \in E$: if $\|e\|$ exceeds the corresponding bottleneck Steiner distance we know that removing e from ST splits the tree into two connected components that can be reconnected in a cheaper way along a minimum spanning tree over the terminals of ST. The problem is that for our purposes this method is not applicable for two reasons: The first one is that we are given only partial information about the Steiner tree in form of a subtopology $\mathcal{T}$, that is we have to make statements that are true for all full topologies with these partial topologies as subtopologies. The second problem is, that we are not given a Steiner tree explicitly in the sense that we only know a subtopology over some subset of the terminals. Therefore, we have to argue about all possible lengths that e can have for all Steiner trees with $\mathcal{T}$ as a subtopology. The basic idea is now the following: removing e from a Steiner tree $ST := (\mathcal{T} \cdot \mathcal{R})$ decomposes into two connected components C_1 and C_2. Now we try to find a complemented labeling μ for ST such that $\text{cost}[\mu](e)$ is bigger than the distance of some pair $(c_1, c_2) \in C_1 \times C_2$. If so, we know that ST cannot be a Steiner minimum tree. Since $\mathcal{R}$ is not known in advance in the pruning algorithm we are only allowed to consider labelings μ that are complemented labelings for any $\mathcal{T} \cdot \mathcal{R}$.

Before we state the main results, let us first proof some technical lemmas and let us give some definitions that we will need later on.

Lemma 1.3 *Given a full topology $ST = \mathcal{T} \cdot C$ with $\mathcal{T} \in X$ as constructed in our algorithm. For any two instantiation μ_1 and μ_2 of the $\mu_F[ST]$ of ST we have:*

$$\|\mu_1(\text{root}(\mathcal{T})),\ \mu_1(\text{root}(C))\| = \|\mu_2(\text{root}(\mathcal{T})),\ \mu_2(\text{root}(C))\|$$

Proof: Recall that for all $i \in \{1, \ldots, d\}$

$$\mu_F(\text{root}(ST))_i = \begin{cases} \mu_F(\text{root}(\mathcal{T}))_i \cup \mu_F(\text{root}(C))_i & \text{if } \mu_F(\text{root}(\mathcal{T}))_i \cap \mu_F(\text{root}(C))_i = \emptyset \quad (I) \\ \mu_F(\text{root}(\mathcal{T}))_i \cap \mu_F(\text{root}(C))_i & \text{if } \mu_F(\text{root}(\mathcal{T}))_i \cap \mu_F(\text{root}(C))_i \neq \emptyset \quad (II) \end{cases}$$

Now we have for $i \in \{1, \ldots, d\}$:

$$\|\mu_1(\text{root}(\mathcal{T}))_i, \mu_1(\text{root}(C))_i\| = 1 \ \Leftrightarrow\ \|\mu_2(\text{root}(\mathcal{T}))_i, \mu_2(\text{root}(C))_i\| = 1$$

which can be seen as follows. In case of (I) both distances must be 1 since any element from $\mu_F(\text{root}(ST))_i$ propagated down is contained either in $\mu_F(\text{root}(\mathcal{T}))_i$ or in $\mu_F(\text{root}(C))_i$. In case of (II) both distances must be 0 since any element in $\mu_F(\text{root}(ST))_i$ is contained in $\mu_F(\text{root}(\mathcal{T}))_i$ as well as in $\mu_F(\text{root}(C))_i$. This completes the proof. □

In the next lemma we proof that if you consider a rooted topology $\mathcal{T}$, the Fitch range function value $\mu_F[\mathcal{T}](\text{root}(\mathcal{T}))$ is maximal, that is for any other way of labeling the $\text{root}(\mathcal{T})$ than given in $\mu_F[\mathcal{T}](\text{root}(\mathcal{T}))$, the resulting cost of $\mathcal{T}$ cannot be minimal, no matter how we label the other nodes in $\mathcal{T}$.

Lemma 1.4 *Given a rooted topology $\mathcal{T} = ((V_\mathcal{T}, E_\mathcal{T}), \mu_\mathcal{T}, r)$ over T. Then there is no complemented labeling μ such that*

$$\mu(r) \notin \mu_F[\mathcal{T}](r)$$

Proof: Assume there exists such a complemented labeling μ with $\mu(r) \notin \mu_F[\mathcal{T}](r)$. We proof the claim by induction. If $\mathcal{T}$ has exactly 1 node in the base case, the claim is trivially true. We restrict – without loss generality – the following discussion to the i-th components of the points. For the inductive step consider $\mathcal{T} = \mathcal{T}_1 \cdot \mathcal{T}_2$. According to Fitch's algorithm we distinguish between the following two cases:

- $\mu_F[\mathcal{T}](r)_i = \mu_F[\mathcal{T}_1](\text{root}(\mathcal{T}_1))_i \cup \mu_F[\mathcal{T}_2](\text{root}(\mathcal{T}_2))_i$:
  Since $\mu(r)_i \notin \mu_F[\mathcal{T}](r)_i$ we have:

  $$\mu(r)_i \notin \mu_F[\mathcal{T}_1](\text{root}(\mathcal{T}_1))_i \quad \text{and} \quad \mu(r)_i \notin \mu_F[\mathcal{T}_2](\text{root}(\mathcal{T}_2))_i \qquad (I)$$

  We have

  $$\begin{aligned} \text{cost}[\mu_i](\mathcal{T}) &= \sum_{j\in\{1,2\}} (\|\mu_i(\text{root}(\mathcal{T}_j)), \mu_i(r)\| + \text{cost}[\mu_i](\mathcal{T}_j) \\ &\geq 2 + \sum_{j\in\{1,2\}} \text{cost}_i(\mathcal{T}_j) \\ &= 1 + \text{cost}_i(\mathcal{T}) \;>\; \text{cost}_i(\mathcal{T}) \end{aligned}$$

  Here, the first inequality holds since for all $j \in \{1,2\}$ either $\|\mu_i(\text{root}(\mathcal{T}_j)), \mu_i(r)\| = 1$ or $\text{cost}[\mu_i](\mathcal{T}_j) > \text{cost}_i(\mathcal{T}_j)$ following from the induction hypothesis and the fact that (I) holds. Thus, μ cannot be a complemented labeling for $\mathcal{T}$.

- $\mu_F[\mathcal{T}](r)_i = \mu_F[\mathcal{T}_1](\text{root}(\mathcal{T}_1))_i \cap \mu_F[\mathcal{T}_2](\text{root}(\mathcal{T}_2))_i$:
  Since $\mu(r)_i \notin \mu_F[\mathcal{T}](r)_i$ we know that for at least one $j \in \{1,2\}$:

  $$\mu(r) \notin \mu_F[\mathcal{T}](\text{root}(\mathcal{T}_j))$$

  In this case we have

  $$\begin{aligned} \text{cost}[\mu_i](\mathcal{T}) &= \sum_{j\in\{1,2\}} (\|\mu_i(\text{root}(\mathcal{T}_j)), \mu_i(r)\| + \text{cost}[\mu_i](\mathcal{T}_j)) \\ &\geq 1 + \sum_{j\in\{1,2\}} \text{cost}_i(\mathcal{T}_j) \\ &= 1 + \text{cost}_i(\mathcal{T}) \;>\; \text{cost}_i(\mathcal{T}) \end{aligned}$$

  Let without loss of generality $\mu(r) \notin \mu_F[\mathcal{T}](\text{root}(\mathcal{T}_1))$. Then the first inequality holds because either $\|\mu_i(\text{root}(\mathcal{T}_1)), \mu_i(r)\| = 1$ or $\text{cost}[\mu_i](\mathcal{T}_1) > \text{cost}_i(\mathcal{T}_1)$. Therefore, μ cannot be a complemented labeling for $\mathcal{T}$.

Thus μ cannot be a complemented labeling for $\mathcal{T}$. □

Note that the second phase of Fitch's algorithm can be generalized. Instead of choosing an arbitrary element in a range that we propagate down, we choose the complete range for propagation and obtain the so called *propagated Fitch ranges*. These ranges have useful properties of which we will make use in the following.

Definition 1.23 (propagated Fitch range function) *Given a rooted topology $\mathcal{T} = ((V,E),\mu,r)$ over some terminal set T and Fitch ranges associated with its nodes. Similar to the second phase of Fitch's algorithm we define the* propagated Fitch range

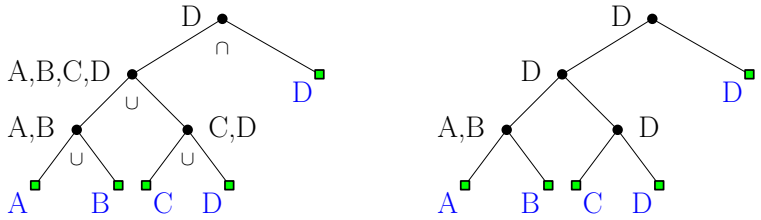


Figure 1.5: An example of the propagated Fitch range function on one character terminals (blue characters). The left topology shows the Fitch range function μ_F, the right one the propagated Fitch range function μ_F^p.

function $\mu_F^p[\mathcal{T}]\colon V \to \mathcal{P}(\Sigma)^d$ *as follows: We traverse* $\mathcal{T}$ *recursively starting from* r. *Consider now the current node* u. *If* u *has a parent* u_f *we set*

$$\mu_F^p[\mathcal{T}](u)_i := \begin{cases} \mu_F[\mathcal{T}](u)_i \cap \mu_F^p[\mathcal{T}](u_f)_i & , \text{if } \mu_F[\mathcal{T}](u)_i \cap \mu_F[\mathcal{T}](u_f)_i \neq \emptyset \\ \mu_F[\mathcal{T}](u)_i & , \text{if } \mu_F[\mathcal{T}](u)_i \cap \mu_F[\mathcal{T}](u_f)_i = \emptyset \end{cases}$$

for all $i \in \{1, \ldots, d\}$. *Then, if* u *is not a leaf we recurse on its children (see figure 1.5 for an example).*

In the next lemma and the corresponding corollary we will answer the following question. If you are given a topology $\mathcal{T}$ over some set of points S and another point p which you want to connect to a Steiner minimum tree with topology $\mathcal{T}$, how can you do this in a cheap way? The obvious answer is to find a complemented labeling for $\mathcal{T}$ such that the distance between any node in $\mathcal{T}$ and p is minimized. We show in the following how the concept of the propagated Fitch range function can be used to do so for a large set of possible complemented labelings.

Lemma 1.5 *Consider a rooted topology* $\mathcal{T} = ((V_\mathcal{T}, E_\mathcal{T}), \mu_T, r)$ *over* T, *some node* $p \in V_\mathcal{T}$, *and some point* $s \in \mu_F^p[\mathcal{T}](p)$. *Then there is an instantiation* μ *of* $\mu_F[\mathcal{T}]$ *such that*

$$\mu(p) = s$$

Proof: Consider the unique path $r \rightsquigarrow p = (r = p_0, p_1, p_2, \ldots, p_m = p)$ in $\mathcal{T}$. We prove the claim by induction over the length l of the path $r \rightsquigarrow p$ in $(V_\mathcal{T}, E_\mathcal{T})$. Let $l = 1$ for the induction base, that is $r = p$. In this case $\mu_F^p[\mathcal{T}](p) = \mu_F[\mathcal{T}](p)$. Thus, according to Fitch's algorithm we can choose $\mu(p) = s$. Now let $l > 1$ in the inductive step. Let p' be the father node of p. We can distinguish between the following two cases:

- If $s_i \in \mu_F^p[\mathcal{T}](p')_i$ by the induction hypothesis there is an instantiation $\mu \in \mathcal{L}(\mathcal{T})$ such that $\mu(p')_i = s_i$. Let $\mathcal{T}'$ be the subtopology of $\mathcal{T}$ rooted at p. Since $s_i \in \mu_F^p[\mathcal{T}](p)_i$ we also have that $s_i \in \mu_F[\mathcal{T}](p)_i$ and thus $s_i \in \mu_F[\mathcal{T}'](p)_i$. Thus, according to Fitch's algorithm we can choose $\mu(p)_i = s_i$, still allowing an optimal labeling of $\mathcal{T}'$. Therefore, there is an optimal labeling for $\mathcal{T}$ with $\mu(p)_i = s_i$.

- Now let us – without loss of generality – restrict our discussion to the i-th component of the range points such that $s_i \notin \mu_F^p[\mathcal{T}](p')_i$. For some $x_i \in \mu_F^p[\mathcal{T}](p')_i$ we

set $\mu(p')_i = x_i$. We know that $x_i \notin \mu_F[\mathcal{T}](p)_i$ for the following reason: according to the definition of the propagated Fitch range function we have the following two cases:

1. $\mu_F^p[\mathcal{T}](p)_i = \mu_F[\mathcal{T}](p)_i \cap \mu_F^p[\mathcal{T}](p')_i$
2. $\mu_F^p[\mathcal{T}](p)_i = \mu_F[\mathcal{T}](p)_i$

Since $s_i \in \mu_F^p[\mathcal{T}](p)_i$ we also have: $s_i \in \mu_F[\mathcal{T}](p)_i$ which excludes the first of these two cases since $s_i \notin \mu_F^p[\mathcal{T}](p')_i$ by assumption. Thus the second case holds which means that $\mu_F^p[\mathcal{T}](p')_i \cap \mu_F[\mathcal{T}](p)_i = \emptyset$. Since $x_i \in \mu_F^p[\mathcal{T}](p')_i$, $x_i \notin \mu_F[\mathcal{T}](p)_i$ which allows us to choose an arbitrary element in $\mu_F[\mathcal{T}](p)_i$ namely s_i. Therefore, there is an optimal labeling for $\mathcal{T}$ with $\mu(p)_i = s_i$.

Which completes the proof. □

Let us now extend Lemma 1.5 to labelings that cannot be constructed using Fitch's algorithm:

Corollary 1.1 *Given a non-rooted topology $\mathcal{T} = ((V_\mathcal{T}, E_\mathcal{T}), \mu_\mathcal{T})$ over T and $p \in V_\mathcal{T}$, then for all $x \in \Sigma^d$ with*

$$x_i \in \bigcup_{e \in E_\mathcal{T}} \mu_F^p[\mathcal{T}_e](p)_i$$

where $\mathcal{T}_e$ denotes the topology induced by the edge e, there is a complemented labeling μ of $\mathcal{T}$ such that $\mu(p) = x$.

Proof: Given d complemented labelings $(\overline{\mu}_j)_{j \in \{1,\ldots,d\}}$ for $\mathcal{T}$, consider the labeling μ with $\mu(u)_j := \overline{\mu}_j(u)_j$ for all $u \in V_\mathcal{T}$. Then

$$\begin{aligned} \text{cost}[\mu](\mathcal{T}) &= \sum_{i=1}^{d} \text{cost}[\mu_i](\mathcal{T}) \\ &= \sum_{i=1}^{d} \text{cost}[(\overline{\mu}_i)_i](\mathcal{T}) \quad = \quad \text{cost}(\mathcal{T}) \end{aligned}$$

where the last equality follows from the following fact: consider two complemented labelings μ' and μ'' for $\mathcal{T}$. Then for any $j \in \{1, \ldots, d\}$, $\text{cost}[\mu'_j](\mathcal{T}) = \text{cost}[\mu''_j](\mathcal{T})$ since otherwise you would be able to find a cheaper labeling than μ' respectively than μ'' which would contradict the complementarity of μ' respectively of μ''. Since Lemma 1.5 directly implies that for each $x_i \in \bigcup_{e \in E_\mathcal{T}} \mu_F^p[\mathcal{T}_e](p)_i$ there is a complemented labeling $\overline{\mu}_i$ with $x_i \in \overline{\mu}_i(p)_i$ the claim follows. □

Note that the complemented labeling μ of $\mathcal{T}$ in corollary 1.1 does not have to be a labeling that can be constructed by Fitch's algorithm but on the other hand for each instantiation $\mu' \in \mathcal{L}(\mathcal{T})$ there is such a labeling μ with $\mu'(p) = \mu(p)$. Thus the corollary usually allows us to construct for the node p much bigger ranges than it would be possible, if we would only make use of Fitch's algorithm and Lemma 1.5.

Now let us examine the following question. Given a partial topology $\mathcal{T}$, how can we decide for an edge e in $\mathcal{T}$ if e can be part of an optimal full topology, that is we want

to find provable properties of $\mathcal{T}$ that allow us to draw conclusions on the lengths of e in instantiations of a full topology $\mathcal{T} \cdot \mathcal{R}$. Therefore, we introduce several modifications of Fitch's algorithm that allow us to compute lower bounds on the length of e in a complemented labeling for $\mathcal{T} \cdot \mathcal{R}$.

The first algorithm under consideration is the so called *modified Fitch algorithm*. The idea behind this algorithm is the following (assume for a moment that $d = 1$): if you consider the edge $e = \{u, v\}$ with $v = \text{parent}(u)$ in a topology $\mathcal{T}$ and you want to find a complemented Fitch labeling μ for $\mathcal{T}$ such that the length of e becomes 1 then clearly you have to set $\mu(v)_1 \neq \mu(u)_1$. But this means that you have to choose the rest of the labeling in such a way that it is possible to choose for v some $\mu(v)_1 \notin \mu_F[\mathcal{T}](u)_1$ (considering labelings constructible by Fitch's algorithm) without violating the optimality of μ. The idea is now to modify Fitch's algorithm in such a way that – whenever it can – it chooses an element not contained in $\mu_F[\mathcal{T}](u)_1$. More generally we define

Definition 1.24 (modified Fitch algorithm) *We are given a rooted topology $\mathcal{T} = ((V_\mathcal{T}, E_\mathcal{T}), r_\mathcal{T})$ over T and a family of sets $(A_j)_{j\in\{1,\dots,d\}} \subseteq \mathcal{P}(\Sigma)^d$. We modify Fitch's algorithm by adding the following rule: if in the second phase of Fitch's algorithm we have to choose an arbitrary element from $\mu_F[\mathcal{T}_r](u)_i$ for the current node u, we first check if $\mu_F[\mathcal{T}_r](u)_i \setminus A_i$ is non-empty. In this case we choose an arbitrary element from $\mu_F[\mathcal{T}_r](u)_i \setminus A_i$ instead of $\mu_F[\mathcal{T}_r](u)_i$. We call the algorithm the* modified Fitch algorithm. *We denote by $\mathcal{L}_A(\mathcal{T})$ the set of labelings that can be computed by this algorithm.*

For the modified Fitch algorithm the goal was to find a complemented labeling for a topology $\mathcal{T}$. But in our case $\mathcal{T}$ is only a partial topology that is used to build a full topology $\mathcal{T} \cdot \mathcal{R}$ for some other partial topology $\mathcal{R}$ that we do not know in advance. To be able to also cope with this case we introduce the *restricted Fitch algorithm*. Assuming that we do not know anything about the topology $\mathcal{R}$ we have to assume that the Fitch ranges of the root of $\mathcal{R}$ are given in such a way that the resulting lower bounds on the length of e are as weak as possible. It turns out that this is the case if the Fitch range is a subset of $\mu_F[\mathcal{T}](u)_1$. So we have to change the modified Fitch algorithm in such a way that it can handle this situation, leading to

Definition 1.25 (restricted Fitch algorithm) *We are given a rooted topology $\mathcal{T} = ((V_\mathcal{T}, E_\mathcal{T}), r_\mathcal{T})$ over T, a family of sets $(A_j)_{j\in\{1,\dots,d\}} \subseteq \mathcal{P}(\Sigma)^d$ and a family of elements in Σ, namely $(b_j)_{j\in\{1,\dots,d\}} \subseteq \Sigma^d$. We now alter the modified Fitch algorithm by adding the following rule: if the current node u is the root node and if $b_i \in \mu_F[\mathcal{T}](r)_i$ we set $\mu(r)_i := b_i$ otherwise we proceed as in the modified Fitch algorithm. We call the algorithm the* restricted Fitch algorithm. *We denote by $\mathcal{L}^b_A(\mathcal{T})$ the set of labelings that can be computed by this algorithm (see figure 1.6 for an example).*

Let us now prove an important property of the labelings computed by the modified Fitch algorithm. Lemma 1.6 states that all instantiations created by this algorithm are essentially the same if considering the length of a given edge and if you choose the appropriate parameters.

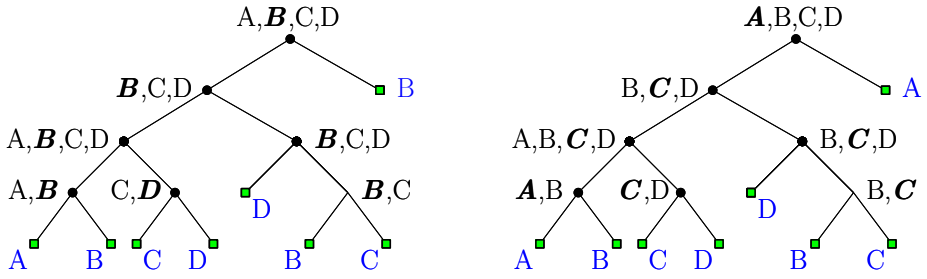


Figure 1.6: An example of the restricted Fitch algorithm on one character terminals (blue characters). The letters at the nodes are the Fitch ranges of the topology. The example shows two possible labelings (bold letters) of the topology that were computed by the restricted Fitch algorithm. In both examples we have $A_1 := \{A, B\}$. In the left topology we have set $b_1 := B$ and in the right one $b_1 := A$.

Lemma 1.6 *Given a rooted topology $\mathcal{T} = ((V_\mathcal{T}, E_\mathcal{T}), \mu_\mathcal{T}, r_\mathcal{T})$ over T and an edge $e = \{u, v\}$ where v is the father of u, we have for all $i \in \{1, \ldots, d\}$: if*

$$\exists\, \overline{\mu} \in \mathcal{L}_A(\mathcal{T}) : \|\overline{\mu}(u)_i,\ \overline{\mu}(v)_i\| = 1$$

then

$$\forall\, \overline{\mu} \in \mathcal{L}_A(\mathcal{T}) : \|\overline{\mu}(u)_i,\ \overline{\mu}(v)_i\| = 1$$

where $A_j := \mu_F[\mathcal{T}](u)_j$

Proof: Let us assume that the premise holds, that is $\exists\, \overline{\mu} \in \mathcal{L}_A(\mathcal{T}) : \|\overline{\mu}(u)_i, \overline{\mu}(v)_i\| = 1$ and let u' be the sibling of u. Then $\mu_F[\mathcal{T}](v)_i = \mu_F[\mathcal{T}](u)_i \cup \mu_F[\mathcal{T}](u')_i$ because otherwise we would have:

$$\begin{aligned} \forall\, \mu \in \mathcal{L}(\mathcal{T}) : \quad & \mu(v)_i \in \mu_F[\mathcal{T}](u)_i \ \wedge\ \mu(v)_i \in \mu_F[\mathcal{T}](u')_i \\ \Rightarrow\ & \mu(u)_i = \mu(u')_i = \mu(v)_i \\ \Rightarrow\ & \|\mu(u)_i,\ \mu(v)_i\| = 0 \end{aligned}$$

which would yield a contradiction since $\mathcal{L}_A(\mathcal{T}) \subseteq \mathcal{L}(\mathcal{T})$ and therefore,

$$\forall\, \mu \in \mathcal{L}_A(\mathcal{T}) : \|\mu(u)_i,\ \mu(v)_i\| = 0$$

We prove the claim now by induction over the length of the path $v \rightsquigarrow r$. Assume that in the base case $|v \rightsquigarrow r_\mathcal{T}| = 1$, that is if $v = r_\mathcal{T}$. As just argued, $\mu_F[\mathcal{T}](r_\mathcal{T})_i = \mu_F[\mathcal{T}](u)_i \cup \mu_F[\mathcal{T}](u')_i$. Thus with $A_i := \mu_F[\mathcal{T}](u)_i$ the modified Fitch algorithm chooses for $r_\mathcal{T}$ an arbitrary element from

$$\begin{aligned} \mu_F[\mathcal{T}](r_\mathcal{T})_i &= \mu_F[\mathcal{T}](u)_i \cup \mu_F[\mathcal{T}](u')_i \setminus A_i \\ &= \mu_F[\mathcal{T}](u')_i \end{aligned}$$

Thus

$$\forall\, \overline{\mu} \in \mathcal{L}_A(\mathcal{T}) : \|\overline{\mu}(u)_i,\ \overline{\mu}(v)_i\| = 1$$

Now let us consider the inductive step. Let $(v \rightsquigarrow r_{\mathcal{T}}) := (v = v_1, v_2, \ldots, v_k = r_{\mathcal{T}})$. We know by Fitch's algorithm that their are two possibilities for a node $v_l \in (v_1 \rightsquigarrow v_k)$: either $\overline{\mu}(v_{l+1})_i = \overline{\mu}(v_l)_i$ or $\overline{\mu}(v_{l+1})_i \notin \mu_F[\mathcal{T}](v_l)_i$. We now distinguish two cases: in the first one let $v_{l+1} \in (v_1 \rightsquigarrow v_k)$ be the first node on the path for which $\overline{\mu}(v_{l+1})_i \notin \mu_F[\mathcal{T}](v_l)_i$, that is $\overline{\mu}(v_{l+1})_i \neq \overline{\mu}(v_l)_i$. By induction hypothesis we now know that

$$\forall\, \overline{\mu} \in \mathcal{L}_A(\mathcal{T}) : \|\overline{\mu}(v_{l+1})_i,\ \overline{\mu}(v_l)_i\| = 1$$

Now consider any call of the modified Fitch algorithm when reaching node v_l. Note that for all nodes w on the path $v_2 \rightsquigarrow v_l$ we have $\overline{\mu}(v)_i \in \mu_F[\mathcal{T}](w)_i$. Thus following the path down to v_1 the algorithm will always choose some element not contained in $\mu_F[\mathcal{T}](u)_i$. Thus any call produces a labeling $\overline{\mu}$ with

$$\|\overline{\mu}(u)_i,\ \overline{\mu}(v)_i\| = 1$$

which proves the claim. Now consider the case in which no such v_{l+1} exists. Then for all nodes w on the path $v_2 \rightsquigarrow r_{\mathcal{T}}$ we have $\overline{\mu}(v)_i \in \mu_F[\mathcal{T}](w)_i$. Following the same argument as just given above the claim also holds in this case which completes the proof. □

The following lemma formulates a criterion such that if a partial topology $\mathcal{T}$ satisfies this criterion then we know that a given edge in $\mathcal{T}$ has at least a certain length in a (not any!) complemented labeling for any topology $\mathcal{T} \cdot \mathcal{R}$.

Lemma 1.7 *Given a rooted topology $\mathcal{T} = ((V_{\mathcal{T}}, E_{\mathcal{T}}), \mu_{\mathcal{T}}, r_{\mathcal{T}})$ over some $T' \subseteq T$ and an edge $e = \{u, v\}$ where v is the father of u and a rooted topology $\mathcal{R} = ((V_{\mathcal{R}}, E_{\mathcal{R}}), \mu_{\mathcal{R}}, r_{\mathcal{R}})$ over $T \setminus T'$, then we have $\forall\, i \in \{1, \ldots, d\}$: if*

$$\begin{aligned} &\forall\, b \in \Sigma^d \quad \textit{with} \quad b_j \in \mu_F[\mathcal{T}](u)_j \cap \mu_F[\mathcal{T}](r_{\mathcal{T}})_j : \\ &\exists\, \overline{\mu} \in \mathcal{L}_A^b(\mathcal{T}) : \|\overline{\mu}(u)_i,\ \overline{\mu}(v)_i\| = 1 \end{aligned}$$

where $A_j := \mu_F[\mathcal{T}](u)_j$, then

$$\exists\, \mu \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R}) : \|\mu(u)_i,\ \mu(v)_i\| = 1$$

Proof: Let us first consider the case

$$\mu_F[\mathcal{T} \cdot \mathcal{R}](\mathrm{root}(\mathcal{T} \cdot \mathcal{R}))_i = \mu_F[\mathcal{T}](r_{\mathcal{T}})_i \cup \mu_F[\mathcal{T}](r_{\mathcal{R}})_i$$

Consider any $(b_j)_{j \in \{1,\ldots,d\}}$ and $\overline{\mu} \in \mathcal{L}_A^b(()\mathcal{T})$ such that $\|\overline{\mu}(u)_i, \overline{\mu}(v)_i\| = 1$. Then with $\mu := \overline{\mu} \odot \mu'_R \odot \mu^r$ where μ'_R is an arbitrary Fitch labeling in $\mathcal{L}(\mathcal{R})$ and where

$$\mu^r : \{\mathrm{root}(\mathcal{T} \cdot \mathcal{R})\} \to \Sigma^d; u \mapsto \overline{\mu}(r_{\mathcal{T}})$$

we have $\mu \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R}) : \|\mu(u)_i, \mu(v)_i\| = 1$ which can be seen as follows: consider a call of Fitch's algorithm on $\mathcal{T} \cdot \mathcal{R}$ constructing μ. Since

$$\mu_F[\mathcal{T} \cdot \mathcal{R}](\mathrm{root}(\mathcal{T} \cdot \mathcal{R}))_i = \mu_F[\mathcal{T}](r_{\mathcal{T}})_i \cup \mu_F[\mathcal{T}](r_{\mathcal{R}})_i$$

we know that $\overline{\mu}(r_{\mathcal{T}})_i \in \mu_F[\mathcal{T} \cdot \mathcal{R}](\mathrm{root}(\mathcal{T} \cdot \mathcal{R}))_i$, because $\overline{\mu}(r_{\mathcal{T}})_i \in \mu_F[\mathcal{T}](r_{\mathcal{T}})_i$. By the definition of Fitch's algorithm we can choose $\mu(\mathrm{root}(\mathcal{T} \cdot \mathcal{R}))_i := \overline{\mu}(r_{\mathcal{T}})_i$. Since we have

assumed that $\mu_F[\mathcal{T}](r_\mathcal{T})_i \cap \mu_F[\mathcal{T}](r_\mathcal{R})_i = \emptyset$ we know that $\overline{\mu}(r_\mathcal{T})_i \notin \mu_F[\mathcal{R}](r_\mathcal{R})_i$ thus we can set $\mu(x)_i := \mu'_R(x)_i$ for all $x \in V_\mathcal{R}$. According to Fitch's algorithm we now have to set $\mu(r_\mathcal{T})_i := \overline{\mu}(r_\mathcal{T})_i$. But because $\overline{\mu}$ is also contained in $\mathcal{L}(\mathcal{T})$ we can also set $\mu(x)_i := \overline{\mu}(x)_i$ for all other nodes $x \in V_\mathcal{T}$, completing the first case.
So let us assume now that

$$\mu_F[\mathcal{T} \cdot \mathcal{R}](\text{root}(\mathcal{T} \cdot \mathcal{R}))_i = \mu_F[\mathcal{T}](r_\mathcal{T})_i \cap \mu_F[\mathcal{T}](r_\mathcal{R})_i \qquad (I)$$

In the following we will assume for sake of simplicity that $d = 1$. Recall that we can do so without loss of generality since we consider a fixed topology $\mathcal{T}$. We distinguish now between the two cases whether $\mu_F[\mathcal{T}](r_\mathcal{T})_i \cap \mu_F[\mathcal{T}](u)_i = \emptyset$ or not. Let us first assume that the intersection is empty. Since (I) holds we also know that

$$\mu_F[\mathcal{T} \cdot \mathcal{R}](\text{root}(\mathcal{T} \cdot \mathcal{R}))_i \subseteq \mu_F[\mathcal{T}](r_\mathcal{T})_i$$

and therefore,

$$\mu_F[\mathcal{T} \cdot \mathcal{R}](\text{root}(\mathcal{T} \cdot \mathcal{R}))_i \cap \mu_F[\mathcal{T}](u)_i = \emptyset$$

Thus, for any $\overline{\mu} \in \mathcal{L}^b_A(\mathcal{T})$ we have $\overline{\mu} \in \mathcal{L}_A(\mathcal{T})$. On the other hand Lemma 1.6 implies that

$$\forall\, \overline{\mu} \in \mathcal{L}_A(\mathcal{T}) : \|\overline{\mu}(u)_i, \overline{\mu}(v)_i\| = 1$$

which in turn implies that for any $\mu' \in \mathcal{L}_A(\mathcal{T} \cdot \mathcal{R})$ we also have $\|\mu'(u)_i, \mu'(v)_i\| = 1$. But since

$$\mu_F[\mathcal{T} \cdot \mathcal{R}](\text{root}(\mathcal{T} \cdot \mathcal{R}))_i \cap \mu_F[\mathcal{T}](u)_i = \emptyset$$

we also know that $\mu' \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R})$ showing the claim.
Now consider the second case in which

$$\mu_F[\mathcal{T}](r_\mathcal{T})_i \cap \mu_F[\mathcal{T}](u)_i \neq \emptyset$$

We again distinguish now between two cases: if $\mu_F[\mathcal{T}](r_\mathcal{T})_i \subseteq \mu_F[\mathcal{T}](u)_i$ we are in the same situation as above, since for any $\overline{\mu} \in \mathcal{L}^b_A(\mathcal{T})$ we also have $\overline{\mu} \in \mathcal{L}_A(\mathcal{T})$ in that case. So let us assume that $\mu_F[\mathcal{T}](r_\mathcal{T})_i$ contains – but not only – elements from $\mu_F[\mathcal{T}](u)_i$. If $\mu_F[\mathcal{T} \cdot \mathcal{R}](\text{root}(\mathcal{T} \cdot \mathcal{R}))_i \subseteq \mu_F[\mathcal{T}](u)_i$ we are done by choosing an arbitrary element for $\mu(\text{root}(\mathcal{T} \cdot \mathcal{R}))_i$. Thus $\mu(r_\mathcal{T})_i = \mu(\text{root}(\mathcal{T} \cdot \mathcal{R}))_i \in \mu_F[\mathcal{T}](u)_i$ and by the premise follows the claim. The only case left is that $\mu_F[\text{root}(\mathcal{T} \cdot \mathcal{R})]_i \cap \mu_F[\mathcal{T}](u)_i = \emptyset$. We prove this case by contradiction. Assume for this that the claim does not hold, that is

$$\nexists\, \mu \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R}) : \|\mu(u)_i,\ \mu(v)_i\| = 1$$

or equivalently

$$\forall\, \mu \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R}) : \|\mu(u)_i,\ \mu(v)_i\| = 0$$

since $\mathcal{L}_A(\mathcal{T} \cdot \mathcal{R}) \subseteq \mathcal{L}(\mathcal{T} \cdot \mathcal{R})$, in particular we have

$$\forall\, \mu \in \mathcal{L}_A(\mathcal{T} \cdot \mathcal{R}) : \|\mu(u)_i,\ \mu(v)_i\| = 0 \qquad (II)$$

So let us consider now any $\mu \in \mathcal{L}_A(\mathcal{T} \cdot \mathcal{R})$. We know that $\mu(\text{root}(\mathcal{T} \cdot \mathcal{R}))_i \notin \mu_F[\mathcal{T}](u)_i$ and thus $\mu(r_\mathcal{T})_i \notin \mu_F[\mathcal{T}](u)_i$. Consider now the path $(v \rightsquigarrow r_\mathcal{T}) = (v = v_1, \ldots, v_k = r_\mathcal{T})$. Since we have assumed (II), $\mu(v)_i \in \mu_F[\mathcal{T}](u)_i$. Now there must exist an $v_l \in (v \rightsquigarrow r_\mathcal{T})$

with $\mu(v_l)_i \neq \mu(v)_i$ since otherwise, the premise of the claim would be violated with $b_j := \mu(v)_i$. But this means that $\mu(v_l)_i \notin \mu_F[\mathcal{T}](v_{l-1})_i$. Since $\mu \in \mathcal{L}_A(\mathcal{T} \cdot \mathcal{R})$ we know that $\mu_F[\mathcal{T}](v_{l-1})_i \subseteq \mu_F[\mathcal{T}](u)$, because otherwise $\mu(v_{l-1})_i \in A_i$ would not have been chosen by the algorithm. Let us denote by $\mathcal{T}'$ the subtopology of $\mathcal{T}$ rooted at v_{l-1}. From the existence of μ follows that

$$\exists\, \mu' \in \mathcal{L}_A(\mathcal{T}') \colon \|\mu'(u)_i,\ \mu'(v)_i\| = 0$$

Lemma 1.6 implies

$$\forall\, \mu' \in \mathcal{L}_A(\mathcal{T}') \colon \|\mu'(u)_i,\ \mu'(v)_i\| = 0$$

Now consider any call of the restricted Fitch algorithm constructing $\overline{\mu} \in \mathcal{L}_A^b(\mathcal{T})$. When reaching node $v_{l-1} \neq r_{\mathcal{T}}$ we know from the existence of μ' that $\|\overline{\mu}(u)_i,\ \overline{\mu}(v)_i\| = 0$ which is a contradiction to the premise of the lemma. Thus the assumption

$$\nexists\, \mu \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R}) : \|\mu(u)_i,\ \mu(v)_i\| = 1$$

that we have made must be must be wrong, showing the claim in this last case. This completes the proof. □

Corollary 1.2 *Given a rooted topology $\mathcal{T} = ((V_{\mathcal{T}}, E_{\mathcal{T}}), \mu_{\mathcal{T}}, r_{\mathcal{T}})$ over some $T' \subseteq T$ and an edge $e = \{u, v\}$ where v is the father of u and a rooted topology $\mathcal{R} = ((V_{\mathcal{R}}, E_{\mathcal{R}}), \mu_{\mathcal{R}}, r_{\mathcal{R}})$ over $T \setminus T'$, then we have $\forall\, i \in \{1, \ldots, d\}$: if*

$$\forall\, b \in \Sigma^d \quad \textit{with} \quad b_j \in \mu_F[\mathcal{T}](u)_j \cap \mu_F[\mathcal{T}](r_{\mathcal{T}})_j \cap \bigcup_{t \in \text{leaves}(\mathcal{R})} \mu_{\mathcal{R}}(t) \colon$$

$$\exists\, \overline{\mu} \in \mathcal{L}_A^b(\mathcal{T}) : \|\overline{\mu}(u)_i,\ \overline{\mu}(v)_i\| = 1$$

where $A_j := \mu_F[\mathcal{T}](u)_j$, then

$$\exists\, \mu \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R}) : \|\mu(u)_i,\ \mu(v)_i\| = 1$$

Proof: Follows directly from the proof of Lemma 1.7: in the case

$$\mu_F[\mathcal{T} \cdot \mathcal{R}](\text{root}(\mathcal{T} \cdot \mathcal{R}))_i = \mu_F[\mathcal{T}](r_{\mathcal{T}})_i \cup \mu_F[\mathcal{R}](r_{\mathcal{R}})_i$$

exactly the same argumentation stays valid and in the case

$$\mu_F[\mathcal{T} \cdot \mathcal{R}](\text{root}(\mathcal{T} \cdot \mathcal{R}))_i = \mu_F[\mathcal{T}](r_{\mathcal{T}})_i \cap \mu_F[\mathcal{R}](r_{\mathcal{R}})_i$$

the claim is also valid since

$$\nexists\, x \in \mu_F[\mathcal{R}](r_{\mathcal{R}})_i \colon x \notin \bigcup_{t \in \text{leaves}(\mathcal{R})} \mu_{\mathcal{R}}(t)$$

□

But what is the complexity of checking whether the premise in Lemma 1.7 is satisfied for a given $i \in \{1, \ldots, d\}$? Once we have computed the Fitch range function for $\mathcal{T}$ using

$O(|V_{\mathcal{T}}|)$ basic set operations, we have to perform the second phase of the restricted Fitch algorithm for each element

$$b_j \in B := \mu_F[\mathcal{T}](u)_j \cap \mu_F[\mathcal{T}](r_{\mathcal{T}})_j \cap \bigcup_{t \in \text{leaves}(\mathcal{R})} \mu_{\mathcal{R}}(t)$$

Note that we have to perform this second phase just for the path from the root of the topology to the edge e, but since the depth of the topology can be $O(|V_{\mathcal{T}}|)$ this phase involves $O(|V_{\mathcal{T}}|)$ basic set operations. Thus in the worst case the complexity is given by $O(|V_{\mathcal{T}}| \cdot |\Sigma|)$ basic set operations. But we can easily improve that running time. The idea for this is as follows: instead of traversing $\mathcal{T}$ for each b_j separately, we do it only once, bookkeeping which node labels have to be set to b_j in this traversal just because the algorithm could have chosen $\overline{\mu}(r_{\mathcal{T}})_i := b_j$. For doing so we have to pass down a subset S of B (initially we set $S := B$) along any path of $\mathcal{T}$ in the second phase of the algorithm. Before the algorithm decides which element to pick it first checks whether S is empty or not. If it is empty the algorithm proceeds as the usual, if not S is set to $S \cap \mu_F[\mathcal{T}](w)_i$ where w is the child of the current node in the traversal. If the algorithm reaches node v and the set S is non-empty we have to check if $S \cap \mu_F[\mathcal{T}](u)_i$ is non-empty. If yes, then we know that the premise given in Lemma 1.7 is not satisfied. Let us call this algorithm the *set-restricted Fitch algorithm*. The running time is obviously bounded by $O(|V_{\mathcal{T}}|)$ basic set operations. We associate with this algorithm the indicator function

$$\mathcal{SRF}(\mathcal{T},\ \{u,v\},\ i)\colon V_{\mathcal{T}} \times E_{\mathcal{T}} \times \{1,\ldots,d\} \to \{0,1\}$$

which maps to 1 if and only if the set-restricted Fitch algorithm shows that

$$\begin{aligned} &\forall\ (b_j)_{j\in\{1,\ldots,d\}} \quad \text{with} \quad b_j \in \mu_F[\mathcal{T}](u)_j \cap \mu_F[\mathcal{T}](r_{\mathcal{T}})_j\colon \\ &\exists\ \overline{\mu} \in \mathcal{L}_A^b(\mathcal{T}) : \|\overline{\mu}(u)_i,\ \overline{\mu}(v)_i\| = 1 \end{aligned}$$

where $A_j := \mu_F[\mathcal{T}](u)_j$

Note that the function $\mathcal{SRF}$ is well defined since Lemma 1.6 implies that in the above expression

$$\begin{aligned} & \exists\ \overline{\mu} \in \mathcal{L}_A^b(cT) : \|\overline{\mu}(u)_i,\ \overline{\mu}(v)_i\| = 1 \\ \Rightarrow \quad & \forall\ \overline{\mu} \in \mathcal{L}_A^b(cT) : \|\overline{\mu}(u)_i,\ \overline{\mu}(v)_i\| = 1 \end{aligned}$$

Corollary 1.3 *Given a rooted topology $\mathcal{T} = ((V_{\mathcal{T}}, E_{\mathcal{T}}), \mu_{\mathcal{T}}, r_{\mathcal{T}})$ over some $T' \subseteq T$ and an edge $e = \{u, v\}$ where v is the father of u and a rooted topology $\mathcal{R} = ((V_{\mathcal{R}}, E_{\mathcal{R}}), \mu_{\mathcal{R}}, r_{\mathcal{R}})$ over $T \setminus T'$ and an $i \in \{1, \ldots, d\}$, then*

$$\mathcal{SRF}(\mathcal{T},\ e,\ i) = 1 \quad \Rightarrow \quad \exists\ \mu \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R}) : \|\mu(u)_i,\ \mu(v)_i\| = 1$$

Definition 1.26 *Let $\mathcal{T} = ((V_{\mathcal{T}}, E_{\mathcal{T}}), \mu_{\mathcal{T}}, r)$ be a rooted topology over T. For an edge $e = \{u, v\} \in E_{\mathcal{T}}$ where v is the father of u we define*

$$\|e\|_m \quad := \quad |\{\ i \in \{1,\ldots,d\} \mid \mathcal{SRF}(\mathcal{T},\ e,\ i) = 1\ \}|$$

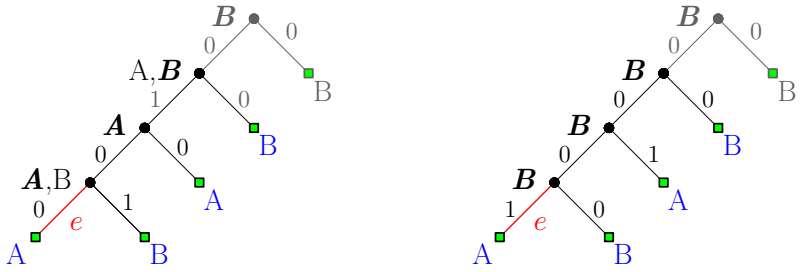


Figure 1.7: An example of the non-optimality of $\|\cdot\|_m$ bound. Consider the topologies $\mathcal{T}$ (black edges), $\mathcal{R}$ (grey edges) and the edge $e := \{v, u\}$ where $v = \text{parent}(u)$. Since $\mu_F[\mathcal{T}](\text{parent}(v)) = \{A\}$, for all instantiations $\mu \in \mathcal{L}(\mathcal{T}) \cup \mathcal{L}(\mathcal{T} \cdot \mathcal{R})$, $\text{cost}[\mu](e) = 0$ and in particular $\|e\|_m = 0$. But on the other hand the right figure shows a non-Fitch optimal labeling μ' for $\mathcal{T} \cdot \mathcal{R}$ with $\text{cost}[\mu'](e) = 1$.

Corollary 1.4 *Let $\mathcal{T} = ((V_\mathcal{T}, E_\mathcal{T}), \mu_\mathcal{T})$ be a rooted topology over $T' \subsetneq T$, an edge $\{u, v\} \in E_\mathcal{T}$ where v is the father of u and let $\mathcal{R} = ((V_\mathcal{R}, E_\mathcal{R}), \mu_\mathcal{R})$ be a rooted topology over $T \setminus T'$. Then*

$$\exists\, \mu \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R}) : \|\mu(u),\ \mu(v)\| \geq \|u,\ v\|_m$$

Note that Corollary 1.4 makes a statement only about labelings that can be computed using Fitch's algorithm. Figure 1.7 shows an example which demonstrates that there is still room for improvement as there can exist an optimal labeling not contained in $\mathcal{L}(\mathcal{T} \cdot \mathcal{R})$ that produces a long edge e even if it is not detected by our approach.

Now that we know how to predict a long/heavy edge in a Steiner minimum tree with full topology $\mathcal{T} \cdot \mathcal{R}$, we can derive properties that tell us when we are allowed to prune a partial topology as it cannot be part of an optimal one. The idea of Theorem 1.4 and Theorem 1.5 is as follows. If we decompose a topology $\mathcal{T} \cdot \mathcal{R}$ by deleting an edge e in $\mathcal{T}$ we test if it is possible to re-connect the resulting connected components by adding another edge e'. If the cost of e' is now smaller than the cost of e we that $\mathcal{T} \cdot \mathcal{R}$ cannot be optimal and thus $\mathcal{T}$ can be pruned. The only difference between the two theorems is that they consider a different sets of possible edges for the re-connection phase.

Theorem 1.4 (heavy edge property (I)) *Let $\mathcal{T} = ((V_\mathcal{T}, E_\mathcal{T}), \mu_\mathcal{T})$ be a rooted topology over $T' \subsetneq T$, an edge $e := \{u, v\} \in E_\mathcal{T}$ where v is the father of u and let $\mathcal{R} = ((V_\mathcal{R}, E_\mathcal{R}), \mu_\mathcal{R})$ be a rooted topology over $T \setminus T'$. Let $\mathcal{T}_u = ((V_{\mathcal{T}_u}, E_{\mathcal{T}_u}), \mu_{\mathcal{T}_u})$ be the subtopology of $\mathcal{T}$ rooted at u. Then $\mathcal{T} \cdot \mathcal{R}$ can not be an optimal topology over T if the following condition holds:*

$$\|e\|_m > \min\{\ \|\overline{p},\ \mu(t)\|_{min} \mid p \in V_{\mathcal{T}_u} \setminus \{u\},\ t \in \text{leaves}(\mathcal{T} \cdot \mathcal{R}) \setminus \text{leaves}(\mathcal{T}_u)\ \}$$

where $\mu := \mu_\mathcal{T} \odot \mu_\mathcal{R}$ and $\overline{p} \in \mathcal{P}(\Sigma)^d$ with $\overline{p}_i := \bigcup_{e \in E_\mathcal{T}} \mu_F^p[(\overline{\mathcal{T}_u})_e](p)_i$

Proof: This proof mainly consists of two parts. In the first one we argue that there is a complemented labeling for the full topology $\mathcal{T} \cdot \mathcal{R}$ that yields a long edge e. Then we

show that this allows us to construct a cheaper full topology $\mathcal{F}$ showing non-optimality of $\mathcal{T} \cdot \mathcal{R}$.

Let $\mathcal{T} \cdot \mathcal{R} = ((V_{\mathcal{T}\cdot\mathcal{R}}, E_{\mathcal{T}\cdot\mathcal{R}}), \mu_{\mathcal{T}\cdot\mathcal{R}})$. Consider an instantiation $\mu \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R})$ with $\|\mu(u),\ \mu(v)\| \geq \|e\|_m$. We know that such an instantiation must exist because of Corollary 1.4. Then

$$\begin{aligned}
\text{cost}[\mu](\mathcal{T} \cdot \mathcal{R}) &= \sum_{e' \in E_{\mathcal{T}\cdot\mathcal{R}}} \text{cost}[\mu](e') \\
&= \text{cost}[\mu](\mathcal{T}_u) + \text{cost}[\mu](e) + \sum_{e' \in E_{\mathcal{T}\cdot\mathcal{R}} \setminus E_{\mathcal{T}_u} \setminus \{e\}} \text{cost}[\mu](e') \\
&= \text{cost}(\mathcal{T}_u) + \text{cost}[\mu](e) + \sum_{e' \in E_{\mathcal{T}\cdot\mathcal{R}} \setminus E_{\mathcal{T}_u} \setminus \{e\}} \text{cost}[\mu](e') \\
&\geq \text{cost}(\mathcal{T}_u) + \|e\|_m + \sum_{e' \in E_{\mathcal{T}\cdot\mathcal{R}} \setminus E_{\mathcal{T}_u} \setminus \{e\}} \text{cost}[\mu](e')
\end{aligned}$$

Let

$$\begin{aligned}
(p, t) := \text{argmin}\{ \quad & \|\overline{p},\ \mu(t)\|_{min} \mid \\
& (p \in V_{\mathcal{T}_u} \setminus \{u\},\ t \in \text{leaves}(\mathcal{T} \cdot \mathcal{R}) \setminus \text{leaves}(\mathcal{T}_u)) \ \} \quad (I)
\end{aligned}$$

and let $x_p \in \overline{p}$ be a point that minimizes the term $\|\overline{p},\ \mu(t)\|_{min}$ in (I). Following Corollary 1.1, there exists a complemented labeling μ_u for $\overline{\mathcal{T}_u}$ such that

$$\mu_u(p) = x_p$$

and clearly, $\forall\ \mu^* \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R}) : \mu^*(t) = \mu(t)$ since t is a terminal node. Now consider the topology

$$\begin{aligned}
\mathcal{F} \ := \ & ((V_{\mathcal{T}\cdot\mathcal{R}} \setminus \{u\}, \\
& E_{\mathcal{T}\cdot\mathcal{R}} \setminus E_{\mathcal{T}_u} \cup E_{\overline{\mathcal{T}_u}} \setminus \{e\} \cup \{e_r := \{p, t\}\}), \\
& \mu_{\mathcal{F}} := \mu_r \odot \mu|_{(V_{\mathcal{T}\cdot\mathcal{R}} \setminus V_{\mathcal{T}_u})})
\end{aligned}$$

Note that the underlying tree of $\mathcal{F}$ is connected and thus $\mathcal{F}$ is indeed a topology over T. The cost of $\mathcal{F}$ is given by

$$\begin{aligned}
\text{cost}(\mathcal{F}) &= \text{cost}[\mu_F](\overline{\mathcal{T}_u}) + \text{cost}[\mu_F](e_r) + \sum_{e' \in E_{\mathcal{T}\cdot\mathcal{R}} \setminus E_{\overline{\mathcal{T}_u}} \setminus \{e\}} \text{cost}[\mu_F](e') \\
&= \text{cost}[\mu_r](\overline{\mathcal{T}_u}) + \text{cost}[\mu_F](e_r) + \sum_{e' \in E_{\mathcal{T}\cdot\mathcal{R}} \setminus E_{\overline{\mathcal{T}_u}} \setminus \{e\}} \text{cost}[\mu](e') \\
&= \text{cost}(\mathcal{T}_u) + \text{cost}[\mu_F](e_r) + \sum_{e' \in E_{\mathcal{T}\cdot\mathcal{R}} \setminus E_{\overline{\mathcal{T}_u}} \setminus \{e\}} \text{cost}[\mu](e') \\
&< \text{cost}(\mathcal{T}_u) + \|e\|_m + \sum_{e' \in E_{\mathcal{T}\cdot\mathcal{R}} \setminus E_{\overline{\mathcal{T}_u}} \setminus \{e\}} \text{cost}[\mu](e') \\
&\leq \text{cost}(\mathcal{T} \cdot \mathcal{R})
\end{aligned}$$

which implies that the topology $\mathcal{T} \cdot \mathcal{R}$ can not be an optimal one. □

Note that Theorem 1.4 is a generalization of the bottleneck Steiner distance criterion given in Theorem 1.3 since for each terminal node t_u spanned by $\mathcal{T}_u$ we have $\mu_F^p[(\overline{\mathcal{T}_u})_e](t_u) = \mu(t_u)$ and thus

$$\forall\, i \in \{1, \ldots, d\} \colon \mu(t_u)_i \in \overline{p}_i$$

what implies that

$$\|\overline{p},\ \mu(t)\|_{\min} \ \leq\ \|\mu(t_u),\ \mu(t)\|_{\min}$$

for all $p \in V_{\mathcal{T}_u} \setminus \{u\}$, $t \in \text{leaves}(\mathcal{T} \cdot \mathcal{R}) \setminus \text{leaves}(\mathcal{T}_u)$ and $t_u \in \text{leaves}(\mathcal{T}_u)$.

The idea of the following theorem is very similar to the one of Theorem 1.4, but instead of considering for e' the edges formed by a node in $\mathcal{T}_u$ and a terminal node in $\text{leaves}(\mathcal{T} \cdot \mathcal{R}) \setminus \text{leaves}(\mathcal{T}_u)$, we consider edges consisting of a node in $\mathcal{T}_u$ and a (possibly inner node) in $\text{nodes}(\mathcal{T}) \setminus \text{nodes}(\mathcal{T}_u)$.

Theorem 1.5 (heavy edge property (II)) *Let $\mathcal{T} = ((V_\mathcal{T}, E_\mathcal{T}), \mu_\mathcal{T})$ be a rooted topology over $T' \subsetneq T$, an edge $e := \{u, v\} \in E_\mathcal{T}$ where v is the father of u and let $\mathcal{R} = ((V_\mathcal{R}, E_\mathcal{R}), \mu_\mathcal{R})$ be a rooted topology over $T \setminus T'$. Let $\mathcal{T}_u = ((V_{\mathcal{T}_u}, E_{\mathcal{T}_u}), \mu_{\mathcal{T}_u})$ be the subtopology of $\mathcal{T}$ rooted at u. Then $\mathcal{T} \cdot \mathcal{R}$ can not be an optimal topology over T if*

$$\|e\|_m > \min\{\ \|\overline{p},\ y\|^* \mid p \in V_{\mathcal{T}_u} \setminus \{u\},\ y \in V_\mathcal{T} \setminus V_{\mathcal{T}_u}\ \}$$

with

$$\|\overline{p}, y\|^* := |\{\ i \in \{1, \ldots, d\} \mid\ \mu_F[\mathcal{T}](y)_i \not\subseteq \overline{p}_i\ \}|$$

and where $\mu := \mu_\mathcal{T} \odot \mu_\mathcal{R}$ and $\overline{p} \in \mathcal{P}(\Sigma)^d$ with $\overline{p}_i := \bigcup_{e \in E_\mathcal{T}} \mu_F^p[(\overline{\mathcal{T}_u})_e](p)_i$

Proof: Also this proof essentially consists of two parts. In the first one we argue that there is a complemented labeling for the full topology $\mathcal{T} \cdot \mathcal{R}$ that yields a long edge e. Then we show that this allows us to construct a cheaper full topology $\mathcal{F}$ showing non-optimality of $\mathcal{T} \cdot \mathcal{R}$.

Let $\mathcal{T} \cdot \mathcal{R} = ((V_{\mathcal{T}\cdot\mathcal{R}}, E_{\mathcal{T}\cdot\mathcal{R}}), \mu_{\mathcal{T}\cdot\mathcal{R}})$. Consider a $\mu \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R})$ with $\|\mu(u),\ \mu(v)\| \geq \|e\|_m$. We know that such an instantiation must exist because of Corollary 1.4. Recall that for all $w \in V_\mathcal{T} \colon \mu_F[\mathcal{T}](w) = \mu_F[\mathcal{T} \cdot \mathcal{R}](w)$ and thus for all $i \in \{1, \ldots, d\}$:

$$\mu_F[\mathcal{T}](y)_i \subseteq \overline{p}_i \ \Rightarrow\ \mu_F[\mathcal{T} \cdot \mathcal{R}](y)_i \subseteq \overline{p}_i$$

Since $\mu(y)_i \in \mu_F[\mathcal{T} \cdot \mathcal{R}](y)_i$ we know that

$$\mu_F[\mathcal{T}](y)_i \subseteq \overline{p}_i \ \Rightarrow\ \mu(y)_i \in \overline{p}_i$$

Recall that Lemma 1.1 implies that there exists a complemented labeling μ_u for $\overline{\mathcal{T}_u}$ such that $\mu_u(y)_i \in \overline{p}_i$ and thus

$$\mu_F[\mathcal{T}](y)_i \subseteq \overline{p}_i \ \Rightarrow\ \|\mu(y_i),\ \overline{p}_i\|_{\min} = 0$$

But this implies directly that for any $\mu \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R})$ there is a complemented labeling μ_u for $\overline{\mathcal{T}_u}$ with

$$\begin{aligned} \|\mu_u(p),\ \mu(y)\| &\leq d - |\{\ i \in \{1,\ldots,d\} \mid\ \mu_F[\mathcal{T}](y)_i \subseteq \overline{p}_i\ \}| \\ &= |\{\ i \in \{1,\ldots,d\} \mid\ \mu_F[\mathcal{T}](y)_i \not\subseteq \overline{p}_i\ \}| \end{aligned}$$

Now consider the topology

$$\begin{aligned} \mathcal{F} \ :=\ & ((V_{\mathcal{T}\cdot\mathcal{R}} \setminus \{u\}, \\ & E_{\mathcal{T}\cdot\mathcal{R}} \setminus E_{\mathcal{T}_u} \cup E_{\overline{\mathcal{T}_u}} \setminus \{e\} \cup \{e_r := \{p,y\}\}), \\ & \mu_{\mathcal{F}} := \mu_r \odot \mu|_{(V_{\mathcal{T}\cdot\mathcal{R}} \setminus V_{\mathcal{T}_u})}) \end{aligned}$$

with

$$(p,y) := \operatorname{argmin}\{\ \|\overline{p},\ y\|^* \mid\ (p \in V_{\mathcal{T}_u} \setminus \{u\},\ y \in V_{\mathcal{T}} \setminus V_{\mathcal{T}_u})\ \}$$

Then

$$\begin{aligned} \operatorname{cost}(\mathcal{F}) &= \|\mu_u(p),\ \mu(y)\| + \operatorname{cost}[\mu_u](\overline{\mathcal{T}_u}) + \sum_{e' \in E_{\mathcal{T}\cdot\mathcal{R}} \setminus E_{\overline{\mathcal{T}_u}} \setminus \{e\}} \operatorname{cost}[\mu](e') \\ &\leq |\{\ i \in \{1,\ldots,d\} \mid \mu_F[\mathcal{T}](y)_i \not\subseteq \overline{p}_i\ \}|) + \\ &\quad \operatorname{cost}(\mathcal{T}_u) + \sum_{e' \in E_{\mathcal{T}\cdot\mathcal{R}} \setminus E_{\overline{\mathcal{T}_u}} \setminus \{e\}} \operatorname{cost}[\mu](e') \\ &< \|e\|_m + \operatorname{cost}(\mathcal{T}_u) + \sum_{e' \in E_{\mathcal{T}\cdot\mathcal{R}} \setminus E_{\overline{\mathcal{T}_u}} \setminus \{e\}} + \operatorname{cost}[\mu](e') \\ &\leq \operatorname{cost}(\mathcal{T} \cdot \mathcal{R}) \end{aligned}$$

which implies that the topology $\mathcal{T} \cdot \mathcal{R}$ can not be an optimal one. □

So far we have only examined the edges contained in the partial topology $\mathcal{T}$. But there is one more edge left whose length can be bounded, namely the edge connecting $\mathcal{T}$ and $\mathcal{R}$.

Theorem 1.6 (heavy root edge property) *Let $\mathcal{T} = ((V_{\mathcal{T}}, E_{\mathcal{T}}), \mu_{\mathcal{T}}, r_{\mathcal{T}})$ be a rooted topology over $T' \subsetneq T$ and let $\mathcal{R} = ((V_{\mathcal{R}}, E_{\mathcal{R}}), \mu_{\mathcal{R}})$ be a rooted topology over $T \setminus T'$. Then $\mathcal{T} \cdot \mathcal{R}$ can not be an optimal topology over T if the following condition holds:*

$$\begin{aligned} & |\{\ i \in \{1,\ldots,d\} \mid \mu_F[\mathcal{T}](r_{\mathcal{T}})_i \cap \Big(\bigcup_{t \in T \setminus T'} \{t_i\} \Big) = \emptyset\ \}| \\ > \ & \min\{\ \|\overline{p},\ \mu(t)\|_{min} \mid p \in V_{\mathcal{T}_u} \setminus \{u\},\ t \in \operatorname{leaves}(\mathcal{T} \cdot \mathcal{R}) \setminus \operatorname{leaves}(\mathcal{T}_u)\ \} \end{aligned}$$

where $\mu := \mu_{\mathcal{T}} \odot \mu_{\mathcal{R}}$ and $\overline{p} \in \mathcal{P}(\Sigma)^d$ with $\overline{p}_i := \bigcup_{e \in E_{\mathcal{T}}} \mu_F^p[(\overline{\mathcal{T}_u})_e](p)_i$

Proof: The proof is completely analogous to the proof of Theorem 1.4. The only thing to show is that the left hand side of the inequality in the premise of the theorem is a lower bound for the edge connecting $\mathcal{T}$ and $\mathcal{R}$ in $\overline{\mathcal{T} \cdot \mathcal{R}}$. First note that

$$\forall\, i \in \{1,\ldots,d\} \colon \mu_F[\mathcal{R}](\operatorname{root}(\mathcal{R}))_i \subseteq \bigcup_{t \in T \setminus T'} \{t_i\}$$

Consider now any $\mu \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R})$. Then for all $i \in \{1, \ldots, d\}$:

$$\begin{aligned}
& \|\mu(\text{root}(\mathcal{T} \cdot \mathcal{R}))_i,\ \mu(\text{root}(\mathcal{T}))_i\| + \|\mu(\text{root}(\mathcal{T} \cdot \mathcal{R}))_i,\ \mu(\text{root}(\mathcal{R}))_i\| = 1 \\
\Leftrightarrow\ & \mu_F[\mathcal{T} \cdot \mathcal{R}](\text{root}(\mathcal{T} \cdot \mathcal{R}))_i = \mu_F[\mathcal{T}](\text{root}(\mathcal{T}))_i \cup \mu_F[\mathcal{R}](\text{root}(\mathcal{R}))_i \\
\Leftrightarrow\ & \mu_F[\mathcal{T}](\text{root}(\mathcal{T}))_i \cap \mu_F[\mathcal{R}](\text{root}(\mathcal{R}))_i = \emptyset \\
\Leftarrow\ & \mu_F[\mathcal{T}](\text{root}(\mathcal{T}))_i \cap \bigcup_{t \in T \setminus T'} \{t_i\} = \emptyset
\end{aligned}$$

Thus

$$\begin{aligned}
& \|\mu(\text{root}(\mathcal{T} \cdot \mathcal{R})),\ \mu(\text{root}(\mathcal{T}))\| + \|\mu(\text{root}(\mathcal{T} \cdot \mathcal{R})),\ \mu(\text{root}(\mathcal{R}))\| \\
\geq\ & |\{\ i \in \{1, \ldots, d\} \mid \mu_F(\text{root}(\mathcal{T}))_i \cap \left(\bigcup_{t \in T \setminus T'} \{t_i\} \right) = \emptyset\ \}|
\end{aligned}$$

completing the proof. □

Certainly the pruning conditions given in Theorems 1.4 and 1.5 can even be improved by several means. One idea uses the observation that after removing the edge $e = \{u, v\}$ the subtopology $\mathcal{T}_u$ of the given partial topology $\mathcal{T}$ can be changed in an arbitrary way, that is instead of considering $\mathcal{T}_u$, one could try to find an alternative network interconnecting the terminal nodes of $\mathcal{T}_u$. This could help improving the pruning tests in two ways. The first one is, that the cost of $\mathcal{T}_u$ could be improved and the second one is that another – possibly more costly – tree interconnecting the leaves of $\mathcal{T}_u$ could be constructed such that the cost of the reconnection is decreased. This is the subject of the next section.

1.5.2 Topology Replacement Tests

In the previous section we have discussed pruning tests based on a simple ruin-and-recreate strategy that involved removing single edges and re-establishing the connectivity property of the spanning tree. In this section we will again employ this strategy but instead of replacing edges we will try to substitute whole (partial) topologies. Since the efficiency of our implementation also heavily relies on the efficiency with which we can compute the pruning tests, we will focus on tests that can be computed very fast.

We start with a pruning test that is based on the computation of lower bounds for Steiner minimum trees over a subset of the terminals. As mentioned in the preamble of the pruning test section we postpone the question of how to compute such bounds on the cost a Steiner minimum tree to section 1.6. The basic idea is quite simple: if you are given a partial topology $\mathcal{T}$ spanning a subset T' of the terminal set T, then clearly there cannot be an optimal full topology $\mathcal{T} \cdot \mathcal{R}$ if

$$\text{cost}(\mathcal{T}) + \text{cost}_{\text{lb}}(T \setminus T') > \text{cost}_{\text{up}}(T)$$

where $\text{cost}_{\text{lb}}(T)$ denotes a lower bound on the cost of a Steiner minimum tree ST over T and $\text{cost}_{\text{up}}(T)$ denotes an upper bound on the cost of ST. This is due to the fact that the Steiner tree problem is monotone, that is if you consider only a subset of terminal points the cost of an optimal solution can only decrease.

Theorem 1.7 (lower bound test) *Let $\mathcal{T} = ((V_\mathcal{T}, E_\mathcal{T}), \mu_\mathcal{T}, r_\mathcal{T})$ be a rooted topology over $T' \subsetneq T$ and let $\mathcal{R} = ((V_\mathcal{R}, E_\mathcal{R}), \mu_\mathcal{R}, r_\mathcal{R})$ be a rooted topology over $T \setminus T'$. Then $\mathcal{T} \cdot \mathcal{R}$ can not be an optimal topology over T if*

$$\begin{aligned}\operatorname{cost}(\mathcal{T}) \;>\; & \operatorname{cost}_{\mathrm{up}}(T) - \operatorname{cost}_{\mathrm{lb}}(T \setminus T') - \\ & |\{\, i \in \{1,\dots,d\} \mid \mu_F(r_\mathcal{T})_i \cap \Big(\bigcup_{t \in T \setminus T'} \{t_i\} \Big) = \emptyset \,\}|\end{aligned}$$

Proof: Let $\mu \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R})$. Recall that

$$\begin{aligned}\|\mu(r_\mathcal{T}),\, \mu(r_\mathcal{R})\| &= |\{\, i \in \{1,\dots,d\} \mid \mu(r_\mathcal{T})_i \neq \mu(r_\mathcal{R})_i \,\}| \\ &\geq |\{\, i \in \{1,\dots,d\} \mid \mu_F(r_\mathcal{T})_i \cap \Big(\bigcup_{t \in T \setminus T'} \{t_i\} \Big) = \emptyset \,\}|\end{aligned}$$

since $\mu(r_\mathcal{R})_i \in \mu_F[\mathcal{R}](r_\mathcal{R})_i \subseteq \bigcup_{t \in T \setminus T'} \{t_i\}$. Thus

$$\begin{aligned}\operatorname{cost}(\mathcal{T} \cdot \mathcal{R}) &= \operatorname{cost}(\mathcal{T}) + \operatorname{cost}(\mathcal{R}) + \|\mu(r_\mathcal{T}),\, \mu(r_\mathcal{R})\| \\ &> \operatorname{cost}_{\mathrm{up}}(T) + \big[\operatorname{cost}(\mathcal{R}) - \operatorname{cost}_{\mathrm{lb}}(T \setminus T')\big] + \\ &\quad \|\mu(r_\mathcal{T}),\, \mu(r_\mathcal{R})\| - |\{\, i \in \{1,\dots,d\} \mid \mu_F(r_\mathcal{T})_i \cap \Big(\bigcup_{t \in T \setminus T'} \{t_i\} \Big) = \emptyset \,\}| \\ &\geq \operatorname{cost}_{\mathrm{up}}(T) + \\ &\quad \|\mu(r_\mathcal{T}),\, \mu(r_\mathcal{R})\| - |\{\, i \in \{1,\dots,d\} \mid \mu_F(r_\mathcal{T})_i \cap \Big(\bigcup_{t \in T \setminus T'} \{t_i\} \Big) = \emptyset \,\}| \\ &\geq \operatorname{cost}_{\mathrm{up}}(T)\end{aligned}$$

where the first inequality follows from the premise of the theorem and the second from the fact that $\operatorname{span}(\mathcal{R}) = T \setminus T'$ and with it $\operatorname{cost}(\mathcal{R}) \geq \operatorname{cost}_{\mathrm{lb}}(T \setminus T')$. But this proves that $\mathcal{T} \cdot \mathcal{R}$ is not an optimal topology. □

The idea of the next pruning tests is the following. Consider two partial topologies $\mathcal{T}$ and $\mathcal{T}'$ spanning the same set of terminals. If one of them – let's say $\mathcal{T}'$ – is sufficiently smaller than the other, you can substitute the bigger topology by the smaller one in a full topology $\mathcal{T} \cdot \mathcal{R}$ such that the resulting full topology $\mathcal{T}' \cdot \mathcal{R}$ is cheaper than $\mathcal{T} \cdot \mathcal{R}$, showing the non-optimality of $\mathcal{T} \cdot \mathcal{R}$. In the following theorem we discuss when it is possible to replace $\mathcal{T}$ by $\mathcal{T}'$ when connecting $\mathcal{T}'$ to the root of $\mathcal{T}$.

Theorem 1.8 (root substitutable topologies) *Let $\mathcal{T} = ((V_\mathcal{T}, E_\mathcal{T}), \mu_\mathcal{T}, r_\mathcal{T})$ be a rooted topology over $T' \subsetneq T$ and let $\mathcal{R} = ((V_\mathcal{R}, E_\mathcal{R}), \mu_\mathcal{R}, r_\mathcal{R})$ be a rooted topology over $T \setminus T'$. Then $\mathcal{T} \cdot \mathcal{R}$ can not be an optimal topology over T if there exists a topology $\mathcal{T}' = ((V_{\mathcal{T}'}, E_{\mathcal{T}'}), \mu_{\mathcal{T}'}, r_{\mathcal{T}'})$ with $\operatorname{span}(\mathcal{T}) = \operatorname{span}(\mathcal{T}')$ such that*

$$\operatorname{cost}(\mathcal{T}) > \operatorname{cost}(\mathcal{T}') + \|\mu_F[\mathcal{T}](r_\mathcal{T}),\, \mu_F[\mathcal{T}'](r_{\mathcal{T}'})\|_{\subsetneq}$$

where $\|x, y\|_{\not\subseteq} := \{\, i \in \{1,\dots,d\} \mid x_i \not\subseteq y_i \,\}$.

Proof: Let $\mu \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R})$ and $\mu' \in \mathcal{L}(\mathcal{T}')$ such that

$$\forall\, i \in \{1, \ldots, d\} \text{ with } \mu_F[\mathcal{T}](r_\mathcal{T})_i \subseteq \mu_F[\mathcal{T}'](r_{\mathcal{T}'})_i \colon\ \mu'(r_{\mathcal{T}'})_i = \mu(r_\mathcal{T})_i$$

Note that such a labeling μ' must exist because by the definition of Fitch's algorithm we have $\mu(r_\mathcal{T})_i \in \mu_F[\mathcal{T}](r_\mathcal{T})_i$. Thus, if $\mu_F[\mathcal{T}](r_\mathcal{T})_i \subseteq \mu_F[\mathcal{T}'](r_{\mathcal{T}'})_i$ then also $\mu(r_\mathcal{T})_i \in \mu_F[\mathcal{T}'](r_{\mathcal{T}'})_i$, that is there exists a $\mu' \in \mathcal{L}(\mathcal{T}')\colon \mu'(r_{\mathcal{T}'})_i = \mu(r_\mathcal{T})_i$. Now consider the full topology

$$\begin{aligned} \mathcal{F} \;:=\; & ((V_\mathcal{R} \cup V_{\mathcal{T}'} \cup \{r_\mathcal{T}\}, \\ & E_{\mathcal{T}'} \cup E_\mathcal{R} \cup \{\{r_{\mathcal{T}'},\ r_\mathcal{T}\}, \{r_\mathcal{T},\ r_\mathcal{R}\}\}), \\ & \mu_\mathcal{F} := \mu' \odot \mu|_{(V_\mathcal{R} \cup \{r_\mathcal{T}\})}) \end{aligned}$$

Then

$$\begin{aligned} \text{cost}(\mathcal{F}) &= \text{cost}(\mathcal{T}') + \text{cost}(\mathcal{R}) + \|\mu(r_\mathcal{T}),\ \mu'(r_{\mathcal{T}'})\| + \|\mu(r_\mathcal{T}),\ \mu(r_\mathcal{R})\| \\ &< \text{cost}(\mathcal{T}) + \text{cost}(\mathcal{R}) + \|\mu(r_\mathcal{T}),\ \mu(r_\mathcal{R})\| + \\ &\quad \|\mu(r_\mathcal{T}),\ \mu'(r_{\mathcal{T}'})\| - \|\mu_F[\mathcal{T}](r_\mathcal{T}),\ \mu_F[\mathcal{T}'](r_{\mathcal{T}'})\|_{\not\subseteq} \\ &\leq \text{cost}(\mathcal{T}) + \text{cost}(\mathcal{R}) + \|\mu(r_\mathcal{T}),\ \mu(r_\mathcal{R})\| \\ &= \text{cost}(\mathcal{T} \cdot \mathcal{R}) \end{aligned}$$

where the second inequality follows from the way in which we have chosen μ':

$$\begin{aligned} & \|\mu_F[\mathcal{T}](r_\mathcal{T}),\ \mu_F[\mathcal{T}'](r_{\mathcal{T}'})\|_{\not\subseteq} \\ = \;& \{\ i \in \{1, \ldots, d\} \mid \mu_F[\mathcal{T}](r_\mathcal{T})_i \not\subseteq \mu_F[\mathcal{T}'](r_{\mathcal{T}'})_i\ \} \\ = \;& d - \{\ i \in \{1, \ldots, d\} \mid \mu_F[\mathcal{T}](r_\mathcal{T})_i \subseteq \mu_F[\mathcal{T}'](r_{\mathcal{T}'})_i\ \} \\ \geq \;& d - \{\ i \in \{1, \ldots, d\} \mid \mu(r_\mathcal{T})_i = \mu'(r_{\mathcal{T}'})\ \} \\ = \;& \|\mu(r_\mathcal{T}),\ \mu'(r_{\mathcal{T}'})\| \end{aligned}$$

But this contradicts the optimality of $\mathcal{T} \cdot \mathcal{R}$. □

While the idea in the last theorem was to connect $\mathcal{T}'$ to $\mathcal{R}$ over the root node of $\mathcal{T}$, in the next theorem we discuss the possibility to connect $\mathcal{T}'$ and $\mathcal{R}$ via an edge reaching from a node in $\mathcal{T}$ to a terminal in $\mathcal{R}$.

Theorem 1.9 (substitutable topologies) *Let $\mathcal{T} = ((V_\mathcal{T}, E_\mathcal{T}), \mu_\mathcal{T}, r_\mathcal{T})$ be a rooted topology over $T' \subsetneq T$ and let $\mathcal{R} = ((V_\mathcal{R}, E_\mathcal{R}), \mu_\mathcal{R}, r_\mathcal{R})$ be a rooted topology over $T \setminus T'$. Then $\mathcal{T} \cdot \mathcal{R}$ can not be an optimal topology over T if there exists a topology $\mathcal{T}' = ((V_{\mathcal{T}'}, E_{\mathcal{T}'}), \mu_{\mathcal{T}'})$ with $\text{span}(\mathcal{T}) = \text{span}(\mathcal{T}')$ such that*

$$\begin{aligned} \text{cost}(\mathcal{T}) \;>\; & \text{cost}(\mathcal{T}') + \\ & \min\{\ \|\overline{p},\ \mu_R(t)\|_{min} \mid p \in V_{\mathcal{T}'},\ t \in \text{leaves}(\mathcal{R})\ \} - \\ & |\{\ i \in \{1, \ldots, d\} \mid \mu_F[\mathcal{T}](r_\mathcal{T})_i \cap \Big(\bigcup_{t \in T \setminus T'} \{t_i\} \Big) = \emptyset\ \}| \end{aligned}$$

where $\mu := \mu_\mathcal{T} \odot \mu_\mathcal{R}$ and $\overline{p} \in \mathcal{P}(\Sigma)^d$ with $\overline{p}_i := \bigcup_{e \in E_\mathcal{T}} \mu_F^p[(\overline{\mathcal{T}'})_e](p)_i$

Proof: Let

$$(p,t) := \text{argmin}\{\ \|\overline{p},\ \mu_R(t)\|_{min} \mid (p \in V_{\mathcal{T}'},\ t \in \text{leaves}(\mathcal{R}))\ \} \quad (I)$$

and let $x_p \in \overline{p}$ be a point that minimizes the term $\|\overline{p},\ \mu_R(t)\|_{min}$ in (I). Following Corollary 1.1, there exists a complemented labeling μ' for $\overline{\mathcal{T}'}$ such that

$$\mu'(p) = x_p$$

and clearly, $\forall\ \mu^* \in \mathcal{L}(\mathcal{R}) : \mu^*(t) = \mu_R(t)$ since t is a terminal node. Consider now the topology

$$\begin{aligned} \mathcal{F} \ := \ & ((V_{\mathcal{T}'} \cup V_{\mathcal{R}}, \\ & E_{\mathcal{T}'} \cup E_{\mathcal{R}} \cup \{e_r := \{p,t\}\}), \\ & \mu_{\mathcal{F}} := \mu' \odot \mu'_R) \end{aligned}$$

where $\mu'_R \in \mathcal{L}(\mathcal{R})$. Then

$$\begin{aligned} \text{cost}(\mathcal{F}) &= \text{cost}(\mathcal{T}') + \text{cost}(\mathcal{R}) + \text{cost}(e_r) \\ &= \text{cost}(\mathcal{T}') + \text{cost}(\mathcal{R}) + \|\mu'(p),\ \mu_R(t)\| \\ &< \text{cost}(\mathcal{T}) + \text{cost}(\mathcal{R}) + \\ &\quad |\{\ i \in \{1,\ldots,d\} \mid \mu_F[\mathcal{T}](r_{\mathcal{T}})_i \cap \left(\bigcup_{t \in T \setminus T'} \{t_i\} \right) = \emptyset\ \}| \\ &\leq \text{cost}(\mathcal{T}) + \text{cost}(\mathcal{R}) + \|\mu(r_{\mathcal{T}}),\ \mu(r_{\mathcal{R}})\| \\ &= \text{cost}(\mathcal{T} \cdot \mathcal{R}) \end{aligned}$$

for a $\mu \in \mathcal{L}(\mathcal{T} \cdot \mathcal{R})$. Note that this is a contradiction to the optimality of the full topology $\mathcal{T} \cdot \mathcal{R}$. □

The idea of the following pruning test is a kind of a generalization of the edge replacement tests that we have seen in the previous section. Instead of replacing a single edge of $\mathcal{T}$ we remove all edges and try to re-connect the terminal nodes in $\mathcal{T}$ in a cheap way. The difference between the former approaches and this one is now the following. In the former theorems we have tried to find a network re-connecting these terminals disregarding the fact that we have some knowledge about the terminals spanned by $\mathcal{R}$, namely that they are already connected. Thus, when trying to connect the terminal nodes of $\mathcal{T}$ again we can make use of $\mathcal{R}$ as a mean of taking shortcuts. The idea is now to determine a minimum spanning tree over the terminal nodes of $\mathcal{T}$ and a new node M that represents the minimal distance of a point in leaves($\mathcal{T}$) and a terminal in $\mathcal{R}$ (see Figure 1.8 for an example in Euclidean space).

Theorem 1.10 (MST-substitute) *Let $\mathcal{T} = ((V_{\mathcal{T}}, E_{\mathcal{T}}), \mu_{\mathcal{T}}, r_{\mathcal{T}})$ be a rooted topology over $T' \subsetneq T$ and let $\mathcal{R} = ((V_{\mathcal{R}}, E_{\mathcal{R}}), \mu_{\mathcal{R}}, r_{\mathcal{R}})$ be a rooted topology over $T \setminus T'$. Consider the complete weighted graph $G = (V, E, w)$ with*

$$V := \text{leaves}(\mathcal{T}) \cup \{M\}$$

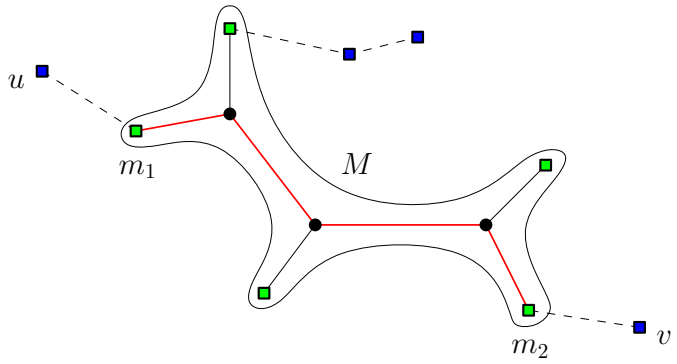


Figure 1.8: An example of the MST-substitute property in the Euclidean plane. The green points represent the terminals spanned by a (not known) partial topology $\mathcal{R}$. The blue points correspond to the terminals spanned by the partial topology $\mathcal{T}$. If you want to connect the nodes u and v, instead of adding the expensive edge $\{u, v\}$, you can use the network of $\mathcal{R}$ as a shortcut (red path) and add the cheaper edges $\{u, m_1\}$ and $\{v, m_2\}$.

for some $M \notin \text{leaves}(\mathcal{T})$ *and*

$$\forall\, u, v \in \text{leaves}(\mathcal{T}): \qquad w(\{u,v\}) := \|\mu_{\mathcal{T}}(u),\ \mu_{\mathcal{T}}(v)\| \quad \textit{and}$$
$$w(\{u,M\}) := \min_{m \in \text{leaves}(\mathcal{R})} \{\ \|\mu_{\mathcal{T}}(u),\ \mu_{\mathcal{T}}(m)\|\ \}$$

Then $\mathcal{T} \cdot \mathcal{R}$ *can not be an optimal topology over* T *if*

$$\text{cost}(\mathcal{T}) > |\text{MST}(G)| - |\{\ i \in \{1,\ldots,d\} \mid \mu_F[\mathcal{T}](r_{\mathcal{T}})_i \cap \Big(\bigcup_{t \in T \setminus T'} \{t_i\}\Big) = \emptyset\ \}|$$

where $|\text{MST}(G)|$ *is the size of a minimum weight spanning tree over* G*.*

Proof: The proof is straightforward. Let $\text{MST}(G) = (V_M, E_M)$, consider now the topology

$$\begin{aligned} \mathcal{F} \;:=\; & ((V_{\mathcal{R}} \cup \text{leaves}(\mathcal{T}), \\ & E_{\mathcal{R}} \cup E'), \\ & \mu_{\mathcal{F}} := \mu_{\mathcal{R}} \odot \mu_{\mathcal{T}}|_{\text{leaves}(\mathcal{T})} \end{aligned}$$

where

$$\begin{aligned} E' \;=\; & \{\ \{u,v\} \in E_M \mid u \notin \mathcal{R}\ \wedge\ v \notin \mathcal{R}\ \} \cup \\ & \{\ \{u,m\} \in E_M \mid \{u,M\} \in E_M\ \wedge\ m = \operatorname*{argmin}_{m \in \text{leaves}(\mathcal{R})} \{\ \|\mu_{\mathcal{T}}(u),\ \mu_{\mathcal{T}}(m)\|\ \}\ \} \end{aligned}$$

Clearly, by definition of the minimum spanning tree $\mathcal{F}$ is connected and thus a topology.

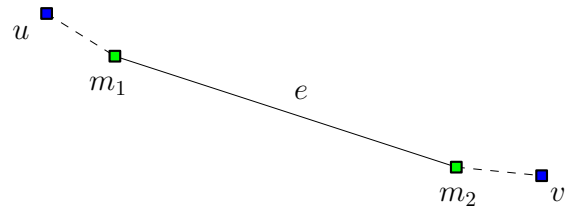


Figure 1.9: An example demonstrating the performance of the MST-substitute test.

The cost of $\mathcal{F}$ is then

$$\begin{aligned}
\text{cost}(\mathcal{F}) &= \text{cost}(\mathcal{R}) + |MST(G)| \\
&< \text{cost}(\mathcal{R}) + \text{cost}(\mathcal{T}) + \\
&\quad |\{\, i \in \{1,\ldots,d\} \mid \mu_F[\mathcal{T}](r_{\mathcal{T}})_i \cap \Big(\bigcup_{t \in T \setminus T'} \{t_i\} \Big) = \emptyset \,\}| \\
&\leq \text{cost}(\mathcal{T} \cdot \mathcal{R})
\end{aligned}$$

thus showing that $\mathcal{T} \cdot \mathcal{R}$ cannot be optimal. □

Interestingly Polzin describes a similar pruning test in his thesis [Pol03]. Following his approach we would compare the cost of $\mathcal{T}$ with the the cost of a minimum spanning tree over leaves($\mathcal{T}$) with respect to the bottleneck Steiner distances of T. Let us first argue that our approach is not inferior to his. Recall that the cost of an edge e in a minimum spanning tree never exceeds the minimal cost of an edge in the cut induced by e. Consider now an edge $e = \{x, y\} \in E$ in the minimum spanning tree as constructed by our approach. Let us denote by $(X, Y) \subseteq V \times V$ the partition of the node set of G induced by e. Let us assume without loss of generality that $M \in Y$. Let us simplify the discussion by setting

$$\|x, y\| := \|(\mu_{\mathcal{T}} \odot \mu_{\mathcal{R}})(x), (\mu_{\mathcal{T}} \odot \mu_{\mathcal{R}})(y)\|$$

for $x, y \in V_{\mathcal{R}} \cup V_{\mathcal{T}}$. Then

$$\|x, y\| = \min_{x_j \in X, y_j \in Y \cup \{M\}} \|x_j, y_j\|$$

Furthermore let $\overline{e} = \{\overline{x}, \overline{y} \setminus \{M\}\}$ be the corresponding edge in the cut (X, Y) contained in the minimum spanning tree of Polzin's approach. We have

$$\begin{aligned}
\|\overline{x}, \overline{y}\| &= \min_{x_j \in X, y_j \in Y} \{\text{bnsd}(x_j, y_j)\} \\
&= \min_{x_j \in X, y_j \in Y} \{ \max_{(x', y') \in (x_j \rightsquigarrow y_j)} \{\|x', y'\|\} \}
\end{aligned}$$

Consider now the $x_i \in X, y_i \in Y$ and $x' \in \text{leaves}(\mathcal{T} \cdot \mathcal{R}), y' \in \text{leaves}(\mathcal{T} \cdot \mathcal{R})$ attaining the value $\|\overline{x}, \overline{y}\|$ in this expression. Let $e_k = (x_k, y_k) \in (x_j \rightsquigarrow y_j)$ be the first edge leaving X, that is e_k is the first edge on the path $x_j \rightsquigarrow y_j$ such that $y_k \notin X$. If

$y_k \in Y$, $\|x, y\| \leq \|x_k, y_k\|$ since $\|x, y\| \leq \min_{x_j \in X, y_j \in Y} \|x_j, y_j\|$. If $y_k \notin Y$, we know that $y_k \in \text{leaves}(\mathcal{R})$, but again $\|x, y\| \leq \|x_k, y_k\|$ since

$$\begin{aligned} \|x, y\| &\leq \min_{x_j \in X} \|x_j, M\| \\ &= \min_{x_j \in X, z \in \text{leaves}(\mathcal{R})} \|x_j, z\| \end{aligned}$$

Since $\text{bnsd}(x_j, y_j) \geq \|e_j\|$ we have in particular that $\|x, y\| \leq \|\overline{x}, \overline{y}\|$. But our approach is not only at least as good as the bottleneck Steiner distance approach, we are actually even better. Consider the example in the Euclidean space drawn in Figure 1.9. The bottleneck Steiner distance of the nodes u and v is obviously given by $|e|$. But on the other hand, the size of the minimum spanning tree that we construct is given by $|\{u, m_1\}| + |\{v, m_2\}| < |e|$.

1.5.3 Preprocessing Techniques

So far we have described properties that can show the non-optimality of a given partial topology. In this section, we discuss methods to shrink the search space in advance before starting our pruning based algorithm.

In [NK00] the authors describe a preprocessing technique for decreasing the length of the input strings respectively for decreasing the dimension d of a problem instance. A column C^i is called *parsimony informative* if there are at least two elements in C^i, each appearing at least twice. If a column C^i is not parsimony informative, an optimal topology for the instance excluding the i-th components of the strings in T is also optimal for the complete instance T. This leads to the following preprocessing method: first, for each parsimony non-informative column C^i remove the i-th element from the input sequences in T. After this step it is possible that two sequences have become identical, thus in the second step we check for duplicate sequences and delete them (i.e., we keep exactly one of them). Note, that after removing sequences from T it can happen that columns have become parsimony non-informative that have been informative before. Thus we start again with the first step until we cannot find anymore a non-informative column.

Another preprocessing method that we propose is justified by the following theorem. The main insight here is that if we can show for some terminal t that the distance of t to some other terminal is never bigger than the distance to any constructible Steiner point then we can first construct a Steiner minimum tree for $T \setminus \{t\}$ and then connect t to the closest terminal.

Theorem 1.11 *Given an instance of the Steiner minimum tree problem $T \subseteq \Sigma^d$, if*

$$\exists\, t \in T\colon \min_{t' \in T \setminus \{t\}} \{\, \|t,\, t'\| \,\} \;\leq\; \|t,\, r\|_{\min}$$

where $r_i := \bigcup_{f \in T \setminus \{t\}} \{f_i\}$, then with $t' := \text{argmin}_{t' \in T \setminus \{t\}} \{\, \|t,\, t'\| \,\}$

$$S := ((V_R \cup \{v_t\},\; E_R \cup \{v_{t'}, v_t\},\; \mu_R \odot \mu')$$

is a Steiner minimum tree over T if $S_R := ((V_R, E_R), \mu_R)$ is a Steiner minimum tree over $T \setminus \{t\}$ and where $\mu' \colon \{v_t\} \to \Sigma^d; v_t \mapsto t'$ and $v_{t'} \in V_R \colon \mu_R(v_{t'}) = t'$.

Proof: Since S_R is a Steiner minimum tree, the underlying tree is connected and therefore, S is connected too. So we know that S is a topology. The next step is to show that S is optimal. Given an arbitrary Steiner minimum tree $X = ((V_X, E_X), \mu_X)$ over T, following Lemma 1.4 we have

$$\forall\, v \in V_X \colon \mu_X(v) \in \bigcup_{f \in T} \{f_i\}$$

Since $\min_{t' \in T \setminus \{t\}} \{\ \|t, t'\|\ \} \leq \|t, r\|_{\min}$ we can transform X into another Steiner minimum tree $X' = (V_{X'}, E_{X'}), \mu_{X'})$ such that there is an edge $\{v_{t'}, v_t\} \in E_{X'}$ with $\mu_{X'}(v_{t'}) = t'$ and $\mu_{X'}(v_t) = t$, which completes the proof. □

Again, Theorem 1.11 allows us to reduce the number of input sequences which could lead to parsimony non-informative columns. Thus, one has to apply both preprocessing steps in a cyclic fashion until no further input reduction is possible.

1.5.4 Implementation Issues

Let us make some remarks on the implementation concerning the pruning tests of this section. Comments on the implementation of the lower bound tests can be found in the next section 1.6.

Cascading Pruning Tests

The main theorems of this section are very powerful in the sense that they cover a lot of possible pruning tests. For example – as stated already above – Theorem 1.4 includes the bottleneck Steiner distance test. Instead of checking whether a partial topology $\mathcal{T}$ satisfies one of these constraints, we extract some easier to compute but less strict pruning conditions from these theorems. If we can find that $\mathcal{T}$ is prunable by such a fast test, we do not have to apply the more costly tests. For example before we test the bottleneck Steiner distance condition we check if the edge under consideration is bigger as the so called global bottleneck Steiner distance which is simply the biggest of all bnsds. This test is very simple as this value is independent of $\mathcal{T}$ and therefore does not have to computed separately for each partial topology.

Amortized Fitch Range Costs

If you consider a partial topology $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ and you want to apply for example Theorem 1.4 or 1.5 you have to compute the Fitch range function values for all nodes in $\mathcal{T}$ and for all possible ways of rooting $\overline{\mathcal{T}}$. Note that this can be done in time linear in the size of $\mathcal{T}$ in the following way.

For each node u in $\overline{\mathcal{T}}$ we store three range points r_i, each representing the Fitch range function value for node u when considering the topology $(\overline{\mathcal{T}})_{\{u,v_i\}}$ induced by the edge $\{u, v_i\}$ where v_i is an adjacent node of u. Note that for computing such a Fitch range function value r_i one has to compute the corresponding r_i of two child nodes of u, that is we recurse these child nodes and stop if we have already computed the

values r_i before or if we reach a leaf – in which case the r_i are exactly the labels of the leaves. Since for each node each r_i is computed exactly once, the total running time for computing all r_i is linear in the number of nodes.

Topology Replacement Tests

For the Theorems 1.8 and 1.9 we store for a small set of promising candidates pointers to them in a hash table whose keys are fingerprints of the leaves that are spanned by the topology. This enables us to find in constant time topologies that we have encountered before and against we test the current partial topology.

1.6 Lower Bounds

As we have promised in Section 1.5.2 we will now show how one can efficiently compute lower bounds on the cost of a Steiner minimum tree over some $T \subseteq \Sigma^d$.

First, let us simplify some notation from previous sections. In section 1.3 we have defined a Steiner tree over T to be a fully labeled topology over T. The purpose of this was to define Steiner trees as special cases of topologies. In this section this is not necessary anymore, since we do not use the concept of a topology anymore. Thus, when we talk about a Steiner tree in the following we mean:

Definition 1.27 (Steiner tree) *Given a set $T \subseteq \Sigma^d$ of* terminals*, a* Steiner tree *ST over T is a tree $ST = (T \cup S, E)$ spanning T for some subset $S \subseteq \Sigma^d$, called the set of* steiner nodes*.*

Note that there is a bijective mapping from this definition of a Steiner tree to the definition given in section 1.3, that is both definitions are equivalent, that is all definitions and notations concerning Steiner trees given in 1.3 transfer directly to the above definition.

This sections is organized in two parts. In the first part we derive simple lower bounds by the fact that the Steiner minimum tree problem is monotone. In the second part we derive lower bounds by decomposing problem instances into smaller instances. The main idea is to compute lower bounds for these subproblems and then to combine them to obtain a lower bound for the original problem instance.

1.6.1 Lower Bounds by Minors

The key insight for obtaining lower bounds in this section is the fact that the Steiner minimum tree problem is monotone, that is

Theorem 1.12 *Given a problem instance $T \subseteq \Sigma^d$. Then for all $T' \subseteq T$ we have*

$$\text{cost}(\text{SMT}(T')) \leq \text{cost}(\text{SMT}(T))$$

Proof: Follows directly by the fact that any Steiner minimum tree over T is also a spanning tree over T'. □

Since problem instances of very small size are easy to compute by simple exhaustive search, the idea is to determine a small subset T' of T such that $\text{cost}(\text{SMT}(T'))$ is

large. Because $|T|$ is usually fairly small (as we will see in the experimental section) it is even possible to determine such a set T' by trying out all possible small subsets of T. If we choose sizes for T' that are too large for doing so we propose the following heuristic that turned out to perform very good in practice: choose a pair of terminals with largest distance, then consecutively add one point after the other that maximizes the distance to all other so far added points until you desired size for T'.

Even though this way of obtaining lower bounds seems to be very naive, it is a very fast way. Note that each lower bound pruning test that succeeds in discarding a topology using these bounds saves us from determining the more sophisticated lower bounds of the next section that waste much more running time.

1.6.2 Lower Bounds by Dimension Partitioning

As mentioned in the introduction of this section the idea of computing lower bounds in this part is to decompose a problem instance into smaller problems and to derive for these subproblems lower bounds. Before we start diving into the problem we have to state some definition and fix some notation:

Let us first define the kind of subproblems that we consider. Given a problem instance $T \subseteq \Sigma^d$ of the Steiner minimum tree problem, for some $P \subseteq \mathbb{N}$ we define

$$\begin{aligned}\mathcal{S}_T(P) := \{\ & t \in \Sigma^{|P|} \mid \exists\, \bar{t} \in T \colon \forall\, j \in \{1,\dots,|P|\} \colon \bar{t}_{i_j} = t_j \\ & \text{where } P = \{\, i_1, i_2, \dots, i_{|P|} \,\} \text{ and } i_1 \le i_2 \le \dots \le i_{|P|} \,\}\end{aligned}$$

that is, $\mathcal{S}_T(P)$ is the set of sub-sequences of the ones in T defined by the indices given in P. Furthermore let us extend some definition that we have given before in 1.3.

Definition 1.28 *Let $\mathcal{T} = (T \cup S, E)$ be a Steiner tree over $T \subseteq \Sigma^d$. Then for some $P \subseteq \{1,\dots,d\}$ we define*

$$\mathrm{cost}_P(\mathcal{T}) = \sum_{\{u,v\} \in E} \left(\sum_{i \in P} \|u_i,\ v_i\| \right)$$

that is, $\mathrm{cost}_P(\mathcal{T})$ denotes the cost of $\mathcal{T}$ only considering these elements of the points in $\mathcal{T}$ whose positions are contained in P.

The justification for decomposing the problem into such subproblems is given by the following two theorems (see also [HHPM05]).

Theorem 1.13 (partitioning theorem) *Assume, you are given a partitioning*

$$(P_i)_{i \in \{1,\dots,k\}} \in \mathcal{P}(\{1,\dots,d\})^k$$

of the set $\{1,\dots,d\}$ then

$$\sum_{i \in \{1,\dots,k\}} \mathrm{cost}_{\mathrm{lb}}(\mathcal{S}_T(P_i)) \ \le\ \mathrm{cost}(\mathrm{SMT}(T))$$

Proof: Let $\mathcal{T} = (V, E)$ b a Steiner minimum tree over T. Then

$$\begin{aligned}
\text{cost}(\mathcal{T}) &= \sum_{e \in E} \text{cost}(e) = \sum_{\{u,v\} \in E} \sum_{i \in \{1,\ldots,d\}} \|u_i,\ v_i\| \\
&= \sum_{j \in \{1,\ldots,k\}} \left(\sum_{\{u,v\} \in E} \sum_{i \in P_j} \|u_i,\ v_i\| \right) \\
&\geq \sum_{j \in \{1,\ldots,k\}} \text{cost}(\mathcal{S}_T(P_i)) \\
&\geq \sum_{j \in \{1,\ldots,k\}} \text{cost}_{\text{lb}}(\mathcal{S}_T(P_i))
\end{aligned}$$

□

But how can we generalize Theorem 1.13 to the case in which we have lower bounds for overlapping indices? The answer lies in the formulation of an integer linear program:

Theorem 1.14 (ILP lower bounds) *Suppose you are given subsets*

$$(C_j)_{j \in \{1,\ldots,k\}} \in \mathcal{P}(\{1, \ldots, d\})^k$$

Let $x \in N^d$ be a solution of the following integer linear program:

$$\begin{aligned}
\textit{minimize} \quad & \sum_{i=1}^{d} x_i \\
s.t.: \quad & \sum_{i \in C_j} x_i \geq \text{cost}_{\text{lb}}(\mathcal{S}_T(C_j)) \quad \textit{for all } j \in \{1, \ldots, k\} \\
& x \in \mathbb{N}^d
\end{aligned}$$

Then $\sum_{i=1}^{d} x_i$ is a lower bound on the cost of a Steiner minimum tree over T.

Proof: The proof is straightforward. Consider an arbitrary Steiner tree $\mathcal{T}$ with $x_i := \text{cost}_i(\mathcal{T})$. Since any Steiner tree $\mathcal{T}$ must satisfy the constraints of the ILP the claim is true. □

But how can we solve the integer linear program given in Theorem 1.14? Unfortunately it is NP-hard to find an exact solution, since vertex cover is a special case of this ILP by considering $x_i \in \{0, 1\}$ as the decision variables describing if node i is contained in the cover or not and the by considering C_i as the two dimensional subsets of the node set denoting the edges. On the other hand the LP-relaxation of this ILP also yields a lower bounds since any optimal solution for the relaxed ILP is not bigger than an optimal solution for the ILP.

When restricting the discussion to the case in which $|C_j| \leq 2$ for all $j \in \{1, \ldots, k\}$, instead of using a linear programming solver we can determine this lower bound combinatorially. We first observe that in this case the above ILP is equivalent to the following ILP

$$
\begin{aligned}
\textit{minimize} \quad & \sum_{i=1}^{d} x_i \\
s.t.: \quad & x_i + x_j \geq \text{cost}_{\text{lb}}(\mathcal{S}_T(\{i,j\})) && \forall\, i,j \in \{1,\ldots,d\}, i \neq j \\
& x_i \geq \text{cost}_{\text{lb}}(\mathcal{S}_T(\{i\}) && \forall\, i \in \{1,\ldots,d\} \\
& x \in \mathbb{N}^d
\end{aligned}
$$

which can be seen as follows. For any subset $S \subseteq \{1,\ldots,d\}$ of cardinality at most 2 that is not contained in $(C_i)_{i\in\{1,\ldots,k\}}$ we can set $\text{cost}_{\text{lb}}(\mathcal{S}_T(S)) := 0$ yielding constraints that are also valid for the original ILP since $x \geq 0$. By subtracting from each variable x_i its lower bound $\text{cost}_{\text{lb}}(\mathcal{S}_T(\{i\})$ we obtain the equivalent ILP

$$
\begin{aligned}
\textit{minimize} \quad & \sum_{i=1}^{d} y_i \\
s.t.: \quad & y_i + y_j \geq l_{ij} && \forall\, i,j \in \{1,\ldots,d\}, i \neq j \\
& y \in \mathbb{N}^d
\end{aligned}
$$

where $l_{ij} := \text{cost}_{\text{lb}}(\mathcal{S}_T(\{i,j\})) - \text{cost}_{\text{lb}}(\mathcal{S}_T(\{i\}) - \text{cost}_{\text{lb}}(\mathcal{S}_T(j))$.

In the next step we construct the dual of the relaxed ILP

$$
\begin{aligned}
\textit{maximize} \quad & \sum_{\substack{i,j\in\{1,\ldots,d\} \\ i\neq j}} l_{ij} \cdot z_{ij} \\
s.t.: \quad & \sum_{\substack{j\in\{1,\ldots,d\} \\ j\neq i}} z_{ij} \leq 1 && \forall\, i \in \{1,\ldots,d\} \\
& z_{ij} \geq 0
\end{aligned}
$$

But this linear program can be solved very efficiently since it can be translated into a maximum weight bipartite matching problem in a certain graph:

Consider the complete bipartite graph G with vertex sets $\{u_1,\ldots,u_d\}$ and $\{v_1,\ldots v_d\}$. We define the weight w of an edge (u_i,v_j) as $w((u_i,v_j)) := l_{ij}$. We show that the optimal objective value of the dual linear program is exactly half the weight of the maximum weight matching in G: For any matching M, we can construct a solution of the LP with a cost of $w(M)/2$ by setting $z_{ij} := |\{(u_i,v_j),(u_j,v_i)\} \cap M|/2$. On the other hand, let z be an LP solution and let $x_{(u_i,v_j)} = x_{(v_i,u_j)} = z_{ij}$. It is easy to see that x is contained in the matching polytope of the bipartite graph. Thus there is a matching of weight at least $w(x)$ which is twice the value of z.

Another advantage of considering the dual linear program is that any feasible solution is by duality theory automatically a lower bound on the cost of a Steiner minimum tree. We will explain later how we exploit this in our implementation. In this context note that the lower bound $\frac{1}{(d-1)} \sum_{1\leq i<j\leq d} l_{ij}$ proposed in [HHPM05] is not better than

this LP bound as $x_{ij} = 1/(d-1)$ is a feasible solution of the dual LP with this objective value.

Now that we know how to combine lower bounds for these subproblems to a lower bound for the whole problem we have not answered the question how to obtain lower bounds for these low dimensional problems. This is what we will do next.

One-Dimensional Steiner Minimum Trees

A well known and very easy to see property of one-dimensional Steiner minimum trees is given by

Theorem 1.15 (discrepancy bound) *If $d = 1$ then the cost of a Steiner minimum tree over T is given by $|T| - 1$.*

Proof: We will show later in Theorem 1.17 that $\text{cost}(\text{SMT}(T)) \geq |T| - 1$. So what is left to show is that also $\text{cost}(\text{SMT}(T)) \leq |T| - 1$. But this is trivially true since any minimum spanning tree over T is also a Steiner tree over T with an empty set of Steiner nodes. Since any spanning tree over T has exactly $|T| - 1$ edges all of which have a length of exactly 1, the cost of such a minimum spanning tree is exactly $|T| - 1$. Thus, in total we have that $\text{cost}(\text{SMT}(T)) = |T| - 1$. □

Using the partitioning Theorem 1.13, we can construct the so called *single column discrepancy* lower bound for the entire problem by simply adding up the discrepancy bounds for all $(\mathcal{S}_T(\{i\}))_{i\in\{1,\ldots,d\}}$. Note that this bound is usually not optimal even for very small problem instances like

$$T = \{\ (A, A), (A, B), (B, A), (A, B)\ \}$$

where the optimal cost is 3 while the single column discrepancy yields a bound of 2.

Two-Dimensional Steiner Minimum Trees

In this subsection we will show that the Steiner minimum tree problem in two dimensions is exactly solvable in polynomial time. We do this by showing that a minimum spanning tree over the terminal set T is a Steiner minimum tree over T with an empty set of Steiner nodes.

Theorem 1.16 (MST bound) *If $d = 2$ then the cost of a Steiner minimum tree over T is given by* $\text{cost}(\text{MST}(T))$.

Proof: Let $\mathcal{T} = \text{MST}(T \cup S)$ be an SMT with the minimal number of Steiner points S. Note that we make no assumptions on the length of the edges or on the degree of the Steiner points. If such a Steiner tree is not unique, we choose one that minimizes $\sum_{s\in S} \deg(s)$.

We want to show $S = \emptyset$. Assume otherwise that S is non-empty and let s be a Steiner point in S. If the degree of s is smaller than two we can simply remove s and its incident edges. So let us assume that the degree of s is at least two. If the degree of s equals to two, we can remove s from S and connect the two points adjacent to

obtain a tree of at most the same cost as $\mathcal{T}$ with fewer Steiner points – recall that the cost function is a metric. Similarly, we can show that we are able to remove s if one of the edges incident to s has length 0.

Now assume that $\deg(s) \geq 3$. If the length of an edge $\{u, s\}$ incident to s is 2, that is maximal, we can remove this edge from $\mathcal{T}$ and obtain two connected components, both containing at least one terminal - otherwise the component itself containing terminal nodes would be a smaller Steiner tree. We connect u to a terminal in the other component and get a Steiner tree of the same length as $\mathcal{T}$ where the total degree of the Steiner points is smaller.

Thus we can assume that all edges incident to s have length 1. We partition the set of nodes adjacent to s into two sets S_1, S_2, where the ith element of the points in S_i is the ith element of s. Removing all edges incident to s and connecting the vertices in S_1 and S_2 with edges of length 1 gives two connected components. The total cost for doing so is the cost of $\mathcal{T}$ minus 2. Connecting two terminals in the two components gives a Steiner tree of the same cost as $\mathcal{T}$ whose number of Steiner points is smaller. Thus, $S = \emptyset$, which completes the proof. □

Three-Dimensional Steiner Minimum Trees

In this subsection we will show lower bounds for the case in which the dimension of the Steiner minimum tree problem is given by 3.

The following lemma states that if we have components of points with pairwise distance not bigger than three, then we can simply compute Steiner minimum trees for these components and connect them via a minimum spanning tree to obtain a Steiner minimum tree over T.

Lemma 1.8 *Let $d = 3$ and let $C_1, \ldots C_k$ be the connected components of the graph $G = (T, E)$ with*

$$E = \{ \{u, v\} \in T \times T \mid \|u,\ v\| \leq 2 \}$$

Furthermore, let c_i be an arbitrary representative of the component C_i. Then

$$\bigcup_{i=1}^{k} SMT(C_k) \cup MST(\bigcup_{i=1}^{k} \{c_i\})$$

is a Steiner minimum tree over T.

Proof: Given an Steiner minimum tree, we call its maximal subtrees whose inner nodes are Steiner nodes, its *full components*. Consider a Steiner minimum tree $\mathcal{S}$ with a maximal number of full components. Note that it is sufficient to show that this tree has no full component spanning terminals contained in different connected components in $(C_i)_{i \in \{1,\ldots,k\}}$. Assume otherwise and let t be a full component containing terminals from different connected components. The idea is now to show that we can find another Steiner tree spanning the same terminal as t which is not more costly but consists of more than one full component. Consider for this the topology $\mathcal{T}_t = ((V, E), \mu_t)$ corresponding to t. Using the same method as used in Theorem 1.1 we modify $\mathcal{T}_t$ in such a way that for all Steiner nodes $s \in V$ we have that $\deg(s) = 3$. Note that after this modification $\mathcal{T}$ is in standard form since t was a full component and therefore, for

all leaves l in $\mathcal{T}_t$, $\deg(l) = 1$. It is easy to see that $\mathcal{T}_t = \overline{(\mathcal{T}_1 \cdot \mathcal{T}_2) \cdot \mathcal{T}_3} = \overline{(\mathcal{T}_1 \cdot \mathcal{T}_3) \cdot \mathcal{T}_2}$ for topologies $\mathcal{T}_i$ with:

$$\exists\, i, j \in \{1, \ldots, k\}, i \neq j \colon \operatorname{span}(\mathcal{T}_1) \subseteq C_i \text{ and } \operatorname{span}(\mathcal{T}_2) \subseteq C_j$$

Let us define $r_1 := \operatorname{root}(\mathcal{T}_1)$, $r_2 := \operatorname{root}(\mathcal{T}_2)$, $r_3 := \operatorname{root}(\mathcal{T}_3)$, $r_{12} := \operatorname{root}(\mathcal{T}_1 \cdot \mathcal{T}_2)$, $r_{13} := \operatorname{root}(\mathcal{T}_1 \cdot \mathcal{T}_3)$ and

$$\bar{i} := |\{\, i \in \{0,1,2\} \mid \mu_F[\mathcal{T}_1 \cdot \mathcal{T}_2](r)_i \cap \mu_F[\mathcal{T}_3](\operatorname{root}(\mathcal{T}_3))_i \neq \emptyset \,\}|$$

If $\bar{i} > 0$ we assume in the following discussion that without loss of generality in the definition of $\bar{i}$ the ranges with smallest indices are not disjoint. We know that for all $i \in \{0,1,2\}$:

$$\mu_F[\mathcal{T}_1](r_1)_i \cap \mu_F[\mathcal{T}_2](r_2)_i = \emptyset$$

since the pairwise distance of elements in $\operatorname{span}(\mathcal{T}_1)$ and in $\operatorname{span}(\mathcal{T}_2)$ is 3. Thus

$$\mu_F[\mathcal{T}_1 \cdot \mathcal{T}_2](r_{12})_i = \mu_F[\mathcal{T}_1](r_1)_i \cup \mu_F[\mathcal{T}_2](r_2)_i$$

Then we distinguish between the four cases

- $\bar{i} = 0$: consider the topology $(\mathcal{T}_1 \cdot \mathcal{T}_2) \cdot \mathcal{T}_3$. There is a $\mu \in \mathcal{L}((\mathcal{T}_1 \cdot \mathcal{T}_2) \cdot \mathcal{T}_3)$ such that $\|\mu(r_1), \mu(r)\| = 3$. But then we can simply remove this edge $\{r_1, r\}$ from $\mathcal{T}_t$ and reconnect the resulting components via an arbitrary edge between two terminals in these components without increasing the cost, thus increasing the number of full components.

- $\bar{i} = 1$: consider the topology $(\mathcal{T}_1 \cdot \mathcal{T}_2) \cdot \mathcal{T}_3$. We have

  $$\mu_F[\mathcal{T}_1 \cdot \mathcal{T}_2](r)_0 \cap \mu_F[\mathcal{T}_3](r_3)_0 \neq \emptyset$$

  thus there is a terminal $t_3 \in \operatorname{span}(\mathcal{T}_3)$ such that $(t_3)_0 \in \mu_F[\mathcal{T}_1 \cdot \mathcal{T}_2](r)_0$. Thus – without loss of generality – there exists $t_1 \in \operatorname{span}(\mathcal{T}_1)$: $(t_3)_0 = (t_1)_0$, implying that $\|t_3, t_1\| \leq 2$. On the other hand there exists a $\mu \in \mathcal{L}((\mathcal{T}_1 \cdot \mathcal{T}_2) \cdot \mathcal{T}_3)$ with $\|\mu(r), \mu(r_1)\| = 2$. Again, removing (r, r_1) and adding (t_3, t_1) increases the number of full components without increasing the cost.

- $\bar{i} = 2$: now consider the topology $(\mathcal{T}_1 \cdot \mathcal{T}_3) \cdot \mathcal{T}_2$. We have

  $$\mu_F[\mathcal{T}_1 \cdot \mathcal{T}_2](r)_0 \cap \mu_F[\mathcal{T}_3](r_3)_0 \neq \emptyset \quad \text{and}$$

  $$\mu_F[\mathcal{T}_1 \cdot \mathcal{T}_2](r)_1 \cap \mu_F[\mathcal{T}_3](r_3)_1 \neq \emptyset$$

  Consider now node r_1. If

  $$\mu_F[\mathcal{T}_1](r_1)_0 \cap \mu_F[\mathcal{T}_3](r_3)_0 \neq \emptyset \quad \text{and}$$

  $$\mu_F[\mathcal{T}_1](r_1)_1 \cap \mu_F[\mathcal{T}_3](r_3)_1 \neq \emptyset$$

  then there exists a $\mu \in \mathcal{L}((\mathcal{T}_1 \cdot \mathcal{T}_3) \cdot \mathcal{T}_2)$ such that $\|\mu(r), \mu(r_2)\| = 3$ and we have the same case as in $\bar{i} = 0$ with r_2. Thus let us assume that

  $$\mu_F[\mathcal{T}_1](r_1)_0 \cap \mu_F[\mathcal{T}_3](r_3)_0 \neq \emptyset \quad \text{and}$$

  $$\mu_F[\mathcal{T}_1](r_1)_1 \cap \mu_F[\mathcal{T}_3](r_3)_1 = \emptyset$$

  then there exists a $\mu \in \mathcal{L}((\mathcal{T}_1 \cdot \mathcal{T}_3) \cdot \mathcal{T}_2)$ such that $\|r, r_1\| \geq 2$ and we have the same case as in $\bar{i} = 1$.

- $\bar{i} = 3$: again we consider the topology $(\mathcal{T}_1 \cdot \mathcal{T}_3) \cdot \mathcal{T}_2$. As in the preceding case we again consider node r_1 and distinguish between the number of positions i for that

$$\mu_F[\mathcal{T}_1](r_1)_i \cap \mu_F[\mathcal{T}_3](r_3)_i \neq \emptyset$$

. Following the pigeon hole principle one of the above cases always holds for r_1 or for r_2.

which completes the proof. □

Lemma 1.9 *Let $d = 3$ and let $C_1, \ldots C_k$ be the connected components of the graph $G = (T, E)$ with*

$$E = \{ (u, v) \in T \times T \mid \|uv\| \leq 1 \}$$

then

$$\text{cost}(SMT(T)) \geq |T| - 1 + \lceil (k-1)/2 \rceil$$

Proof: First, note that it is sufficient to show that

$$\text{cost}(SMT(T)) \geq |T| - 1 + (k-1)/2$$

since the cost of any Steiner minimum tree is integral.

For a Steiner minimum tree $\mathcal{T}$, let K_i be the number of connected components induced by C_i, i.e., the number of components of the graph with vertex set C_i, where $u, v \in C_i$ are connected by an edge, if there is an edge in $\mathcal{T}$ between these two vertices.

Let $\mathcal{T}$ be a Steiner minimum tree whose edges have length exactly one and such that $\sum_{i=1}^{k} K_i$ is minimal.

We first prove that $K_i = 1$ for each $1 \leq i \leq k$. Assume otherwise and let $u, v \in C_i$ not be connected in the subgraph induced by C_i. Remove an edge st from $\mathcal{T}$ on the path from u to v, such that either s or t is a Steiner point or s and t are from different components (they must exist as u and v are not connected within C_i) and add the edge (u, v) (which has cost 1). The new tree has the same cost as $\mathcal{T}$, the number of components induced by C_i decreases by one and the number of components induced by other C_j remains the same. This is a contradiction to the choice of $\mathcal{T}$.

Let F' be the full components of $\mathcal{T}$ and let F be the non-trivial full components, i.e., the full components having at least one Steiner point. Notice that each full component $f \in F'$ is a Steiner minimum tree of the terminals it spans.

Note that the distance between two terminals in $\text{span}(f)$ is at least two. We first show that if $SMT(f) \geq 3(|\text{span}(f)| - 1)/2$, for all full components $f \in F$ of $\mathcal{T}$, then $\text{cost}(\mathcal{T}) \geq |T| - 1 + (k-1)/2$.

The number of trivial full components is $|T| - k$ as the trivial full components connect all terminals in the same component. Further, $\sum_{f \in F} |\text{span}(f)| = k + |F| - 1$ as a full

component of t terminals connects $t-1$ components. Hence,

$$\begin{aligned}
\text{cost}(T) &= \sum_{f \in F'} \text{cost}(f) \\
&= |T| - k + \sum_{f \in F} \text{cost}(f) \\
&\geq |T| - k + \sum_{f \in F} (3(t(F) - 1)/2) \\
&\geq |T| - k + 3(k-1)/2 + 3(|F| - 1)/2 \\
&\geq |T| - 1 + (k-1)/2
\end{aligned}$$

Now consider a full component f and assume its cost is less than $3(|\text{span}(f)| - 1)/2$. We consider the smallest counterexample, i.e., let T' be a set of strings of length 3 with pairwise distance at least two whose Steiner minimum tree is a full component such that its cost is smaller than $3(|T'| - 1)/2$. Further, assume $|T'|$ is minimal. Note that a Steiner minimum tree for any strict subset of $R \subsetneq T'$ has cost at least $3(|R| - 1)/2$, no matter whether the Steiner minimum tree is a full component or not (as any full component of the Steiner minimum tree of R has less than $|T'|$ terminals).

We assume again that all edges of T' have length exactly one. For $|T'| = 1$ there is nothing to show. Otherwise T' contains a Steiner point s (as all edges have length one and the distance between terminals is at least two). Let $N_i, 1 \leq i \leq 3$, be the set of points adjacent to s whose ith letter differs from the ith letter of s (recall that every point adjacent to s differs from s in exactly one column). Reconnect the sets N_i (by an MST over N_i) with $|N_i| - 1$ edges of length 1 and let T_i be the terminals reachable from N_i. Note that T_i are Steiner trees spanning fewer terminals than T' (they are not necessarily full components). If two of the N_i are empty, one T_i contains all terminals and we have constructed a Steiner tree shorter than T'. If one N_i is empty, say N_3, we have

$$\begin{aligned}
\text{cost}(T') &= \text{cost}(T_1) + \text{cost}(T_2) + 2 \\
&\geq 3 \cdot (|T_1| - 1)/2 + 3(|T_2| - 1)/2 + 2 \geq 3(|T| - 1)/2
\end{aligned}$$

If no N_i is empty, we have

$$\begin{aligned}
\text{cost}(T') &= \text{cost}(T_1) + \text{cost}(T_2) + \text{cost}(T_3) + 3 \\
&\geq 3 \cdot (|T_1| - 1)/2 + 3 \cdot (|T_2| - 1)/2 + 3(|T_3| - 1)/2 + 3 \\
&\geq 3 \cdot (|T| - 1)/2
\end{aligned}$$

which completes the proof. □

k-Dimensional Steiner Minimum Trees

We have seen that it is possible to derive lower bounds for a k-dimensional problem instance I by considering lower bounds for subproblems of I. But there is also a direct way of doing so which is just a generalization of Theorem 1.15. Despite the fact that the bounds obtained by applying this theorem are usually very weak, we will provide it for sake of completeness:

Theorem 1.17 *$|T| - 1$ is a lower bound on the cost of a Steiner minimum tree over T.*

Proof: Let $\mathcal{T} = (T \cup S, E)$ be a Steiner minimum tree over T with Steiner nodes S such that $\sum_{s \in S} \deg(s)$ is minimal. Clearly, there cannot be an edge $\{u, s\} \in E$ with $s \in S$ such that $\|u, s\| = 0$. For seeing this, assume otherwise. Then one could obtain another Steiner minimum tree $\mathcal{T}' = (T \cup S', E')$ by contracting the edge $\{u, s\}$ so that $\sum_{s \in S'} \deg(s) = \sum_{s \in S} \deg(s) - 2$ which contradicts the minimality assumption that we have made for $\mathcal{T}$. Since $\mathcal{T}$ is a tree spanning the points in $T \cup S$, it contains at least $|T \cup S| - 1$ edges – which can easily be seen by a simple induction over the number of edges. In particular, this means that $\mathcal{T}$ has at least $|T| - 1$ edges. But all edges in this tree must have a length of at least 1, because either an edge has two elements in T as endpoints, which have to be unequal or one of the endpoints is in S – recall that we have just shown that in this case the edge cannot have a length of 0. Thus, $|T| - 1$ is a lower bound on the cost of $\mathcal{T}$. □

Note that this bound sometimes improves the single columns discrepancy bound. For example consider again the set

$$T = \{\ (A, A), (A, B), (B, A), (A, B)\ \}$$

Then the sum of the discrepancy bounds is 2 but Theorem 1.17 implies a bound of 3.

1.6.3 Implementation Issues

At this point let us make some remarks on the implementation concerning the computation of the lower bounds.

Cascading Lower Bounds

We have discussed several ways of computing lower bounds. The purpose of providing different methods for determining lower bounds is that they provide a tradeoff between running time and the quality of the computed bound: usually the better the bounds are the more costly it was to calculate them. Thus, we use a cascaded pruning scheme in which we start with low cost lower bounds for proving the non-optimality of a topology and only if this test fails we compute tighter lower bounds at a hight computational cost.

Since a computed lower bound only depends on the terminals that are spanned by a topology we store known lower bounds in a hash table so that we do not have to compute the same lower bounds over and over again.

Maximum Weight Matchings

For solving the dual linear program given in the discussion of Theorem 1.14 we have proposed to compute a lower bound by solving a maximum weight matching. We have carefully compared several implementations for performing this computation, like exact algorithms from the LEDA library [MN99] but also the approximation algorithms and heuristics discussed in the extensive work of Maue and Sanders in [MS07]. One big drawback of their implementations was that they all worked on dynamic graphs,

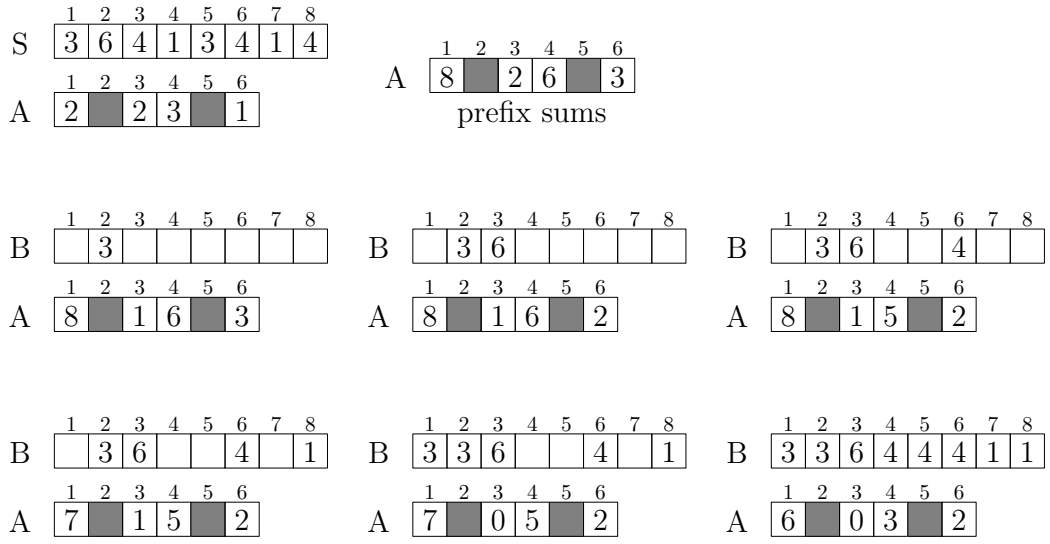


Figure 1.10: An example of the modified counting sort algorithm. The elements in the input sequence S are ordered in such a way that in the output sequence B all identical elements of S are in consecutive positions (see text for further details).

thus putting a large constant to the running times. Therefore, it turned out that our implementation is fastest using a very simple, greedy heuristic on a static graph that we built at the start of our program.

Computing the Subproblems

How fast can we extract a subproblem $\mathcal{S}_T(C)$ out of T for some $C \subseteq \{1, \ldots, d\}$ with $|C| = k$? The difficulty in computing the set $\mathcal{S}_T(C)$ lies in identifying the duplicate sequences in the sequence S of subsequences of T induced by the index set C. The obvious way of doing so is to sort the sequences in S lexicographically using an algorithm like radix sort which runs in $O(|C| \cdot |T| + |C| \cdot |\Sigma|)$ time (see [CLR90]) or a comparison sort algorithm with a running time of $O(\log(|T|) \cdot |T| \cdot |C|)$. But we can do even better by exploiting the fact that sorting the subsequences is a much stronger demand than just finding the duplicate sequences and obtain a running time of $O(|T| \cdot |C|)$ which is clearly optimal since this running time is needed to just can the input.

To achieve this running time we introduce a new algorithm (see Figure 1.10) which is a variation of the well known counting sort algorithm (see [CLR90]): suppose we are given a sequence S of input strings of length 1, we start by allocating an uninitialized array A of size $|\Sigma|$ in constant time. Then we scan through S in $O(|T| \cdot |C|)$ time to set the entries in A corresponding to the elements contained in S to 0. In a second sweep over S we store in A the multiplicities of the occurring elements in A. The next step is the main difference between classical counting sort and our algorithm. In the counting sort algorithm you calculate the prefix-sums in A by scanning the array A

from the smallest index to the highest. In our approach, we do this by scanning the set S once more and determine the prefix-sums according to the order in which the elements appear in S. Then we proceed as usual to construct the output array. Note that we never scan over the array A, but only a constant time over the input sequence S. That is, the total running time for this procedure is in $O(|T|)$. Even though the resulting strings are not in particular order, all identical sequences are now consecutive, yielding the following result:

Lemma 1.10 *The set $\mathcal{S}_T(C)$ can be computed in $O(|T| \cdot |C|)$ time.*

Proof: The idea is to generalize the algorithm given above for finding duplicate sequences in one dimensional strings to strings of length $|C|$ by using the radix sort algorithm (see [CLR90]). For each $i \in C$ we execute the modified counting sort algorithm on the ith elements of S. Since the modified counting sort algorithm is stable, the sequences in S are ordered in such a way that identical sequences are in consecutive positions, so that we can identify and eliminate duplicate sequences in S in time $O(|T|)$. Thus in total the running time for computing $\mathcal{S}_T(C)$ is given by $O(|C| \cdot |T|)$. □

Computing MST Lower Bounds

One of the most important questions concerning the running time of our algorithm is the question how to compute the cost of a Steiner minimum tree in two dimension efficiently. In Theorem 1.16 we have shown that this is equivalent of computing the cost of a minimum spanning tree over the two dimensional problem instance. Since we have to compute $O(d^2)$ such MSTs to obtain one lower bound for setting up the linear program in the lower bound computation via the ILP formulation, it cannot be overrated to speed up this calculation as much as possible. Exploiting the special structure of the problem we can improve the running times achieved by standard algorithms like the ones of Prim and Kruskal by far.

Lemma 1.11 *Given some subset $C \subseteq \{1, \ldots, d\}$ with $|C| = 2$. Then*

$$\text{cost}(\text{MST}(\mathcal{S}_T(C)))$$

can be computed in $O(|T|)$ time.

Proof: Since $\mathcal{S}_T(C)$ is a set of two-dimensional strings we know that all pairwise distances of elements in this set are either 1 or 2. Consider now the graph $G = (\mathcal{S}_T(C), E)$ with $E := \{ (u, v) \mid \|u, v\| \leq 1 \}$ and let $(c_i)_{i \in \{1,\ldots,l\}}$ be the connected components of G. Then

$$\begin{aligned} \text{cost}(\text{MST}(\mathcal{S}_T(C))) &= 2 \cdot (l-1) + \sum_{i=1}^{l} (|c_i| - 1) \\ &= c + |\mathcal{S}_T(C)| - 2 \end{aligned}$$

That is, to compute the size of the minimum spanning tree it is sufficient to know how many connected components the graph G has. The idea of the algorithm is now to determine this number in two steps (see Figure 1.11 for an example). In the first one we

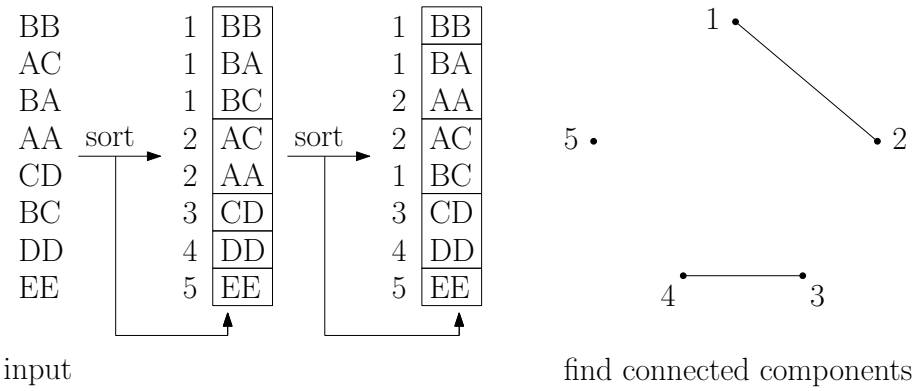


Figure 1.11: Proof illustration for Lemma 1.11.

use the counting sort variant of the preceding section to sort the sequences in $\mathcal{S}_T(C)$ according to the first element, that is we obtain a partitioning of the elements such that each part contains elements with the same first element which in turn means that the pairwise distances between them is exactly 1 - recall that there are no duplicate sequences. For each sequence we store in which part of the partition it was contained. In the second step we sort the sequences – again using counting sort – according to their second element. Again we obtain a partition of the sequences. If now in one part there are sequences of different parts in the first partition we know that the sequences in these parts must belong to the same connect component. That is, we can compute the number of connected components of G by determining the number of connected components in the graph G' that has a node for each part in the first partition and an edge (u, v) if there are two consecutive sequences in a part of the second partition which belong two different parts in the first partition. Since G' has not more than $|T|$ nodes and not more than $|T|$ edges this can be done by a simple depth-first search in time $O(|T|)$ time. Since each counting sort can be computed in time linear in $|T|$ following Lemma 1.10, the total running time is bounded by $O(|T|)$. □

Inner Preprocessing

Note that whenever we want to compute a lower bound for a subproblem $\mathcal{S}_T(C)$ we can apply the preprocessing methods that we have discussed earlier on this set, before we actually compute the lower bounds.

1.7 Experiments

1.7.1 Running Times

We compare the C++ implementation of our pruning algorithm with two famous phylogeny programs from the public domain, MEGA in version 4 (see [KTND07]) and PHYLIP in version 3.68 (see [Fel04]), and the commercial program PAUP in version 4.0 (see [Swo03]). These programs represent very fast implementations of variations

and extensions of Hendy/Penny's branch and bound method. While PHYLIP is probably the most cited phylogeny package, MEGA and PAUP can be considered to be the fastest available programs for computing the maximum parsimony problem. As global upper bounds we use a heuristic as implemented in the PHYLIP package that can be computed within a few seconds.

Since the choice of the test data is very crucial for the results, we decided not to create random instances by our own. Instead we use real life test data and generated data sets from a public source. All tests have been performed on a 1.6 GHz Pentium M machine equipped with 2 GByte of RAM.

- **real life data:** As real life data we use genetic sequences of RNA viruses discussed in [CGH03]. Since these data sets vary much in size and complexity, they reflect a wide spectrum of different difficulty levels in computing a maximum parsimony tree. Since our implementation does not support unknown or partially known nucleotides at the moment, we only use data sets that do not contain such nucleotides. Table 1.3 shows a comparison of the running times of these problems using our pruning approach, MEGA, PHYLIP and PAUP to solve them. The given running times are the CPU-times in seconds consumed by the solvers and include also the time needed to read and parse the data files. The lower part of the table lists the problems that could not be solved by any of the programs within 24 hours.

- **generated data:** The results of the real life data show clearly the superiority of the pruning approach, especially when the problems become more complex. To examine this behavior we created several instances with the tool Seq-Gen [RG97] that simulates the evolution of nucleotide or amino acid sequences along a phylogeny for a given evolution model. The instances that we created have been built on two trees (see figure 1.7.3) with 24 leaves that were chosen randomly from a huge set of trees created by Vincent Ranwez [RG01]. We tested different evolutionary models. As the relative running times didn't depend much on the model, we resign to give details. . In Table 1.2 we compare the running times of MEGA, PAUP and our algorithm. The numbers in the data set names describe branch length scaling factors that can be seen as a measure for the evolutionary distance between the species. With $|T_{pp}|$ we denote the number of terminals after the initial preprocessing steps described in section 1.5. The columns *spts* and *spts*(%) give the absolute number of survived partial topologies and the percentage of all possible partial topologies of sizes at most $\lfloor |T_{pp}|/2 \rfloor$. Furthermore, we list the optimal solutions *opt* together with 2-column lower bounds lb_{mst} using exact matchings and the lower bounds $lb_{Holland}$ using Holland-et.al.'s technique.

As a result one can see that our approach clearly outperforms the approach of Hendy and Penny in practice as soon as the sequences in the data sets become more and more uncorrelated. In our opinion this is due to the fact that the performance of the Hendy/Penny-approach relies only on one simple lower bound based pruning test, while our approach makes use of additional structural informations. As an example for the strength of our tests, consider the strong lower bound results that even outperform existing lower bound techniques that could not have been used in the Hendy/Penny-approach.

1.7.2 Lower Bounds

As stated in section 1.6 we have improved the lower bounding technique by [HHPM05]. To show how strong this improvement is in practice we have compared our lower bounding method with the one of Holland-et.al. on the real life data set. Tables 1.2 and 1.4 show clearly that we can significantly lower the gap between the optimal values and the lower bounds. In several cases it was even possible to close this gap completely. Note that we have used only the lower bounds based on subproblems of dimension 2. Even better results would have been possible using our lower bounds for 3-dimensional subproblems, but involving a significant increase in the running time - even though feasible.

1.7.3 Preprocessing

As described in section 1.5 we use two preprocessing techniques. The first one is a straightforward generalization of the concept of parsimony informative sites. The second one is the so called singleton separation method. Both have been discussed in the pruning section. In table 1.1 we show for some instances of the real world examples how both preprocessing techniques and their combination can significantly decrease the input size.

1.8 Conclusion

We have presented a new approach for solving the maximum parsimony problem exactly. It outperforms all previous methods, especially when the sequences in the data sets become more and more uncorrelated. As we have shown in the experimental section, this closes a gap since the Hendy/Penny approach can only work efficiently when the evolutionary distances between the sequences are small.

Still we see various ways of improving the algorithm. Clearly, the algorithm can be improved by finding further properties that exclude partial topologies from being part of an Steiner minimum tree that can be tested efficiently or by finding new ways of preprocessing the data.

One major key for the efficiency of our approach is the computation of tight lower bounds . These bounds cannot be used in the old approach as their applicability heavily depends on the fact that the terminals not spanned by the current topology are all connected to the root of the topology.

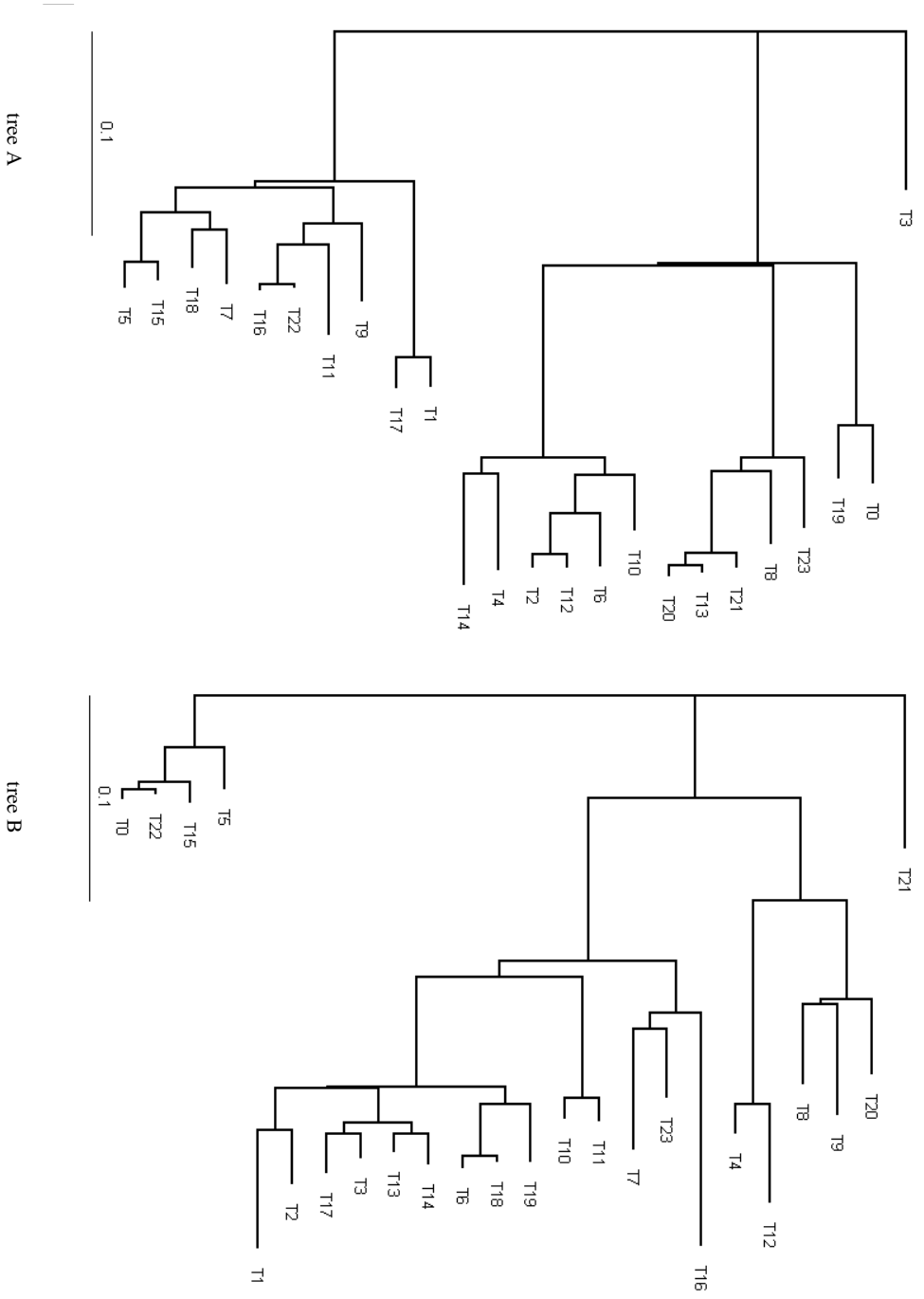


Figure 1.12: The input trees A and B for the Seq-Gen data.

Family	Species	Gene	Seq.	Survived Sequences IPI	SSP	IPI/SSP
Bunyaviridae	*Akabane virus*	NP	26	14	25	14
	Oropouche virus	NP	28	23	28	23
	Puumala virus	G2	27	20	27	17
	Tomato spotted wilt virus	N	16	14	16	13
Paramyxoviridae	*Avian pneumovirus*	M	21	15	21	15
		P	15	11	14	11
		F	14	10	12	9
		N	14	5	6	5
	Bovine respiratory syncytial virus	G	24	17	24	16
	Canine distemper virus	H	31	30	30	29
	Human respiratory syncytial virus A	N	24	16	24	14
	Newcastle disease virus	N	27	25	27	24
Rhabdoviridae	*Infectious hematopoietic virus*	G	19	18	19	17
	Vesicular stomatitis Indiana virus	G	26	17	26	15
	Vesicular stomatitis virus	G	25	23	22	22

Table 1.1: The effect of the preprocessing methods. Here "IPI" stands for the iterated parsimony informative test and "SSP" for singleton separation, both described in the pruning section.

Tree	Data Set	$\|T\|$	$\|T_{pp}\|$	opt	lb_{mst}	$lb_{Holland}$	spts	spts(%)	Running Times (s) pruning	MEGA	PAUP
A	Seq-Gen 50	24	23	885	885	852	46	$5 \cdot 10^{-14}$	0.32	0.51	0.19
	Seq-Gen 100	24	23	1585	1580	1465	67	$1.4 \cdot 10^{-13}$	1.48	4.21	6.98
	Seq-Gen 150	24	24	2188	2143	1948	691	$7.5 \cdot 10^{-13}$	10.30	717.40	1379.05
	Seq-Gen 200	24	24	2560	2473	2257	2703	$2.9 \cdot 10^{-12}$	38.64	10813.80	25663.81
	Seq-Gen 250	24	24	2894	2750	2512	11165	$2.9 \cdot 10^{-13}$	309.65	–	–
	Seq-Gen 300	24	24	3124	2921	2672	57536	$1.5 \cdot 10^{-12}$	4341.93	–	–
B	Seq-Gen 50	24	24	936	934	887	80	$2.1 \cdot 10^{-15}$	1.41	1.44	1.48
	Seq-Gen 100	24	24	1682	1673	1542	141	$3.6 \cdot 10^{-15}$	4.52	92.37	106.98
	Seq-Gen 150	24	24	2282	2238	2025	1071	$2.8 \cdot 10^{-14}$	38.61	11829.37	17313.34
	Seq-Gen 200	24	24	2692	2582	2329	7552	$1.9 \cdot 10^{-13}$	378.74	–	–
	Seq-Gen 250	24	24	3124	2939	2660	41985	$1.1 \cdot 10^{-12}$	4956.58	–	–

Table 1.2: **A running time comparison - generated data:** "-": interrupted after 6 hours of computation.

Table 1.3: **A running time comparison - real life data:** Entries with a "–" sign mark an interrupted call of the corresponding program. We interrupted a program when it consumed more than 2 GByte of RAM or when its running time exceeded a certain threshold that depends on the complexity of the instance: $-^1$: terminated after 6 hours of computation and a progress of less than 3% (stated by the program); $-^2$: process allocated more than 2 GByte of RAM after 1.5 hours of computation; $-^3$: process was terminated after 3 hours of computation: $-^4$: process was interrupted after 24 hours of computation.

			Running Times (s)			
Family/Species	Gene	Seq.	pruning	MEGA	PAUP	phylip
Bornaviridae						
Borna disease virus	G	17	0.02	0.80	0.09	37.75
	M	19	0.01	2.03	0.12	142.68
	NP	17	0.03	1.03	0.07	463.45
Bunyaviridae						
Akabane virus	NP	26	0.02	911.43	54.60	$-^3$
Cache Valley Virus	G	14	0.18	1.36	0.11	64.56
Crimean-Congo hemorrhagic fever virus	G	11	1.06	0.70	0.13	0.28
Dobrava virus	N	23	2.43	4.03	0.22	357.51
Hantaan virus	G1	15	0.38	0.76	0.08	1.92
	N	20	1.05	0.78	0.15	294.09
Oropouche virus	NP	28	19.43	11256.00	84.55	$-^3$
Puumala virus	G1	14	1.75	0.87	0.15	10.92
	G2	27	3.34	242.38	3.49	1066.42
Rift Valley Fever Virus	G2	19	0.08	0.79	0.10	299.51
Tomato spotted wilt virus	N	16	0.01	0.67	0.05	24.07
	NSS	10	0.01	0.88	0.04	0.11
Orthomyxoviridae						
Influenza B virus	HA	21	0.02	1.13	0.09	21.47
	M	26	0.2	13.56	0.79	$-^3$
	NA	23	0.84	21.03	0.79	$-^4$

Family/Species	Gene	Sequ.	Running Times (s) pruning	MEGA	PAUP	phylip
	NP	28	2.53	10.15	0.24	–[4]
	PA	19	0.10	0.64	0.11	109.18
Influenza C virus	CM2	20	0.01	0.69	0.06	56.62
	NP	28	0.18	13.67	0.53	–[1]
Paramyxoviridae						
Avian pneumovirus	M	21	0.02	16442.00	19.74	2038.71
	P	15	0.01	1.21	0.05	34.31
	F	14	0.01	4.29	0.36	72.31
	N	14	< 0.01	50.83	2.69	34.28
Bovine respiratory syncytial virus	G	24	0.06	0.81	0.07	640.78
Canine distemper virus	H	31	588.83	–[1]	15949.47	–[1]
Human parainfluenza virus 3	HN	13	0.03	0.72	0.05	10.33
Human respiratory syncytial virus A	N	24	0.11	2.63	0.18	–[3]
Human respiratory syncytial virus	F	22	0.45	1.32	0.3	1082.48
Measles virus	F	22	0.32	28.94	0.71	–[3]
	HA	20	0.05	0.77	0.09	205.36
	L	27	2.55	13.06	0.93	–[3]
Mumps virus	F	29	20.89	26932.52	0.05	–[4]
	L	11	0.01	0.7	0.15	0.11
Newcastle disease virus	M	32	20531.41	–[4]	–[4]	–[4]
	N	27	155.11	–[1]	–[4]	–[1]
Rhabdoviridae						
Infectious hematopoietic virus	G	19	0.06	0.86	0.07	145.38
Vesicular stomatitis Indiana virus	G	26	0.44	–[2]	–[4]	–[2]
Vesicular stomatitis virus	G	25	0.88	1.0	0.14	4877.56
Viral hemorrhagic septicaemia virus	G	15	0.03	0.33	0.07	91.16

Family/Species	Gene	Sequ.	Running Times (s) pruning	MEGA	PAUP	phylip
	N	10	< 0.01	0.29	0.04	0.13
Unclassified						
Rice stripe Virus	JN	11	< 0.01	0.66	0.02	0.40
Arenaviridae						
Junin virus	NP	45	–	–	–	–
Lassa virus	NP	59	–	–	–	–
Bunyaviridae						
Puumala virus	NP	48	–	–	–	–
Paramyxoviridae						
Measles virus	N	135	–	–	–	–
Newcastle disease virus	P	28	–	–	–	

Table 1.4: **A lower bound comparison of the technique by Hollan et al. and the minimum spanning tree based technique on the real life data set**

			Lower Bounds		Lower Bounds (%)	
Family/Species	Gene	opt	*MST*	*Holland*	*MST*	*Holland*
Bornaviridae						
Borna disease virus	G	434	426	398	98.16	91.71
	M	90	87	81	96.67	90.00
	NP	289	274	253	94.81	87.54
Bunyaviridae						
Akabane virus	NP	94	92	81	97.87	86.17
Cache Valley Virus	G	409	385	351	94.13	85.82
Crimean-Congo hemorrhagic fever virus	G	2018	2017	1827	99.95	90.53
Dobrava virus	N	250	225	188	90.00	75.20
Hantaan virus	G1	1089	1055	915	96.88	84.02
	N	809	704	592	87.02	73.18
Oropouche virus	NP	127	111	97	87.40	76.38
Puumala virus	G1	1773	1584	1322	89.34	74.56
	G2	661	556	465	84.11	70.35
Rift Valley Fever Virus	G2	112	104	91	92.86	81.25
Tomato spotted wilt virus	N	85	84	79	98.82	92.94
	NSS	168	167	159	99.40	94.64
Orthomyxoviridae						
Influenza B virus	HA	247	247	232	**100.00**	93.93
	M	125	121	107	96.80	85.60
	NA	364	347	298	95.33	81.87
	NP	332	318	279	95.78	84.04
	PA	350	350	321	**100.00**	91.71
Influenza C virus	CM2	66	65	61	98.48	92.42
	NP	96	92	77	95.83	80.21
Paramyxoviridae						
Avian pneumovirus	M	83	81	74	97.59	89.16
	P	109	108	103	99.08	94.50
	F	106	106	102	**100.00**	96.22
	N	57	57	56	**100.00**	98.25
Bovine respiratory syncytial virus	G	232	228	204	98.28	87.93

Family/Species	Gene	opt	Lower Bounds *MST*	Lower Bounds *Holland*	Lower Bounds (%) *MST*	Lower Bounds (%) *Holland*
Canine distemper virus	H	769	693	598	90.12	77.76
Human parainfluenza virus 3	HN	222	217	194	97.75	87.39
Human respiratory syncytial virus A	N	496	492	442	99.19	89.11
Human respiratory syncytial virus	F	455	430	388	94.51	85.27
Measles virus	F	261	254	231	97.32	88.51
	HA	278	275	256	98.92	92.09
	L	675	660	614	97.77	90.96
Mumps virus	F	433	403	343	93.07	79.22
	L	517	517	515	**100.00**	99.61
Newcastle disease virus	M	882	720	609	81.63	69.05
	N	1017	792	657	77.88	64.60
Rhabdoviridae						
Infectious hematopoietic virus	G	171	166	151	97.08	88.30
Vesicular stomatitis Indiana virus	G	471	466	428	98.94	90.87
Vesicular stomatitis virus	G	671	656	577	97.76	85.99
Viral hemorrhagic septicaemia virus	G	198	190	170	95.96	85.85
	N	213	211	201	99.06	94.37
Unclassified						
Rice stripe Virus	JN	64	64	63	**100.00**	98.44

1.9 An Extension to Recombination Networks

As we have seen in before, a fundamental class of problems in Computational Biology deals with the reconstruction of phylogenetic trees. In order to build such a tree we compare specific features of the species under the natural assumption that species with similar features are closely related. In modern phylogeny these features are defined by DNA or protein sequences.

But there are situations in which trees do not reflect the biological reality. Under certain conditions evolutionary events can occur that cause horizontal transfer of genetic data. Ancestral relationships that evolve when these *recombination events* have occurred are best described by a network rather than by a simple tree. There has been active research in automatically computing these phylogenetic networks from sequence data, see e.g. [JNST07, JNST05, NJZMC05, Hei93, GEL03, HK05, WZZ01, KG98]. In Section 1.9.2, we discuss in detail existing approaches and relate our work to them. For more information on the reconstruction of phylogenetic networks, we refer to [MKL06, PCH02]. For a list of available tools see [WEBb] or [WEBc].

In this section we again assume that we are given a set T of aligned sequences, that is a set of sequences of length d. Our aim is to construct an optimal ancestral relationship for a given number of recombination events under a generalization of the maximum parsimony criterion.It slightly differs from those in [JNST07, JNST05, NJZMC05, Hei93, GEL03]. We recommend to use our method when recombination events are assumed to happen rarely. Part of this work was published in [AN08].

To test our model that we will describe in more detail in the following, we give in Section 3 an exact algorithm that computes the optimal phylogenetic network with one recombination and show in Section 1.9.4 that the algorithm computes the correct phylogenetic network even for sequences with (still reasonable) long evolutionary distances. Our experiments show that we can compute the correct phylogenetic network for almost the same data sets where the maximum parsimony criterion is able to reconstruct the correct phylogenetic tree when it is given the subsequences that correspond to a specific tree in the evolution.

1.9.1 Model

In contrast to phylogenetic trees as considered in the previous chapter we allow a few nodes to have indegree two. We refer to these nodes as *recombination nodes*. In order to have a valid ancestral relationship, we require the network to be acyclic and we assume that the network is rooted. In these terms we are equivalent to the networks under consideration in [JNST07, Hei93, GEL03]. We call such a network over T with exactly k recombination nodes a *k-recombination network for T*.

The goal is now to find among all possible k-recombination networks for T a network of smallest cost. The difference between our model and the other models considered in the literature is how we evaluate a given network, that is how we determine its cost. For defining this cost, we extend the parsimony criterion by defining a cost for recombination events. Again, we assume that leaves of the network are exactly the terminal sequences in T. We want to find sequences for the inner nodes that minimize the cost of the k-recombination topology. In our case the cost function is easiest described by charging the cost to the nodes instead of charging them to the

edges what we did in the maximum parsimony problem. The cost of the root node is 0. The cost of a non-recombination node u equals to the Hamming distance of u and its father node. On the other hand, the cost of a recombination node u is now defined as

$$\min_{p\in\{0,1\}^d}\left(\sum_{i=1}^{d}(p_i\|u_i,t_i^1\|+(1-p_i)\|u_i,t_i^2\|)+\alpha\cdot|\{1\le i<d\mid p_i\neq p_{i+1}\}|\right)$$

where α is some non-negative constant and t^1 and t^2 are two ancestors of u. Intuitively speaking, the recombinant sequence can have the genetic code of either of its ancestors, paying a certain cost for switching between the two ancestor. This cost reflects the fact that recombination does not happen by randomly choosing genetic code from the two sequences, but it is assumed that there are only a few *jumps*, where the source for the recombinant sequence changes. Note, that same definition of the cost for a recombination node was used by Maydt and Lengauer in [ML06], where the authors try to explain a given sequence by recombinations of other input sequences. Since the cost of a non-recombination node is exactly the cost of the in-going edge in the maximum parsimony model, we sometimes refer to the cost of an edge if it is clear that the corresponding node is a non-recombination node. The cost of a k-recombination network is then just the sum of the costs of its nodes.

In Section 1.9.3, we show that the cost of a 1-recombination topology as well as sequences for the inner nodes can be computed quite efficiently. As computing the tree with maximum parsimony is already NP-hard, computing the optimal k-recombination topology is NP-hard for all k, as choosing some big α will lead to the fact that the cost of the recombination node equals the Hamming distance to one of its ancestors.

1.9.2 Related Work

There are many tools for detecting recombination events in sequence data. Posada et al. [PCH02] divide these tools into five classes: similarity methods, distance methods, phylogenetic methods, compatibility methods and substitution distribution methods. Most methods only try to detect the existence of a recombination event. Moreover it is frequently required that the recombinant sequence is contained in the data set. The only exceptions are phylogenetic methods. Most of these methods detect a recombination by inferring different phylogenetic trees in different parts of the data, that is, they analyze the data by a sliding window approach obtaining different trees and by trying to combine these trees to a prediction. A list of available tools can be found at [WEBb].

Posada et al. [PCH02] conclude that most methods have trouble detecting rare recombination events, especially when sequence divergence is low, that is exactly the case where our algorithm is aimed for.

Huson and Kloepper [HK05], Wang et al. [WZZ01], Gusfield et al. [GEL03] and Kececioglu and Gusfield [KG98] use an extension of the maximum parsimony criterion originally proposed by Hein [Hei93]. The essential difference is that they allow only one jump when constructing a sequence from its two ancestors. Our algorithm can easily be adopted to find the optimal phylogenetic network in this model. Their algorithm does not compute an optimal phylogenetic network in our sense, but it explains the data under a minimal number of recombinations under the infinite state assumption, that is, it is assumed that no back-exchange of genetic code occurred. As back-exchange

is assumed to happen in real evolution, their algorithm typically overestimates the number of recombination events.

The extension of the maximum parsimony criterion proposed by Jin and others in [JNST07, JNST05, NJZMC05] basically corresponds our model with $\alpha = 0$. As shown in our experiments in Section 1.9.4, the prediction of the topology is much less accurate for this model. Furthermore, their algorithm does not compute an optimal phylogenetic network and only works if the starting tree is contained in the phylogenetic network. As observed by Ruths and Nakhleh [RN05], phylogenetic tree algorithms fail to reconstruct data when recombination has occurred. Thus, it is unclear how to obtain a phylogenetic tree to start with.

1.9.3 The Algorithm

As noted before the k-Recombination Phylogeny Network (k-RPN) problem is NP-hard. Thus, unless $P = NP$, there is no polynomial time algorithm for solving it exactly. As in the case of the Steiner minimum tree problem, the k-RPN problem can be solved for small input instances by enumerating the set of possible solutions and by determining among them the one which minimizes the total network length. In our algorithm we focus on the case where $k = 1$. We call the subtree of the recombination network rooted at the recombination node the *recombinant*.

Preliminaries

As we have done in the previous chapter we can use Theorem 1.1 to assume without loss of generality that an optimal Steiner minimum tree is given in standard form with respect to T. Furthermore, we refer to Section 1.3 for the definitions and basic theorem that we will use in the following.

Evaluation of a Recombination Network

Let us first explain how one can compute an optimal sequence for the recombination node. Let t^1 and t^2 be the sequences of the ancestors of the recombination node r, the question is equivalent of finding a vector $p \in \{0,1\}$ such that

$$p = \operatorname*{argmin}_{p' \in \{0,1\}^d} \left(\sum_{i=1}^{d} (p'_i \| u_i, t_i^1 \| + (1 - p'_i) \| u_i, t_i^2 \|) + \alpha \cdot |\{1 \leq i < d \mid p'_i \neq p'_{i+1}\}| \right)$$

But this can be done in $O(d)$ time by the simple dynamic program given in Algorithm 2 whose correctness can easily be seen by a simple induction over $i \in \{1, \ldots, d\}$. We call this operation the $\diamond$-operator. Once we have computed such a p we can also compute in time $O(d)$ the above expression to obtain the cost of the recombination node.

Note that it is trivial to extend this algorithm to range points by just replacing the equality checks for the elements in the sequences by checking whether the elements are non-disjoint. That is, given three range points instead of sequences one can compute in $O(d)$ time the minimal cost of the recombination node over all sequences contained in the range points.

Enumeration Process

The key observation for the enumeration process is now that any 1-recombination network is composed of two binary trees. Formally, if we are given a 1-recombination network $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ over T then there are two trees $\mathcal{R}$ and and $\mathcal{T}'$ such that

$$\begin{aligned} V_\mathcal{T} &= V_{\mathcal{T}'} \cup V_\mathcal{R} \cup \{r_1, r_2\} \\ E_\mathcal{T} &= E_{\mathcal{T}'} \cup E_\mathcal{R} \setminus \\ &\quad \{\{u_1, v_1\}, \{u_2, v_2\}\} \cup \\ &\quad \{\{u_1, r_1\}, \{v_1, r_1\}, \{u_2, r_2\}, \{v_2, r_2\}\} \cup \\ &\quad \{\{\mathrm{root}(\mathcal{R}), r_1\}, \{\mathrm{root}(\mathcal{R}), r_2\} \end{aligned}$$

for two edges $\{u_1, v_1\}$ and $\{u_1, v_1\}$ in $E_{\mathcal{T}'}$. $\mathcal{R}$ then corresponds to the recombinant and $\mathrm{root}(\mathcal{R})$ to the recombination node. Let us write $\mathcal{T} = \triangle_{e_1,e_2}(\mathcal{T}', \mathcal{R})$. Hence one can reduce the enumeration of recombination networks to the enumeration of binary trees.

As we did before, instead of enumerating all binary trees we enumerate topologies over some subsets of T. Let us extend the definition of a topology given in Section 1.3 for this by replacing the constraint that the underlying graph is a tree by the constraint that the the underlying graph is connected. The function $\triangle_{e_1,e_2}(\cdot,\cdot)$ transfers from trees to topologies in the obvious way.

Doing so, we have to discuss how we can find labelings for the inner nodes of a network composed of two topologies in a way such that the resulting 1-recombination network has minimal cost. The crucial observation therefor is the following.

Let $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ be an optimal 1-recombination network with $\mathcal{T} = \triangle_{e_1,e_2}(\mathcal{T}', \mathcal{R})$ for $e_1, e_2 \in E_\mathcal{T}$ and with $\mathcal{T}' = (V_{\mathcal{T}'}, E_{\mathcal{T}'})$. Then,

$$\begin{aligned} \mathrm{cost}(\mathcal{T}) &= \sum_{v \in V_\mathcal{T}} \mathrm{cost}(v) \\ &= \sum_{v \in V_\mathcal{T} \setminus \mathrm{root}(\mathcal{R})} \mathrm{cost}(v) + \mathrm{cost}(\mathrm{root}(\mathcal{R})) \\ &= \sum_{v \in V_\mathcal{T} \setminus \mathrm{root}(\mathcal{R})} \sum_{i \in p_1} \mathrm{cost}_i(v) + \sum_{v \in V_\mathcal{T} \setminus \mathrm{root}(\mathcal{R})} \sum_{i \in p_2} \mathrm{cost}_i(v) + \mathrm{cost}(\mathrm{root}(\mathcal{R})) \\ &= \sum_{v \in V_\mathcal{T} \setminus \mathrm{root}(\mathcal{R})} \sum_{i \in p_1} \mathrm{cost}_i(v) + \sum_{i \in p_1} \|r_i, (r_1)_i\| + \\ &\quad \sum_{v \in V_\mathcal{T} \setminus \mathrm{root}(\mathcal{R})} \sum_{i \in p_2} \mathrm{cost}_i(v) + \sum_{i \in p_2} \|r_i, (r_2)_i\| + \alpha \cdot j \\ &= \sum_{\{u,v\} \in E_\mathcal{T} \setminus \{\{r,r_1\},\{r,r_2\}\}} \sum_{i \in p_1} \|u_i, v_i\| + \sum_{i \in p_1} \|r_i, (r_1)_i\| + \\ &\quad \sum_{\{u,v\} \in E_\mathcal{T} \setminus \{\{r,r_1\},\{r,r_2\}\}} \sum_{i \in p_2} \|u_i, v_i\| + \sum_{i \in p_2} \|r_i, (r_2)_i\| + \alpha \cdot j \\ &= \sum_{e \in E_\mathcal{T} \setminus \{\{r,r_2\}\}} \mathrm{cost}_{p_1}(e) + \sum_{e \in E_\mathcal{T} \setminus \{\{r,r_1\}\}} \mathrm{cost}_{p_2}(e) + \alpha \cdot j \end{aligned}$$

where (p_1, p_2) is the partitioning of $\{1, \ldots, d\}$ obtained by the $\diamond$-operator for the recombination node root($\mathcal{R}$) and where

$$j = |\{\, i \in \{1, \ldots, d\} \mid i \text{ and } i+1 \text{ are contained in different } p_i \,\}|$$

Thus we can simply consider the two rooted topologies $\mathcal{T}_1 = (V_\mathcal{T}, E_\mathcal{T} \setminus \{r, r_1\})$ and $\mathcal{T}_2 = (V_\mathcal{T}, E_\mathcal{T} \setminus \{r, r_2\})$. We re-root each $\mathcal{T}_i$ such that the node r_i becomes the root. Note that Lemma 1.4 implies that for all $j \in \{1, \ldots, d\}$, the $\mu_F[\mathcal{T}_i](r_i)_l$ are maximal. Then we use the $\diamond$-operator to determine an optimal labeling for root($\mathcal{R}$). This way we obtain an optimal labeling for $\mathcal{T}$.

The enumeration process itself works now in almost the same way as it does for the Steiner minimum tree problem discussed earlier (see Algorithm 3). We start with the set X consisting only of the topologies of size 1 namely the terminals itself. We then inductively construct all topologies of size k by combining the ones out of X of size less than k. We do this until all topologies of size $|T| - 1$ are built. In section 1.9.3 we will see why it is sufficient to enumerate only topologies up to a size of $|N| - 1$.

Cutting down the search space works now as follows: In each step when a topology $\mathcal{T}$ was built, we run several pruning tests to decide whether an optimal solution for the problem can contain $\mathcal{T}$ as a subtopology without violating optimality. In the following subsection we will discuss these tests.

Algorithm 2 $\diamond$-operator: $\diamond(t^1, t^2, u)$

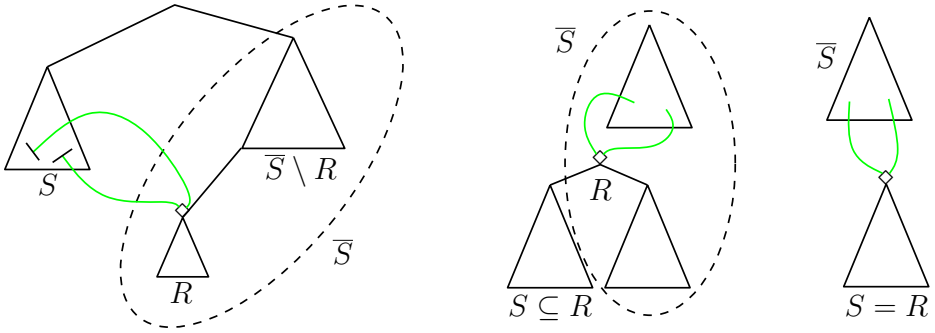
$OPT' := 0; \quad OPT'' := 0$
$p', p'' \in \{0,1\}^d$

for $i = 1 \ldots d$ **do**
  $tent' := 0; \quad tent'' := 0$
  if $t_i^1 = u_i$ **then**
    $tent' = \min(OPT', OPT'' + \alpha); \quad p'_i = 1$
  else
    $tent' = \min(OPT', OPT'' + \alpha) + 1; \quad p'_i = 0$
  end if
  if $t_i^2 = u_i$ **then**
    $tent'' = \min(OPT' + \alpha, OPT''); \quad p''_i = 1$
  else
    $tent'' = \min(OPT' + \alpha, OPT'') + 1; \quad p''_i = 0$
  end if
  $OPT' = tent'; \quad OPT'' = tent''$
end for

if $OPT' < OPT''$ **then**
  return p'
else
  return p''
end if

Algorithm 3 enumeration process

$X := T$
for $i = 2 \dots |T| - 1$ **do**
  for all $\mathcal{T}_k,\ \mathcal{T}_j \in X$ **do**
    if $\text{span}(\mathcal{T}_k) \cap \text{span}(\mathcal{T}_l) = \emptyset$ and
      $|\text{span}(\mathcal{T}_k)| + |\text{span}(\mathcal{T}_l)| = i$ **then**
      if $\neg$ prunable($\mathcal{T}_k \cdot \mathcal{T}_l$) **then**
        $X = X \cup \{\mathcal{T}_k \cdot \mathcal{T}_l\}$
      end if
    end if
  end for
end for



1. case 2. case 3. case

Figure 1.13: The three different cases occurring in the lower bound pruning step. The green edges represent the incoming edges of the $\diamond$-operator.

Pruning the Search Space

For pruning the search space we use very similar ideas as we have used for the Steiner minimum tree problem, thus we restrict the discussion to the main ideas behind these tests. Let us denote by $lb_{\text{SMT}}(\cdot)$ a lower bound on the cost of a Steiner minimum tree as constructed in Section 1.6. Furthermore, we write $ub_{\text{RPN}}(\cdot)$ for an upper bound on the cost of an optimal solution for the 1-recombination network problem.

lower bound test: You are given a topology T_S spanning a terminal set $S \neq T$. Let $\overline{S} := T \setminus S$. Consider now an optimal solution with recombinant T_R spanning the terminal set R. We have to discuss several cases (see Figure 1.13):

1. $S \cap R = \emptyset$: We can prune T_S if $\text{cost}(T_S) + lb_{\text{SMT}}(\overline{S}) - \text{bnsd}(\overline{S}) > ub_{\text{RPN}}(T)$.
   Proof: any solution with T_S as a subtopology clearly has a cost of at least $\text{cost}(T_S) + lb_{\text{SMT}}(R) + lb_{\text{SMT}}(\overline{S} \setminus R)$. Since we do not know R in an optimal solution we have to lower bound the term $lb_{\text{SMT}}(R) + lb_{\text{SMT}}(\overline{S} \setminus R)$. Clearly $lb_{\text{SMT}}(R) + lb_{\text{SMT}}(\overline{S} \setminus R) \geq lb_{\text{SMT}}(\overline{S}) - \text{bnsd}(\overline{S})$.

2. $S \subsetneq R$: T_S can be pruned if $\text{cost}(T_S) + lb_{\text{SMT}}(\overline{S}) - \text{bnsd}(\overline{S}) > ub_{\text{RPN}}(T)$.

3. $S = R$: T can be pruned if $\text{cost}(T_S) + lb_{\text{SMT}}(\overline{S}) > ub_{\text{RPN}}(T)$.

Note that the last cases follow the same arguments as the first case does. Since we do not know in advance which case occurs in an optimal solution we have to take the weakest condition

$$\text{cost}(T_S) + lb_{\text{SMT}}(\overline{S}) - \text{bnsd}(\overline{S}) > ub_{\text{RPN}}(T)$$

Recombination Phase

The last part of our algorithm is the recombination phase in which we construct the recombination networks by combining the enumerated topologies:

Algorithm 4 recombination process

for all $\mathcal{T} \in X$ with: $|\text{span}(\mathcal{T})| \in \{3, .., |T| - 1\}$ **do**
  for all $\mathcal{R} \in X$ with:
  $|\text{span}(\mathcal{T}) \biguplus \text{span}(\mathcal{R})| = T$ **do**
    for all edge pairs e_1, $e_2 \in \mathcal{T}$ **do**
      build $\triangle_{e_1,e_2}(\mathcal{T}, \mathcal{R})$ and a labeling via the $\diamond$-operator
    end for
  end for
end for

As it turns out, in practice the most time consuming part in the algorithm is this recombination phase. Thus we propose a second pruning step to speed up this part significantly. The goal is to reduce the number of $\diamond$-operator calls since they are the most costly operations in this phase - recall that usually the dimension is quite large and the running time of this function grows linearly in the dimension.

lower bound test: Reconsider the lower bound tests for the previous subsection. In the last case, that is $S = R$, we have seen that the tree T_S can be pruned if

$$\text{cost}(T_S) + lb_{\text{SMT}}(\overline{S}) > ub_{\text{RPN}}(T)$$

Note that this condition can be much stronger than the more general condition which involves the subtraction of the bnsd$(\cdot)$ term which can be as big as the dimension of the problem. If a topology T_R satisfies now this condition we know that T_R cannot be the recombinant of an optimal solution, that is we can skip this call of the for-loop without calling the $\diamond$-operator.

minor test: Recall that we want to avoid the computation of the $\diamond$-operator. Consider the recombination phase (see Figure 1.14). Let T_S be a subtopology of $\mathcal{T}$ spanning the terminal set S. If you want to connect the recombinant $\mathcal{R}$ to edges not contained in T_S then the resulting recombination network can clearly not be optimal if

$$\text{cost}(T_S) + lb_{\text{RPN}}(\overline{S}) > ub_{\text{RPN}}(T)$$

But how can we compute $lb_{\text{RPN}}(\overline{S})$? As in the case of the Steiner minimum tree problem in Hamming metric the 1-recombination network problem is also monotone. That is, computing the network for a small subset of $\overline{S}$ yields a lower bound for $lb_{\text{RPN}}(\overline{S})$.

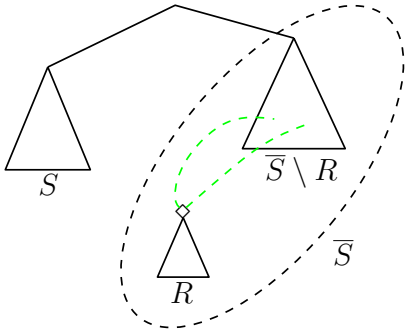


Figure 1.14: Pruning by minors.

1.9.4 Experiments

Performing experiments in the context of reconstructing phylogenetic networks with recombination events is a difficult task since there is always some uncertainty about whether recombination events actually took place, see [ML06] for references. On the other hand synthetically created data sets have the advantage that we know exactly in advance what we expect from our method to return as a phylogenetic network. Furthermore it is easy to create instances of arbitrary size and complexity. Thus we decided to test out model on artificial data sets.

In order to obtain data with a known topology, we use NETGEN [MM06] to simulate an evolution with exactly one recombination event. NETGEN is an event-driven simulator that creates phylogenetic networks by extending the birth-death model to include diploid hybridizations.

Recall that a recombination network with exactly one recombination event can be interpreted as two phylogenetic trees in different parts of the sequence. We use Seq-Gen [RG97] to simulate the evolution of gene sequences along the trees of the recombination networks. Seq-Gen is a program that simulates the evolution of nucleotide or protein sequences along a phylogeny, using common models of the substitution process. A range of models of molecular evolution are implemented, including the general reversible model.

The data for the two trees is then merged by randomly choosing a jump point and by choosing the data alternately from the two trees.

Fixed Recombination Scenarios

To validate our model we generate in each of the discussed cases 40 phylogenetic networks with exactly one recombination. For each network N we simulate with Seq-Gen gene sequences of length 1000 along N partitioning the root sequence of the recombinant randomly such that the expected size of a consecutive block of characters from one tree sequence is 250. As the substitution model we choose the "general reversible process" model (Yang, 1994) for the nucleotide sequences since it is the most general model that is still consistent with the requirement of being reversible.

To show that our method is capable of handling strongly correlated as well as highly uncorrelated input data with high accuracy we scale the branch lengths of the tree obtained by the NETGEN program using the "-s" parameter of Seq-Gen before the

sequences are computed. We show that our model can handle a wide range for this scaling parameter.

Note that we make no limiting assumptions. All sequences involved in the recombination event can be still part of our input but they do not have to be there. Furthermore the size of the recombinant is not limited as well. This is in stark contrast to recombination detection methods like [ML06] where all fixed recombination scenarios contain recent recombinations.

To evaluate our approach we reconstruct the phylogenetic network for the input sequences and compare the resulting network with the original one using the program Treedist which is part of the phylip program package (see [WEBa]). Treedist implements the symmetric difference method (Robinson and Foulds, 1981) to measure tree distances. Notice that as discussed before we can compare the networks by just comparing the two trees that build the network.

Note that this is a much stronger condition than just identifying the recombinant since it involves the proper reconstruction of the non-recombinant part of the network. As far as we know there are no tools available that would handle the general case. Therefore, we compare the reliability of our method with the reliability of the parsimony method for trees. The goal is to show that on data sets on which our method fails, the parsimony method for trees would also return false results. To do so we take the data constructed by Seq-Gen and split it along the two trees T_1 and T_2 out of which we have constructed the network. Then we compute for these sets of sequences the maximum parsimony trees and test via treedist if they represent the topologies T_1 and T_2.

For instances that still contain the recombinant in and for which the size of the recombinant is exactly one, we compare our method with the state-of-the-art recombination detection method of Maydt and Lengauer [ML06] Recco that is specialized for these scenarios. One has to point out that this test is not perfectly fair for any of the two methods, since both programs are not really optimized to output the correct recombinant. It is possible that the correct network is constructed internally but the wrong recombinant is reported. Since we cannot access the internal data structures of Recco we just compare the answers to the question: is the right recombinant sequence reported by the program. Since both implementations suffer from this fact, the experiments are not biased in the direction of any of these approaches.

Results

Even if it is only a special case in our model our experiments clearly show the superiority of our approach to the method of Maydt and Lengauer when the sequence divergence is low (see table 1.5). For small branch lengths we report the correct recombinant more often than the algorithm of Maydt and Lengauer does. Thus, our method seems to work very good in the cases that Posada et al. reported to be hard to detect [PCH02], i.e. in cases where the divergence is low. Note that for these experiments we have chosen a rather dull parameter value of 2 for α in the $\diamond$-operator. We are sure that the results can be better when α is chosen more carefully.

If we consider table 1.6 we can see that the reliability of our method is strongly connected to the reliability of parsimony reconstructions. On one hand if you consider the resulting implications of non-reliability you can see in table 1.7 that almost independently of the branch length scaling factor the unreliability of our method is induced by

s=	0.005	0.01	0.05	0.1	0.5	1.0	1.5
Recomb	31	24	30	31	33	20	15
Recco	18	17	25	26	30	32	21

Table 1.5: **The comparison of Recco and Recomb:** the table gives the number of correctly reported recombinants depending on the branch length scaling factor s. Note that for each choice of s 40 tests have been conducted.

	s=	0.1		0.25		0.5		0.75		1.0		1.25	
R∧P	¬P	40	0	37	0	18	4	12	1	5	1	1	0
¬R	¬ (R∧P)	0	0	1	2	11	7	11	16	9	25	5	34

Table 1.6: **The reliability of Recomb and maximum parsimony I:** the small tables in the big one represent the relation between correct reconstruction of the parsimony trees and the correct reconstruction of the recombination network depending of branch length scaling factor s. Here R∧P means the correct reconstruction of both structures, ¬ (R∧P) denotes that both reconstructions failed and ¬R respectively ¬P means that only the reconstruction of the recombination network respectively the reconstruction of the parsimony trees failed. Note that for each choice of s 40 tests have been conducted.

the unreliability of the parsimony method. On the other hand if the parsimony method fails our method almost always fails as well. Thus, one can conclude that the reliability of our reconstruction method almost corresponds to the reliability of the parsimony method for reconstructing evolutionary trees.

1.9.5 Conclusion

We have presented a new model to compute phylogenetic networks for data, where only a small number of recombination events are assumed to have taken place. Furthermore, we have given an exact algorithm for the case of non or exactly one recombination event. We have shown in our experimental study the high accuracy of our model.

s=	0.1	0.25	0.5	0.75	1.0	1.25
¬R ⇒ ¬P	40	39	29	29	31	35
¬P ⇒ ¬R	40	40	36	39	39	40

Table 1.7: **The reliability of Recomb and maximum parsimony II:** the table show how often the given implication holds. Note that for each choice of s 40 tests have been conducted.

Chapter 2

Problems in Wireless Network Design

2.1 Introduction

Wireless network technology has gained tremendous importance in recent years. It not only opens new application areas with the availability of high-bandwidth connections for mobile devices, but also more and more replaces so far 'wired' network installations. While the spatial aspect was already of interest in the wired network world due to cable costs etc., it has far more influence on the design and operation of wireless networks. The energy required to transmit information via radio waves is heavily correlated with the Euclidean distance of sender and receivers. Hence problems in this area are prime candidates for the use of techniques from computational geometry.

In contrast to wired or cellular networks, ad hoc wireless networks a priori are unstructured in a sense that they lack a predetermined interconnectivity. An ad hoc wireless network is built of a set of radio stations S, each of which consists of a receiver as well as a transmission unit. A radio station v can send a message by setting its *transmission range* $r(v)$ and then by starting the transmission process. All other radio stations at distance at most $r(v)$ from v will be able to receive v's message (we are ignoring interference for now). For transmitting a message across a transmission range $r(v)$, the power consumption of v's transmission unit is proportional to $r(v)^\alpha$, where α is the *transmission power gradient*. In the idealistic setting of empty space, $\alpha = 2$, but it may vary from 2 to more than 6 depending on the environment conditions of the location of the network. Given some transmission range assignment $r : S \to \mathbb{R}_{\geq 0}$ for all nodes in the network, we can derive the so-called *communication graph* $G := (S, E)$. G is a directed graph with vertex set S which has a directed edge (p, q) if and only if $r(p) \geq |pq|$, where $|pq|$ denotes the Euclidean distance between p and q. The cost of the transmission range assignment r is

$$\text{cost}(r) := \sum_{v \in S} r(v)^\alpha$$

Numerous optimization problems can now be considered by looking for the minimum cost transmission range assignment r such that the respective communication graph satisfies some property Π, see [CHP+02] for an overview.

In this work we consider several definitions of Π to solve the following problems:

k-Station Network/k-disk Coverage: *Given a set S of stations and some constant k, we want to assign non-zero transmission powers to at most k stations (senders) such that every station in S can receive a signal from at least one sender.*

k-hop Multicast: *Given a set S of stations, a specific source station $s \in S$, a set of clients/receivers $C \subseteq S \setminus \{s\}$, and some constant k, we want the communication network to contain a directed tree rooted at s spanning all nodes in C with depth at most k. That is, we want to allow a message to be sent from station s to all stations in C using not more than k hops.*

k-set Broadcast: *Given a set S of stations, a specific source station $s \in S$ and some constant k, we want the communication network to contain a directed tree with at most k inner nodes rooted at s and spanning all nodes in S. This means that a communication network should be set up that uses at most k senders but allows s to send messages to all other stations.*

TSP under squared Euclidean distance: *Given a set S of n stations, determine a permutation $p_0, p_1, \ldots p_{n-1}$ of the nodes such that the total energy cost of the TSP tour, i.e. $\sum_{i=0}^{n-1} |p_i p_{(i+1) \bmod n}|^\alpha$ is minimized. The idea is to establish a network such that a token can collect continuously information from the stations.*

All these problems have in common that they are hard to solve in the sense that they are NP-hard or believed to be NP-hard. Thus, in this chapter we aim for algorithms that instead of determining exact solutions, compute approximate ones. For this purpose we also make use of the concept of so-called coresets. That is we show how to identify for a given problem instance I a small sketch that we solve in behalf of the original one. Once solved we show that an approximate solution for I can be computed from the solution of the coreset.

Part of this work was published in [FLN07] and in [FLNL08].

2.2 Preliminaries

Given a set S of n points in the d-dimensional Euclidean space $\mathbb{R}^d$ and a small constant $\alpha \in \mathbb{R}_{\geq 0}$ called the *distance power gradient*. By $|u, v|$ we denote the Euclidean distance between points u and v in $\mathbb{R}^d$.

Definition 2.1 (range/power assignment) *A* range *or* power assignment *r for S is a function $r : S \mapsto \mathbb{R}_{\geq 0}$.*

Definition 2.2 (communication graph) *Given a range assignment r for S. We call the directed graph $ICG_r = (S, E)$ with*

$$E = \{ (u, v) \mid r(u) \geq |u, v| \}$$

the communication graph *induced by r.*

That is, the directed communication graph $ICG_p = (S, E)$ induced by a range assignment r encodes the connectivity information of the network consisting of nodes in S, i.e. there is an edge $(u, v) \in E$ if and only if the node v can be reached by node u.

We call a range assignment *valid* if the induced communication graph satisfies the connectivity property needed by the corresponding problem.

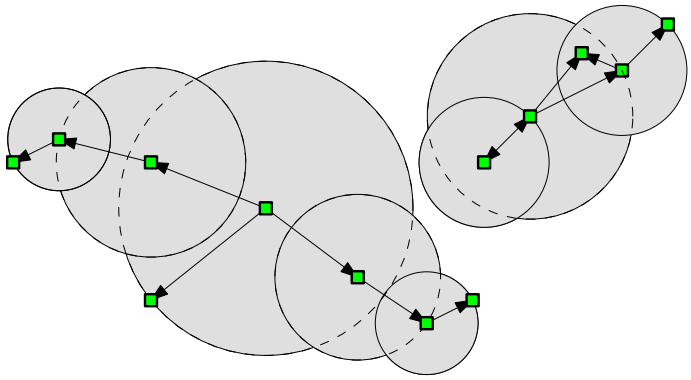


Figure 2.1: Points, a range assignment and the induced communication graph.

Definition 2.3 (cost of a range assignment) *Given a range assignment r for S. We define*

$$\text{cost}(r) := \sum_{u \in S} r(u)^\alpha$$

to be the cost of the range assignment r.

In some of the approximation algorithms which we present, we use the concept of a so-called *coreset*: Given an input S for a some problem a coreset R of S is a (usually small) set with the property that R represents S in an approximate way. That is, from an optimal (or approximate) solution for R we can construct an approximate solution for S. Note that in the analysis of some of the approximation algorithms we assume that we can compute the floor function in constant time.

In the following we will show some technical lemmas which we will use in the following sections.

Lemma 2.1 *For nonnegative r_i, let $\sum_{i=1}^k r_i^\alpha$ be constant. Then for all $\delta \geq 0$ the term*

$$\frac{\sum_{i=1}^k (r_i + \delta)^\alpha}{\sum_{i=1}^k r_i^\alpha}$$

is maximized if $r_1 = r_i$ for all $i \in \{2, \ldots, k\}$.

Proof: Define $\chi := \sum_{i=1}^k r_i^\alpha$ and $(I) := \sum_{i=1}^k (r_i + \delta)^\alpha$. Then $r_i = \left(\frac{\chi}{k}\right)^{1/\alpha}$ $\forall$ $i \in \{2, \ldots, k\}$: Let us first consider the following case: the expression

$$\max_{r_1^\alpha + r_2^\alpha = c} (r_1 + \delta)^\alpha + (r_2 + \delta)^\alpha$$

is maximized when $r_1 = r_2$. Since $r_1^\alpha + r_2^\alpha = c$ we have $r_1 = (c - r_2^\alpha)^{1/\alpha}$. Thus, we want to find the maximum of the function

$$f(r_2) := ((c - r_2^\alpha)^{1/\alpha} + \delta)^\alpha + (r_2 + \delta)^\alpha$$

We have

$$f'(r_2) = \alpha \cdot \left((r_2+\delta)^{\alpha-1} - \left(\frac{r_2 \cdot \left((c - r_2^\alpha)^{1/\alpha} + \delta \right)}{(c-r^\alpha)^{1/\alpha}} \right)^{\alpha-1} \right)$$

Thus,

$$\begin{aligned} f'(r_2) = 0 \quad &\Leftrightarrow \quad r_2 + \delta \;=\; \frac{r_2 \cdot \left((c - r_2^\alpha)^{1/\alpha} + \delta \right)}{(c-r^\alpha)^{1/\alpha}} \\ &\Leftrightarrow \quad r_2 + \delta \;=\; r_2 + \frac{r_2 \cdot \delta}{(c-r^\alpha)^{1/\alpha}} \quad \Leftrightarrow \quad r_2 \;=\; (c/2)^{1/\alpha} \end{aligned}$$

Since $r_1 = (c - r_2^\alpha)^{1/\alpha} = (c/2)^{1/\alpha}$ we have $r_1 = r_2$. Furthermore note that this is the only maximum of the function $f(r_2)$ and that it is indeed a maximum since $f(0) < f((c/2)^{1/\alpha})$ and $f(c^{1/\alpha}) < f((c/2)^{1/\alpha})$. Now let us consider the term (I). Let us assume otherwise that w.l.o.g. $r_1 \neq r_2$ and (I) is maximized. We can restate (I) as follows

$$(I) = \left((r_1+\delta)^\alpha + (r_2+\delta)^\alpha + \sum_{i=3}^{k} (r_i+\delta')^\alpha \right)$$

Let us now fix the r_i for $i \in \{3, \ldots, k\}$. Then (I) is maximal if

$$\max_{r_1^\alpha + r_2^\alpha = \chi - \sum_{i=3}^k r_i^\alpha} (r_1+\delta)^\alpha + (r_2+\delta)^\alpha$$

is maximized. But we have just seen that this is true only if $r_1 = r_2$ which is contradiction to the assumption that $r_1 \neq r_2$. □

Lemma 2.2 *A $(1+\epsilon)^\alpha$-approximation scheme implies a $(1+\omega)$-approximation scheme by setting $\epsilon := \frac{\omega}{c \cdot \alpha}$ assuming that $\omega \leq c$ for a constant $c \geq 0$.*

Proof: Using the inequalities

$$\forall \epsilon \geq 0 : \quad \frac{\epsilon}{\epsilon+1} \;\leq\; \ln(1+\epsilon) \;\leq\; \epsilon$$

we have

$$\begin{aligned} \alpha \ln(1 + \frac{\omega}{c \cdot \alpha}) &\leq \alpha \ln(1 + \frac{\omega}{\omega+1} \cdot \frac{1}{\alpha}) \\ &\leq \alpha \ln(1 + \ln(1+\omega)\frac{1}{\alpha}) \\ &\leq \alpha \cdot \frac{1}{\alpha} \ln(1+\omega) \\ &= \ln(1+\omega) \end{aligned}$$

Thus,

$$(1 + \frac{\omega}{c \cdot \alpha})^\alpha \;\leq\; (1+\omega)$$

□

2.3 The k-Station Network/k-Disk Coverage Problem

In the setting of the k-station network coverage problem we want to identify at most k senders such that every node $u \in S$ can receive a message from at least one of these senders. The goal is to find such a set of senders and a corresponding range assignment r such that $\text{cost}(r)$ is minimized. Depending on which points we allow for being possible senders one can distinguish between the *discrete case* in which the senders have to be contained in S and the *non-discrete case* in which the senders can be arbitrarily located in the Euclidean space. Bilo et al. propose in [BCKK05] to consider this problem as a geometric problem in which the points in S should be covered by at most k disks:

Definition 2.4 (k-station network coverage problem) *Let S be a set of n points in the d-dimensional Euclidean space. Given a $k \in \mathbb{N}$, Find a set of nodes $P \subseteq \mathbb{R}^d$ ($P \subseteq S$ in the discrete case) with $|P| = k$ and a range assignment r for $P \cup S$ with $ICG_p = (P \cup S, E)$ such that:*

- $\forall\, u \in S \setminus P :\ r(u) = 0$
- $\forall\, v \in S\ \exists\, u \in P :\ (u, v) \in E$
- $\text{cost}(r)$ *is minimal*

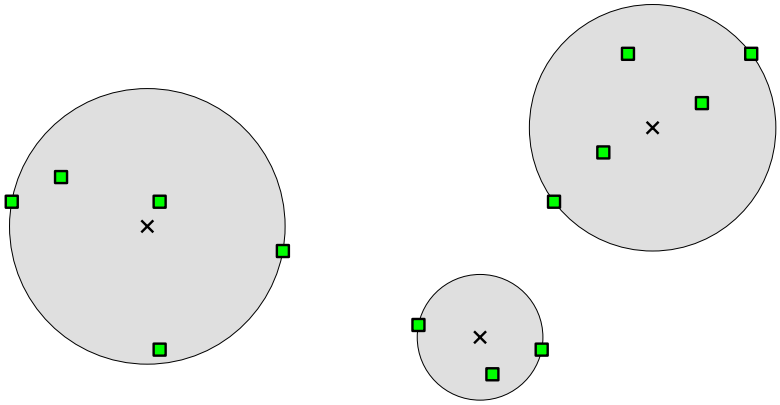


Figure 2.2: Covering input points by 3 discs.

Note that the first property is actually redundant since it is implied by the fact that $\text{cost}(r)$ is minimal.

2.3.1 Our Contribution

In Section 2.3.3 we show how to find a coreset of size **independent of** n and polynomial in k and $1/\epsilon$ for the $k-$Station Network Coverage/k-Disk cover problem. This enables us to improve the running time of the $(1+\epsilon)$-approximation algorithm by Bilo et al.[BCKK05] from

$$n^{((\alpha/\epsilon)^{O(d)})}$$

to

$$O\left(n + \left(\frac{k^{\frac{2d}{\alpha}+1}}{\epsilon^d}\right)^{\min\{\,2k+1,\ (\alpha/\epsilon)^{O(d)}\,\}}\right)$$

that is, we obtain a running time that is *linear* in n. We also present a variant that allows for the senders to be placed arbitrarily (not only within the given set of points) as well as a simple algorithm which is able to tolerate few outliers and runs in polynomial time for constant values of k and a constant number of outliers.

2.3.2 Related Work

As mentioned before the *k-Station Network Coverage* problem was considered by Bilo et al. [BCKK05] as a k-disk cover, i.e. covering a set of n points in the plane using at most k disks such that the sum of the areas of the disks is minimized. They show that obtaining an exact solution is $\mathcal{NP}$-hard and provide a $(1+\epsilon)$-approximation to this problem that runs in time $n^{((\alpha/\epsilon)^{O(d)})}$ based on a plane subdivision and dynamic programming. Variants of the k-disk cover problem were also discussed in [AAB+06].

2.3.3 A Small Coreset For k-Disk Cover

In this section we describe how to find a coreset of size

$$O\left(\frac{k^{\frac{2d}{\alpha}+1}}{\epsilon^d}\right) \quad ,$$

i.e. of size independent of n and polynomial in k and in $1/\epsilon$.

For now let us assume that we are given the cost of a λ-approximate solution S^λ for the point set S. We start by putting a regular d-dimensional grid on S with grid cell width δ depending on $\text{cost}(S^\lambda)$. For each cell C in the grid we choose an arbitrary representative point in $S \cap C$. We denote by R the set of these representatives. We say that C is *active* if $S \cap C \neq \emptyset$. Note that the distance between any point in $S \cap C$ and the representative point of C is at most $\sqrt{d} \cdot \delta$. In the following we write P^{OPT} for an optimal solution for a point set P. Now we obtain a solution R_S^{OPT} by increasing the disks in an optimal solution R^{OPT} by an additive term $\sqrt{d} \cdot \delta$. Since each point in S has a representative in R with distance at most $\sqrt{d} \cdot \delta$, R_S^{OPT} covers S. In the following we will show that

- the cost of R_S^{OPT} is close to the cost of an optimal solution S^{OPT} to the original input set S and
- the size of the coreset R is small.

Theorem 2.1 *We have in the*

$$\begin{aligned} \textit{non-discrete case:} \quad & \text{cost}(R_S^{OPT}) \leq (1+\epsilon)^{\alpha} \cdot \text{cost}(S^{OPT}) \\ \textit{discrete case:} \quad & \text{cost}(R_S^{OPT}) \leq (1+\epsilon)^{2\alpha^2} \cdot \text{cost}(S^{OPT}) \end{aligned}$$

$$\textit{for} \quad \delta := \frac{1}{\sqrt{d}} \cdot \delta' \quad \textit{where} \quad \delta' := \frac{\epsilon}{k^{1/\alpha}} \left(\frac{\text{cost}(S^\lambda)}{\lambda}\right)^{1/\alpha}$$

Proof: Suppose R^{OPT} is given by k balls $(C_i)_{i\in\{1,\dots,k\}}$ with radii $(r_i)_{i\in\{1,\dots,k\}}$. Then

$$\frac{\text{cost}(R_S^{OPT})}{\text{cost}(S^{OPT})} = \frac{\sum_{i=1}^{k}(r_i+\sqrt{d}\cdot\delta)^\alpha}{\text{cost}(S^{OPT})} =: (I)$$

One can easily show that this term is maximized when $r_1 = r_i \;\forall\; i \in \{2,\dots,k\}$ (see Lemma 2.1). Thus we have

$$\begin{aligned}\frac{\text{cost}(R_S^{OPT})}{\text{cost}(S^{OPT})} &\le k\cdot\frac{\left(\frac{\text{cost}(R^{OPT})^{1/\alpha}}{k^{1/\alpha}}+\frac{\epsilon}{k^{1/\alpha}}\left(\frac{\text{cost}(S^\lambda)}{\lambda}\right)^{1/\alpha}\right)^\alpha}{\text{cost}(S^{OPT})}\\ &= \frac{\left(\text{cost}(R^{OPT})^{1/\alpha}+\epsilon\left(\frac{\text{cost}(S^\lambda)}{\lambda}\right)^{1/\alpha}\right)^\alpha}{\text{cost}(S^{OPT})}\\ &\le \left(\frac{\text{cost}(R^{OPT})^{1/\alpha}}{\text{cost}(S^{OPT})^{1/\alpha}}+\epsilon\right)^\alpha =: (II)\end{aligned}$$

Now let us distinguish between the discrete and the non-discrete case:

- In the discrete case $(II) \le (1+\epsilon)$ follows from the monotonicity of the problem, i.e. that for all subsets $P \subseteq S:\ \text{cost}(P^{OPT}) \le \text{cost}(S^{OPT})$. This can easily be seen, since each feasible solution for S is also a feasible solution for P.
- In the non-discrete case $\text{cost}(R^{OPT})$ can be bigger than $\text{cost}(S^{OPT})$ - but not much as we will see: Given an optimal solution S^{OPT}. We transform S^{OPT} into a feasible solution S_P^{OPT} for R by shifting the ball centers to their corresponding representative point in R. Since some points can be uncovered now we have to increase the ball radii by an additional $\sqrt{d}\cdot\delta$. Following exactly the same analysis as above, in this case $\text{cost}(S_R^{OPT}) \le (1+\epsilon)^\alpha\cdot\text{cost}(S^{OPT})$. Hence $\text{cost}(R_S^{OPT}) \le ((1+\epsilon)^\alpha+\epsilon)^\alpha \le (1+\epsilon)^{2\alpha^2}\cdot\text{cost}(S^{OPT})$.

□

Knowing that the coreset is a good representation of the original input set S we will show in the following theorem that R is also small and that the size of the coreset is even smaller in the important case where $\alpha = d$, that is in the case in which the cost of a ball is proportional to its volume.

Theorem 2.2 *The size of the computed coreset R is bounded by*

$$O\left(\frac{k^{\frac{d}{\alpha}+1}\cdot\lambda^{d/\alpha}}{\epsilon^d}\right) \quad \textit{for} \quad \epsilon\in o((k\cdot\lambda)^{1/\alpha}) \tag{2.1}$$

$$O\left(\frac{k\lambda}{\epsilon^d}\right) \quad \textit{for} \quad \epsilon\in o(\lambda^{1/d}) \;\; \textit{and} \;\; \alpha = d \tag{2.2}$$

Proof: We first prove statement 2.2: Observe that the size of R is exactly given by the number of cells that contain a point in S. The idea is now to use an optimal solution S^{OPT} to bound the number of active cells. We can do so because any feasible solution for S covers all points in S and thus a cell C can only be active if such a solution covers

fully or partially C. Thus the number of active cells $\#cc$ cannot be bigger than the volume of such a solution divided by the volume of a grid cell. To ensure that also the partially covered cells are taken into account we increase the radii by an additional term $\sqrt{d} \cdot \delta$. Thus consider S^{OPT} given by by k balls $(C_i)_{i\in\{1,\dots,k\}}$ with radii $(r_i)_{i\in\{1,\dots,k\}}$. Then

$$\begin{aligned}
\#cc &\leq \sum_{i=1}^{k} \frac{2^d \cdot \left(r_i + \sqrt{d}\cdot\delta\right)^d}{\delta^d} = (2\sqrt{d})^d \cdot \sum_{i=1}^{k} \left(1 + \frac{r_i}{\delta'}\right)^d \\
&\overset{(*)}{\leq} (2\sqrt{d})^d \cdot \sum_{i=1}^{k} \left(1 + \frac{(k\cdot\lambda)^{1/\alpha}}{\epsilon} \cdot \left(\frac{\text{cost}(S^{OPT})}{\text{cost}(S^{\lambda})}\right)^{1/\alpha}\right)^d \\
&\leq (2\sqrt{d})^d \cdot k \cdot \left(1 + \frac{(k\cdot\lambda)^{1/\alpha}}{\epsilon}\right)^d
\end{aligned}$$

where inequality $(*)$ follows from the fact that $r_i \leq \text{cost}(S^{OPT})^{1/\alpha}$. Let us now make a case distinction for the possible values of ϵ:

- $\epsilon \in o\left((k\cdot\lambda)^{1/\alpha}\right)$

$$\Rightarrow \quad \#cc \in O\left(\frac{k^{\frac{d}{\alpha}+1}\cdot\lambda^{d/\alpha}}{\epsilon^d}\right)$$

- $\epsilon \in \Omega\left((k\cdot\lambda)^{1/\alpha}\right)$

$$\Rightarrow \quad \#cc \in O(k)$$

We now prove statement 2.2. Since $\alpha = d$ we have for $\epsilon \in o(\lambda^{1/d})$:

$$\begin{aligned}
\#cc &\leq (2\sqrt{d})^d \cdot \sum_{i=1}^{k} \left(1 + \frac{r_i}{\delta'}\right)^d \\
&\overset{\text{lemma 2.1}}{\leq} (2\sqrt{d})^d \cdot k \cdot \left(1 + \frac{k^{1/\alpha}\cdot\lambda^{1/\alpha}}{k^{1/\alpha}\cdot\epsilon} \cdot \left(\frac{\text{cost}(S^{OPT})}{\text{cost}(S^{\lambda})}\right)^{1/\alpha}\right)^d \\
&\leq (2\sqrt{d})^d \cdot k \cdot \left(1 + \frac{\lambda^{1/d}}{\epsilon}\right)^d \in O\left(\frac{k\lambda}{\epsilon^d}\right)
\end{aligned}$$

where the second inequality follows from the fact that the first expression is maximized if $r_i = \left(\frac{\text{cost}(S^{OPT})}{k}\right)^{1/\alpha}$ for $i = 1..k$ (see Lemma 2.1). □

Note that because any $(1+\omega)^\alpha$-approximation algorithm yields a $(1+\epsilon)$-approximation algorithm by setting $\epsilon := \frac{\omega}{c\alpha}$ (we assume that $\omega \leq c$, see Lemma 2.2), doing so increases our coreset by an another *constant factor* of $(c\cdot\alpha)^d$, i.e. the size of R does not change asymptotically. Assuming that $\epsilon \in O(1)$ is reasonable for an approximation scheme.

2.3.4 Algorithms

Still we have to show how to approximate cost(S^{OPT}) for the construction of the grid. In [TD88] Feder et al. show how to compute deterministically a 2-approximate solution for the so-called k-center problem in $O(n \log k)$ time. Furthermore Har-Peled shows in [HP04] how to obtain such an approximation in $O(n)$ expected time for $k = O(n^{1/3}/\log n)$. The k-center problem differs from the k-disk coverage problem just in the objective function which is given by $\max_{i=1..k} \; r_i^\alpha$ where the discs have radii r_i. Since $\max_{i=1..k} r_i^\alpha \geq \frac{1}{k} \cdot \sum_{i=1}^{k} r_i^\alpha$ a 2-approximation for the k-center problem is a $2k$-approximation for the k-disk coverage problem. Using such an approximation the size of our coreset becomes

$$O\left(\frac{k^{\frac{2d}{\alpha}+1}}{\epsilon^d}\right)$$

Now we will show how to solve the coreset.

Discrete Version

Via Bilo et al.: Note that the discrete version of the k-disc cover problem can be solved by the approach of Bilo et. al. [BCKK05]. Recall that their algorithm runs $n^{((\alpha/\epsilon)^{O(d)})}$ time.

Via Exhaustive Search: Alternatively we can find an optimal solution in the following way. We consider all k-subsets of the points in the coreset R as the possible centers of the balls. Note that at least one point in R has to lie on the boundary of each ball in an optimal solution (otherwise you could create a better solution by shrinking a ball). Thus the number of possible radii for each ball is bounded by $n - k$. For each solution we have to check feasibility and to determine the cost which can be done in additional $k \cdot n$ time. Thus we have in total a running time of

$$(n-k)^k \cdot \binom{n}{k} \cdot n \cdot k \leq n^{2k+1}$$

Hence

Corollary 2.1 *The running time of our approximation algorithm in the discrete case is*

$$O\left(n + \left(\frac{k^{\frac{2d}{\alpha}+1}}{\epsilon^d}\right)^{\min\{\, 2k+1,\ (\alpha/\epsilon)^{O(d)} \,\}}\right)$$

Non-Discrete Version

Via Exhaustive Search: Note that on each ball D of an optimal solution there must be at least three points (or two points in diametrical position) that define D - otherwise it would be possible to obtain a smaller solution by shrinking D. Thus for obtaining an optimal solution via exhaustive search it is only necessary to check all k-sets of 3- respectively 2-subsets of R which yields a running time of $O(|R|^{3k+1})$. Hence

Theorem 2.3 *A $(1+\epsilon)$-approximate solution of the non-discrete k-disk coverage problem can be found in*

$$O\left(n + \left(\frac{k^{\frac{2d}{\alpha}+1}}{\epsilon^d}\right)^{3k+1}\right)$$

2.3.5 k-Disk Cover With Few Outliers

Assume we want to cover not all points by disks but we relax this constraint and allow a few points not to be covered, i.e. we allow c outliers. This way, the optimal cover might have a considerably lower power consumption/cost.

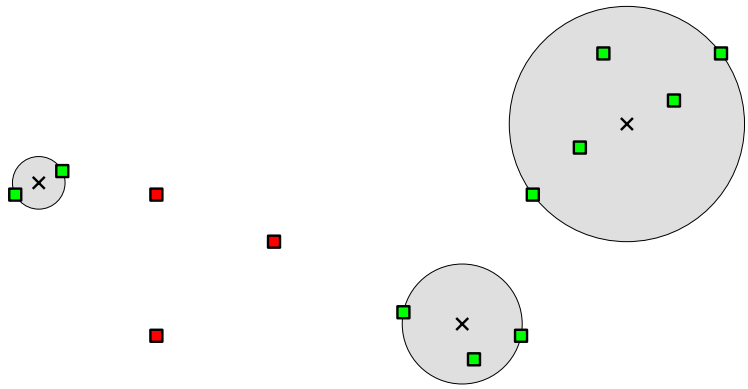


Figure 2.3: Covering input points by 3 discs while allowing to not cover 3 outliers (red points).

Conceptually, we think of a k-disk cover with c outliers as a $(k+c)$-disk cover with c disks having radius 0. Doing so we can use the same coreset construction as above replacing k by $k+c$. Obviously, the cost of an optimal solution to the $(k+c)$-disk cover problem is a lower bound for the k-disk cover with c outliers. Hence, the imposed grid might be finer than actually needed. Thus, snapping each point to its closest representative still ensures a $(1+\epsilon)$-approximation. Constructed as above the coreset has size

$$O\left(\frac{(k+c)^{\frac{2d}{\alpha}+1}}{\epsilon^d}\right)$$

Again, there are two ways to solve this reduced instance, first by a slightly modified version of the algorithm proposed by Bilo et al. [BCKK05] and second by exhaustive search.

We will shortly sketch the algorithm by Bilo et al. [BCKK05] which is based on a hierarchical subdivision scheme proposed by Erlebach et al. in [EJS01]. Each subdivision is assigned a level and they together form a hierarchy. All possible balls are also assigned levels depending on their size. Each ball of a specific level has about the size of an ϵ-fraction of the size of the cells of the subdivision of same level. Now, a cell in the subdivision of a fixed level is called relevant if at least one input point is covered by one ball of the same level. If a relevant cell Z' is included in a relevant

cell Z and no larger cell $Z^{\text{“}}$ exists that would satisfy $Z' \subseteq Z^{\text{“}} \subseteq Z$, then Z' is called a child cell of Z and Z is called the parent of Z'. This naturally defines a tree. It can be shown that a relevant cell has at most a constant number of child cells (the constant only depending on ϵ, α and d). The key ingredient for the algorithm to run in polynomial time is the fact that there exists a nearly optimal solution where a relevant cell can be covered by only a constant number of balls of larger radius. The algorithm then processes all relevant cells of the hierarchical subdivision in a bottom-up way using dynamic programming. A table is constructed that for a given cell Z, a given configuration P of balls having higher level than Z (i.e. large balls) and an integer $i \leq k$ stores the balls of level at most the level of Z (i.e. small balls) such that all input points in Z are covered and the total number of balls is at most i. This is done for a cell Z by looking up the entries of the child cells and iterating over all possible ways to distribute the i balls among them.

The k-disk cover problem with c outliers exhibits the same structural properties as the k-disk cover problem without outliers. Especially, the local optimality of the global optimal solution is preserved. Hence, we can adapt the dynamic programming approach of the original algorithm. In order for the algorithm to cope with c outliers we store not only one table for each cell but c+1 such tables. Each such table corresponds to the table for a cell Z where $0, 1, \ldots, c$ points are not covered. Now, we do not only iterate over all possible ways to distribute the i balls among its child cells but also all ways to distribute $l \leq c$ outliers. This increases the running time to $n^{((\alpha/\epsilon)^{O(d)})} \cdot c^{((\alpha/\epsilon)^{O(d)})} = n^{((\alpha/\epsilon)^{O(d)})}$. Hence running the algorithm on the coreset yields the following result:

Corollary 2.2 *We can compute a minimum k-disk cover with c outliers $(1+\epsilon)$ approximately in time*

$$O\left(n + \left(\frac{(k+c)^{\frac{2d}{\alpha}+1}}{\epsilon^d}\right)^{(\alpha/\epsilon)^{O(d)}}\right).$$

For the exhaustive search approach we consider all assignments of k disks each having a representative as its center and one lying on its boundary. For each such assignment we check in time $O(kn)$ whether the number of uncovered points is at most c. We output the solution with minimal cost.

Corollary 2.3 *We can compute a minimum k-disk cover with c outliers $(1+\epsilon)$ approximately in time*

$$O\left(n + k\left(\frac{(k+c)^{\frac{2d}{\alpha}+1}}{\epsilon^d}\right)^{2k+1}\right).$$

2.4 The k-Hop Multicast Problem

The problem discussed in this section is a constrained broadcast problem in which one wants to determine an energy efficient way to quickly (i.e. within few transmissions) disseminate a message to a small set of receivers in a wireless network:

Definition 2.5 (k-hop multicast problem) *Given a set S of points (stations) in $\mathbb{R}^d$, a distinguished source point $s \in S$ (sender), and a set $C \subset S$ of client points*

(receivers) we want to find a minimal cost range assignment r for S such that the resulting communication graph contains a tree rooted at s spanning all elements in C and with depth at most k.

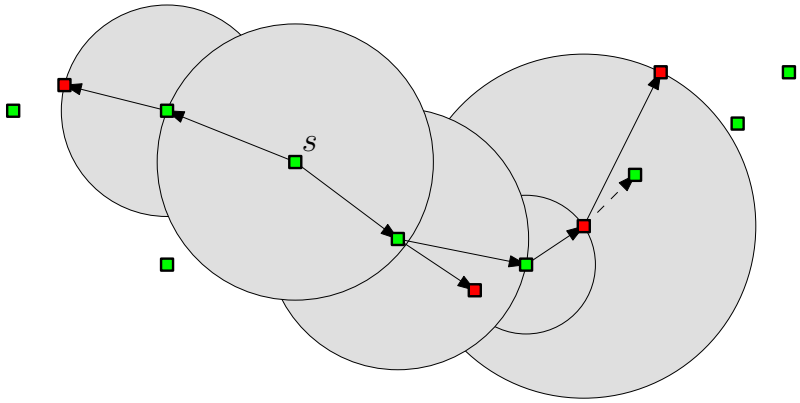


Figure 2.4: The communication graph contains a spanning tree such that messages can be sent from node s to the points in C (red points) within 4 hops.

2.4.1 Our Contribution

As in the previous Section we will solve this problem by first deriving a coreset R of size independent of $|S| =: n$ and then invoking an exhaustive search algorithm. We assume both k and $|C| =: c$ to be constants. The resulting coreset will have size of $O\left(\frac{(kc)^2}{\epsilon^2}\right)$, that is a size *polynomial* in $1/\epsilon$, c and k. For few receivers this is a considerable improvement over the exponential-sized coreset of size $O\left(\left(\frac{1}{\epsilon}\right)^{4k}\right)$ that was used in [FL07] for the k-hop broadcast.

2.4.2 Related Work

The general broadcast problem – assigning powers to stations such that the resulting communication graph contains a directed spanning tree and the total amount of energy used is minimized - has a long history. The problem is known to be $\mathcal{NP}$-hard for $\alpha > 1$ ([CCP+01, CHP+02]), and for arbitrary, non-metric distance functions the problem can also not be approximated better than a log-factor unless $\mathcal{P} = \mathcal{NP}$ [GK99]. Note that for $\alpha \leq 1$ an optimal solution is always given by a single hop. For the Euclidean setting in the plane, it is known ([Amb05]) that the minimum spanning tree induces a range assignment for broadcast which is at most 6 times as costly as the optimal solution. This bound for a MST-based solution is tight ([CCP+01], [WCLF01]). There has also been work on restricted broadcast operations more in the spirit of the *k-hop multicast* problem that we consider in this section. In [ACI+04] the authors examine a *bounded-hop* broadcast operation where the resulting communication graph has to contain a spanning tree rooted at the source node s of depth at most k. They show how to compute an optimal k-hop broadcast range assignment for $k = 2$ in time $O(n^7)$.

For $k > 2$ they show how to obtain a $(1+\epsilon)$-approximation in time $O(n^{O(\mu)})$ where $\mu = (k^2/\epsilon)^{2^k}$, that is, their running time is triply exponential in the number of hops k which shows up in the exponent of n. In recent work [FL07], Funke and Laue show how to obtain a $(1+\epsilon)$ approximation for the k-hop broadcast problem in time doubly exponential in k based on a coreset which has size exponential in k, though.

2.4.3 Preliminaries

As stated above we will use again the concept of a coreset. The k-hop multicast problem differs from the k-disk coverage problem that we have discussed in the previous section in two important ways: first the multicast problem is not monotone, that is removing a point from the input set can decrease or increase the objective value and second the coreset to be constructed will not be a subset of the input set. This makes it necessary to use the idea of coresets more carefully. We will now provide a definition of a γ-coreset in the setting of the k-hop multicast problem and we will then show how the existence of such a coreset leads to the existence of an approximation algorithm.

Definition 2.6 (γ-coreset) *Let S be a set of n points. Consider another set R of points (not necessarily a subset of S). R is called a γ-coreset for (S, s) if*

- *for any valid range assignment $r : S \to \mathbb{R}_{\geq 0}$ there exists a valid range assignment $r' : R \to \mathbb{R}_{\geq 0}$ such that $\text{cost}(r') \leq \gamma \cdot \text{cost}(r)$*
- *for any valid range assignment $r' : R \to \mathbb{R}_{\geq 0}$ there exists a valid range assignment $r : S \to \mathbb{R}_{\geq 0}$ such that $\text{cost}(r) \leq \gamma \cdot \text{cost}(r')$*

Theorem 2.4 *Given a γ-coreset R for a S. Then any λ-approximate solution for R admits a $\lambda \cdot \gamma^2$-approximate solution for S.*

Proof: Let S^{OPT} be an optimal range assignment for S, let R^{OPT} be an optimal range assignment for R and let R^{λ} be a λ-approximate solution for R. Since R is a γ-coreset we have that for R^{λ} there exists a solution S_R feasible for S with $\text{cost}(S_R) \leq \gamma \cdot \text{cost}(R^{\lambda})$ and for S^{OPT} there exists a solution R_S feasible for R such that $\text{cost}(R_S) \leq \gamma \cdot \text{cost}(S^{OPT})$. Thus

$$\begin{aligned} \text{cost}(S_R) &\leq \gamma \cdot \text{cost}(R^{\lambda}) \overset{(*)}{\leq} \lambda \cdot \gamma \cdot \text{cost}(R_S) \\ &\leq \lambda \cdot \gamma^2 \cdot \text{cost}(S^{OPT}) \end{aligned}$$

where inequality $(*)$ holds because R_S is feasible for R. □

2.4.4 A Small Coreset For k-hop Multicast

We will now construct a $(1+\epsilon)$-coreset for the k-hop multicast problem. In the following we will discuss the planar case in $\mathbb{R}^2$, the approach extends in the obvious way to higher dimensions. Assume without loss of generality that the maximal distance of a point $p \in S$ from s is exactly 1. We place a regular grid of cell width $\delta = \frac{1}{\sqrt{2}} \frac{\epsilon}{kc^{1/\alpha}}$ on S. The number of cells in this grid is $O(\frac{k^2 c^{2/\alpha}}{\epsilon^2})$. Now we assign each point in P to its closest grid point. Let R be the set of grid points that had at least one point from S snapped

to it and let C' be the set of grid points that have at least one point from C snapped to it.

It remains to show that R is indeed a coreset. We can transform any given valid range assignment r for S (with respect to the receiver set C) into a valid range assignment r' for R (with respect to the set of receivers C'). We define the range assignment r' for S as

$$r'(p') = \max_{p \text{ was snapped to } p'} r(p) + \sqrt{2}\delta.$$

Since each point p is at most $\frac{1}{\sqrt{2}}\delta$ away from its closest grid point p' we certainly have a valid range assignment for R. It is easy to see that the cost of r' for R is not much larger than the cost of r for S. We have:

$$\begin{aligned}\sum_{p' \in R} (r'(p'))^\alpha &= \sum_{p \in S} (\max_{p \text{ was snapped to } p'} r(p) + \sqrt{2}\delta)^\alpha \\ &\leq \sum_{p \in S} (\max_{p \text{ was snapped to } p'} r(p) + \frac{\epsilon}{kc^{1/\alpha}})^\alpha \\ &\leq \sum_{p \in S} (r(p) + \frac{\epsilon}{kc^{1/\alpha}})^\alpha.\end{aligned}$$

The relative error satisfies

$$\frac{\text{cost}(r')}{\text{cost}(r)} = \frac{\sum_{p \in S} (r(p) + \frac{\epsilon}{kc^{1/\alpha}})^\alpha}{\sum_{p \in S} (r(p))^\alpha}$$

Let us define $\chi := \sum_{p \in S} (r(p))^\alpha$. Lemma 2.1 states that the above expression is maximized if for all $p, q \in S$: $r_p = r_q = (\frac{\chi}{l})^{1/\alpha}$ where $l = |\{r_p \mid p \in S,\ r_p > 0|$. Thus

$$\begin{aligned}\frac{\text{cost}(r')}{\text{cost}(r)} &\leq \frac{l \cdot ((\frac{\chi}{l})^{1/\alpha} + \frac{\epsilon}{kc^{1/\alpha}})^\alpha}{l \cdot (\frac{\chi}{l})} = (1 + \frac{\epsilon \cdot l^{1/\alpha}}{kc^{1/\alpha} \cdot \chi^{1/\alpha}})^\alpha \\ &\overset{(*)}{\leq} (1 + \frac{\epsilon \cdot (kc)^{1/\alpha}}{kc^{1/\alpha} \cdot (k \cdot (\frac{1}{k})^\alpha)^{1/\alpha}})^\alpha = (1+\epsilon)^\alpha\end{aligned}$$

Where inequality $(*)$ follows from the following two observations: First $l \leq kc$ because each of the c receivers must be reached in at most k hops. Second $k \cdot (1/k)^\alpha$ is a lower bound for χ. For this to be seen consider any receiver c_r in C. We are allowed to use $k-1$ intermediate stations for a message to be delivered from s to c_r. In a cost minimal situation these stations lie on the line segment $\overline{s, c_r}$ because the cost function satisfies the triangle inequality. Furthermore Lemma 2.1 implies that the stations are distributed at equal distances on $\overline{s, c_r}$. Thus each of the k sending station contributes $(1/k)^\alpha$ to the cost.

On the other hand we can transform any given valid range assignment r' for R into a valid range assignment r for S as follows. We select for each grid point $g \in R$ one representative g_S from S that was snapped to it. For the grid point to which the source s was snapped we select s as the representative. If we define the range assignment r for S as $r(g_S) = r'(g) + \sqrt{2}\delta$ and $r(p) = 0$ if p does not belong to the chosen representatives, then r is a valid range assignment for S because every point is moved by at most $\delta/\sqrt{2}$. Using the same reasoning as above we can show that $\text{cost}(r) \leq (1+\epsilon)^\alpha \,\text{cost}(r')$. In summary we obtain the following theorem:

Theorem 2.5 *For the k-hop multicast problem with c receivers there exists a coreset of size $O\left(\frac{k^2c^{2/\alpha}}{\epsilon^2}\right)$.*

2.4.5 Solution Via a Naive Algorithm

As we are not aware of any algorithm to solve the k-hop multicast problem we employ an exhaustive search strategy, which we can afford since after the coreset computation we are left with a constant problem size. Essentially we consider every kc-subset of R as potential set of senders and try out the $|R|$ potential ranges for each of the senders. Hence, naively there are at most $\binom{\frac{k^2c^{2/\alpha}}{\epsilon^2}}{kc} \cdot \left(\frac{k^2c^{2/\alpha}}{\epsilon^2}\right)^{kc}$ different range assignments to consider at all. We enumerate all these assignments and for each of them check whether the range assignment is valid with respect to c'; this can be done in time $|R|$. Of all the valid range assignments we return the one of minimal cost. We obtain the following corollary:

Corollary 2.4 *A $(1+\epsilon)$-approximate solution to the k-hop multicast problem on n points in the plane can be computed in time*

$$O\left(n + \left(\frac{kc^{1/\alpha}}{\epsilon}\right)^{4kc}\right)$$

But we still can do better. Since we are only interested in a $(1+\epsilon)$-approximate solution we do not have to solve the coreset instance exactly. Instead of enumerating all possible ranges for all kc-subsets of R we only consider the ranges $((1+\epsilon)^\mu \cdot \delta)_{\mu\in\mathbb{N}}$. Recall that the smallest range that we have to consider is not smaller than the grid cell width and not bigger than 2 because of the scaling step in the beginning. Thus there are only $O(\log_{1+\epsilon} \frac{kc^{1/\alpha}}{\epsilon})$ ranges that have to be considered. Note that any optimal solution considering only this set of ranges is at most by a factor of $(1+\epsilon)$ bigger than an optimal solution considering all possible ranges. This leads to

Corollary 2.5 *A $(1+\epsilon)$-approximate solution to the k-hop multicast problem on n points in the plane can be computed in time*

$$O\left(n + \left(\frac{(kc^{1/\alpha})^2 \log \frac{kc^{1/\alpha}}{\epsilon}}{\epsilon^3}\right)^{kc}\right)$$

2.5 The k-Set Broadcast Problem

As mentioned in the previous section, broadcasting a message from a given source node to all other nodes is a fundamental task during the operation of a wireless network. During a broadcast operation using intermediate nodes to relay messages within the network might decrease the overall energy consumption since the cost of transmitting a message grows super-linearly with the distance. On the other hand using too many intermediate nodes during a broadcast operation increases both latency as well as the chances that some transmission could not properly received (e.g. due to interference).

Recall that if the points are located in the Euclidean plane, the minimum spanning tree (MST) of the point set induces a transmission range assignment which has cost at most a factor of 6 above the optimal solution [Amb05]. On the other hand, there are point sets where the MST-based solution is a factor 6 worse than the optimal solution, so the bound of 6 is tight [WCLF01].

One problem that is particularly prominent for the MST–based solution is the fact that in the resulting transmission range assignment a very large fraction of the network nodes are transmitting (i.e. have non-zero transmission range). In the MST-based range assignment, at least $n/6$ nodes are actually senders during the broadcast operation (since the degree of the minimum spanning tree of a set of points in the Euclidean plane is bounded by 6). This raises several critical issues:

- The more network nodes are transmitting in the process of one broadcast operation, the more likely it is that some nodes in the network experience interference due to several nearby nodes transmitting at the same time (unless special precautions are taken that interference does not occur).
- Every retransmission of a message implies a certain delay which is necessary to set up the transmission unit etc; that is, the more senders are involved in the broadcast operation, the higher the latency becomes. This effect is even amplified by the previous problem if due to interference messages have to be resent.
- Network nodes are not 100% reliable; if for example the probability for a network node to operate properly is 99.9%, the probability for a network broadcast to fail, i.e. not all nodes receiving the message, is $1 - 0.999^{(n/6)}$, which is around 40% for a network of $n = 3000$ nodes!

This suggests to look for broadcast operations in the network that use only very few sending nodes. Of course, this comes at the cost of an increased total power consumption, but the behavior with respect to these critical issues can be drastically improved. Formally we consider in this section the following problem:

Definition 2.7 (k-set broadcast problem) *Let S be a set of n points in the d-dimensional Euclidean space. Given a $k \in \mathbb{N}$ and a specific source node $s \in S$, find a range assignment r for S with $ICG_r = (S, E)$ of minimal cost such that*

- *ICG_r contains a directed tree rooted at s*
- $|\{\, p \in S \mid r(p) > 0 \,\}| \;\leq\; k$

2.5.1 Our Contribution

As just mentioned we consider another constrained broadcast operation, where a source node wants to send a message to all other nodes in the network but at most k nodes are allowed to participate actively, that is transmit the message. Allowing only a small number k of sending nodes during the broadcast operation has several advantages:

- The k transmissions can be easily scheduled in k different time slots, hence avoiding any interference at all.

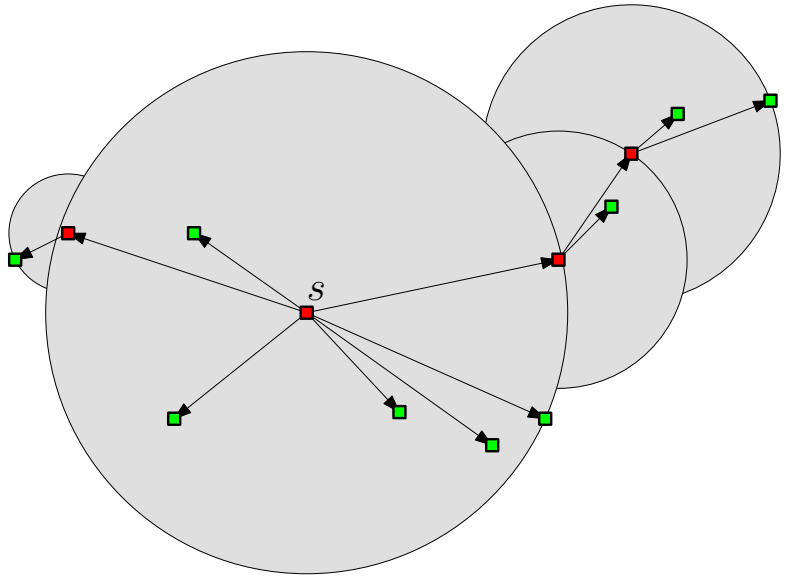


Figure 2.5: The communication graph contains a spanning tree such that messages can be sent from node s to all other nodes, but only 4 nodes (red points) are allowed to send.

- The latency is obviously bounded by $O(k)$.
- In the above scenario the probability of a broadcast operation to fail is $1-0.999^k$, which e.g. for $k = 10$ is 1%.

For the case of network nodes embedded in the Euclidean plane we provide a $(1+\epsilon)$-approximation algorithm which runs in time linear in n and polynomial in $1/\epsilon$ but with an exponential dependence on k. As an alternative we therefore also provide an $O(1)$-approximation whose running time is linear in n and polynomial in k. The existence of a $(1+\epsilon)$-approximation algorithm is in stark contrast to the unconstrained broadcast problem where even in the Euclidean plane no algorithm with approximation factor better than 6 is known so far.

2.5.2 Related Work

As stated before, the unconstrained energy minimal broadcast problem is known to be $\mathcal{NP}$-hard for $\alpha > 1$ ([CCP+01, CHP+02]). For arbitrary, non-metric distance functions the problem can also not be approximated better than a log-factor unless $\mathcal{P} = \mathcal{NP}$ [GK99]. For the Euclidean setting in the plane, [CCP+01] and [WCLF01] have shown a lower bound of 6 for the approximation ratio of the MST-based solution. In a sequence of papers the upper bound for this solution was lowered in several steps (e.g. [CCP+01, WCLF01, FNKP04]) to finally match its lower bound of 6 (Ambühl, [Amb05]). While all these papers focused on analytical worst-case bounds for the algorithm performance, simulation studies e.g. in [CHR+03] show that the actual performance in "real-world" networks is much better. There has also been work on more

restricted broadcast operations in the spirit of k-SEMBC. In [ACI+04] the authors consider a *bounded-hop* broadcast operation where the resulting communication graph has to contain a spanning tree rooted at the source node s of depth at most h. They show how to compute an optimal h-hop broadcast range assignment for $h = 2$ in time $O(n^7)$. For $h > 2$ they show how to obtain a $(1+\epsilon)$-approximation in time $O(n^{O(\mu)})$ where $\mu = (h^2/\epsilon)^{2^h}$, that is, their running time is triply exponential in the number of hops h and this shows up in the exponent of n. In [FL07] Funke and Laue show how to obtain a $(1+\epsilon)$ approximation for the h-hop broadcast problem in time doubly exponential in h. Their approach is also based on a coreset of the network, but in contrast to this work they require a coreset S that has size exponential in h. We note that bounded-hop broadcasts address the issue of latency since a message will arrive at any network node after at most h intermediate stations, still the reliability and interference problems remain as potentially very many network nodes might actively participate in the broadcast. General surveys of algorithmic range assignment problems can be found in [CHP+02, WNE00, KKKP00]. Closely related in particular to the $O(1)$-approximation algorithm that we will present is the work by Bilò et al. [BCKK05]. They consider the problem of covering a set of n points in the plane using at most k disks such that the sum of the areas of the disks is minimized. They provide a $(1+\epsilon)$-approximation to this problem in time $O(n^{\alpha^2/\epsilon^2})$. They do not address the problem of enforcing connectivity which is part of the k-SEMBC problem.

2.5.3 Preliminaries

As mentioned above, the unconstrained broadcast problem is known to be $\mathcal{NP}$-hard and for non-metric distance functions even not well approximable ([CCP+01, GK99]. Since the unconstrained broadcast problem is a special case of the k-set broadcast problem with $k = n$ these hardness results carry over to the k-set broadcast problem, if k is not treated as a constant. If k is regarded a constant, the problem can be solved in polynomial time as we will see in the following.

2.5.4 Algorithms

A Naive, Brute-Force Algorithm

The k-set broadcast problem can be solved in a brute force manner. Essentially, one can try out all $\binom{n}{k-1}$ different subsets for the $k-1$ active senders apart from the source s. For each of those (and the source node s), one then assigns all possible $n-1$ ranges. In total we have then $O(n^{k-1}(n-1)^k) = O(n^{2k})$ potential power assignments. For each of those we can check in $O(n^2)$ time whether it is a valid k-set broadcast.

That is, overall we have the following corollary:

Corollary 2.6 *For n points we can compute the optimal k-set broadcast in time $O(n^{2k+2})$*

For most practical applications, we expect k to be a small constant, but unfortunately not small enough that this naive algorithm can be applied to networks of not too small size (e.g. several thousand nodes). Thus, in the following section we aim for *approximate* solutions to the k-set broadcast problem. This allows for more efficient algorithms as we will see.

Small Coreset of the Network Topology

We will now show that we can find a small coreset to the original problem. We assume that the maximum distance from the source node s to another node is 1.

Lemma 2.3 *For any k-SEMBC instance there exists a $(1+\epsilon)^\alpha$-coreset of size $O\left(\frac{k^2}{\epsilon^2}\right)$.*

Proof: We place a grid of grid width $\delta = \frac{1}{\sqrt{2}}\frac{\epsilon}{k}$ on the plane. Notice, that the grid has to cover an area of radius 1 around the source only because the furthest distance from node s to any other node is 1. Hence its size is $O(\frac{k^2}{\epsilon^2})$. Now we assign each point in S to its closest grid point. Let R be the set of grid points that had at least one point from S snapped to it.

It remains to show that R is indeed a $(1+\epsilon)^\alpha$-coreset. We can transform any given valid range assignment r for S into a valid range assignment r' for R. We define the range assignment r' for R as

$$r'(p') = \max_{p \text{ was snapped to } p'} r(p) + \sqrt{2}\delta.$$

Since each point p is at most $\frac{1}{\sqrt{2}}\delta$ away from its closest grid point p' we certainly have a valid range assignment for R. It is easy to see that the cost of r' for R is not much larger than the cost of r for S. We have:

$$\begin{aligned}
\sum_{p' \in R} (r'(p'))^\alpha &= \sum_{p' \in R} (\max_{p \text{ was snapped to } p'} r(p) + \sqrt{2}\delta)^\alpha \\
&\leq \sum_{p' \in R} (\max_{p \text{ was snapped to } p'} r(p) + \frac{\epsilon}{k})^\alpha \\
&\leq \sum_{p \in S} (r(p) + \frac{\epsilon}{k})^\alpha.
\end{aligned}$$

The relative error satisfies

$$\frac{\text{cost}(r')}{\text{cost}(r)} \leq \frac{\sum_{p \in S}(r(p) + \frac{\epsilon}{k})^\alpha}{\sum_{p \in S}(r(p))^\alpha}.$$

Notice, that $\sum_{p \in S} r(p) \geq 1$ and r is positive for at most k points p. Hence, the above expression is maximized when $r(p) = \frac{1}{k}$ for all points p that are assigned a positive value (see Lemma 2.1). Thus

$$\frac{\text{cost}(r')}{\text{cost}(r)} \leq \frac{k \cdot (\frac{1}{k} + \frac{\epsilon}{k})^\alpha}{k \cdot (\frac{1}{k})^\alpha} = (1+\epsilon)^\alpha.$$

On the other hand we can transform any given valid range assignment r' for R into a valid range assignment r for D as follows. We select for each grid point $g \in R$ one representative g_S from S that was snapped to it. For the grid point to which s (the source) was snapped we select s as the representative. If we define the range assignment r for S as $r(g_S) = r'(g) + \sqrt{2}\delta$ and $r(p) = 0$ if p does not belong to the chosen representatives, then r is a valid range assignment for S because every point is moved by the snapping by at most $\delta/\sqrt{2}$. Using the same reasoning as above we

can show that $\text{cost}(r) \leq (1+\epsilon)^\alpha \,\text{cost}(r')$. Hence, we have shown that R is indeed a $(1+\epsilon)^\alpha$-coreset. □

Once we have solved the k-SEMBC problem for the $(1+\epsilon)^\alpha$-coreset R we can easily transform the obtained solution to a $(1+\epsilon)^{2\alpha}$-approximate solution to the original problem. Let us now concentrate on solving the k-SEMBC problem for the coreset R. Since we were able to reduce the problem size to a constant independent of n, we can employ a brute-force strategy to compute an optimal solution for the reduced problem (R, s).

When looking for an optimal solution for R, it is obvious that for each node only $|R|$ different ranges have to be considered. Hence, naively there are at most $\binom{\frac{k^2}{\epsilon^2}}{k} \cdot \left(\frac{k^2}{\epsilon^2}\right)^k$ different range assignments to consider at all. We enumerate all these assignments and for each of them we check whether the range assignment is valid; this can be done in time $|R|$. Of all the valid range assignments we return the one of minimal cost.

We obtain the following theorem:

Theorem 2.6 *A $(1+\epsilon)^{2\alpha}$-approximate solution to the k-SEMBC problem on n points in the plane can be computed in time $O\left(n + \left(\frac{k}{\epsilon}\right)^{4k}\right)$.*

A simple observation allows us to improve the running time slightly. Since eventually we are only interested in an approximate solution to the problem, we are also happy with only approximating the optimum solution for the coreset R. Such an approximation for R can be found more efficiently by not considering all possible at most $|R|$ ranges for each grid point. Instead we consider as admissible ranges only 0 and $\frac{\epsilon}{k} \cdot (1+\epsilon)^i$ for $i \geq 0$. That is, the number of different ranges a node can attain is at most $1 + \log_{1+\epsilon} \frac{k}{\epsilon} \leq \frac{2}{\epsilon} \cdot \log \frac{k}{\epsilon}$ for $\epsilon \leq 1$. This comes at a cost of a $(1+\epsilon)$ factor by which each individual assigned range might exceed the optimum. The running time of the algorithm improves, though, which leads to our main result in this section:

Theorem 2.7 *A $(1+\epsilon)^{3\alpha}$-approximate solution to the k-SEMBC problem on n points in the plane can be computed in time $O\left(n + \left(\frac{k^2 \log \frac{k}{\epsilon}}{\epsilon^3}\right)^k\right)$.*

Following Lemma 2.2 a $(1+\psi)$-approximate solution can be obtained by choosing $\epsilon = \theta(\psi/\alpha)$.

Faster $O(1)$-Approximations

We now show how to compute a constant approximation for the k-set broadcast problem. The idea is to first cluster the points into k clusters. Then we ensure connectivity of these point sets by increasing their cluster sizes. As clustering we define the *k-disk cover problem*:

Definition 2.8 (k-disk cover problem (k-DCP)) *Given a set S of n points in the Euclidean plane, find a subset $C \subseteq S$ of cardinality at most k and radii $r_v \geq 0$ associated with each element $v \in C$ such that $\sum_{v \in C} r_v^\alpha$ is minimized and all points in S are covered by the disks $D_v := \{x \in \mathbb{R}^2 \mid |xv| \leq r_v\}$.*

Given a k-disk cover $D := (C,\ (r_v)_{v\in C})$ for S with center points C and radii r_v, we associate with D a range assignment r_D on S as follows:

$$\forall v \in S: \quad r_D(v) := \begin{cases} r_v & \text{if } v \in C \\ 0 & \text{otherwise} \end{cases}$$

By D_i we denote a disk in D. Note that the k-DCP with the additional constraint that the communication graph $G^{(r_D)}$ is connected is exactly the k-set broadcast problem. Thus, the cost of an optimal solution for an instance I of the k-DCP is a lower bound on I for the k-set broadcast problem. Unfortunately k-DCP is NP-hard (see [BCKK05]) but it admits a PTAS as shown in [BCKK05] by Bilò et al. A direct consequence of their results is:

Corollary 2.7 *There exists an algorithm for the k-DCP that can compute a $(1+\epsilon)$-approximate solution in time $n^{(\frac{\alpha}{\epsilon})^d}$ for a constant d.*

By setting ϵ to 1 we obtain a 2-approximation algorithm for the k-DCP that runs in time $n^{c'_\alpha}$ for a constant c'_α. Note that their algorithm can easily be modified such that the source s is the center of one of the disks without changing the approximation guarantee and without increasing the running time asymptotically.

Our approximation algorithm works as follows: First we compute an approximate k-disk cover $D := (C,\ (r_v)_{v\in C})$ over S. Then we determine for the center points in C an approximate broadcast with range assignment r_B by using an MST–based algorithm (see [Amb05]) that has an approximation guarantee of 6. Now we construct a range assignment r_A for S in the following way:

$$\forall v \in S: \quad r_A(v) := \begin{cases} \max\{r_v,\ r_B(v)\} & \text{if } v \in C \\ 0 & \text{otherwise} \end{cases}$$

Note that $G^{(r_A)}$ is connected and therefore induces a valid k-set broadcast since only k stations are sending. We still have to show that we have computed an approximate solution:

Theorem 2.8 $\text{cost}(r_A) \leq 36c_\alpha \cdot \text{cost}(r_{opt})$, *where r_{opt} is the range assignment of an optimal k-set broadcast and c_α is a constant depending only on α.*

Proof: The proof idea is as follows: Assuming knowledge about an optimal range assignment r_{opt} for the k-set broadcast we transform the range assignment r_D into r'_D such that

a) r'_D is a valid k-set broadcast

b) the sending nodes in r'_D are exactly the center points of D and

c) r'_D is a constant factor approximation of r_{opt}.

If we know that such a broadcast r'_D exists, we can simply compute an optimal broadcast r_B over the center points of D. Then we know that $\text{cost}(r_B) \leq \text{cost}(r'_D)$ and r_B must also be a constant factor approximation of r_{opt}.

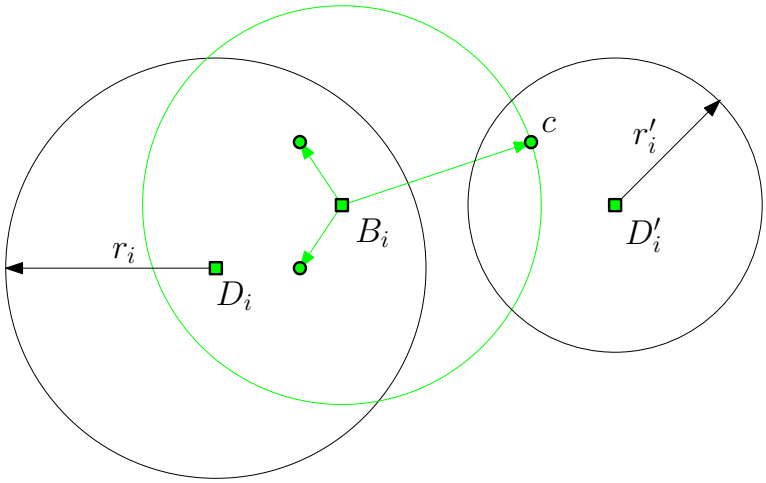


Figure 2.6: Proof illustration for the constant factor approximation algorithm.

Consider now the communication tree T which is defined as a subtree of $G^{(r_{opt})}$ spanning S. The idea of the construction of r'_D is to replace the inner nodes of T (i.e. the sending stations of r_{opt}) by increasing the radii of the disks in C appropriately so that r'_D is a valid range assignment.

We increase the nonzero values of r_D in the following way: with each of the inner nodes B_i of T we associate arbitrarily one disk D_i in which B_i is contained. Note that there must be at least one such disk for each B_i since the disks cover S. We now update r_D in a breath first search manner on T starting from source node s (see Figure 2.6):

Given an inner node B_i of T if all children of B_i in T lie in the associated disk D_i then all of them can be reached from node C_i without increasing r_i. The interesting case is if there are children of B_i that are not contained in D_i but contained in a disk D'_i whose center is not covered so far. Assume that there is exactly one such child c. We then set the radius of D_i to $r_i + r'_i + r_{opt}(B_i)$. If there is more than one such child, let c be the one that maximizes r'_i so that each child of B_i and the centers of the disks in which the children of B_i are contained in can be reached by D_i. Note that it can happen that two different inner nodes B_i and B_j are associated with the same disk D_k, so that D_k is updated more than once in the process. In such a case we update D_k only if r_k is increasing. Now let us assume that for a disk D_i the last update involved disk D'_i then we call disk D'_i the target disk of D_i.

By induction $G^{(r_D)}$ is connected after these updates. furthermore note that the sending stations are still exactly the center points of D. Let $D^* \subseteq D$ be the set of disks that are updated and let B_i be the node in T in the update step for disk $D_i \in D^*$. Summing over all disks, the total cost of the broadcast is therefore bounded by:

$$\underbrace{\sum_{D_i \in D \setminus D^*} r_i^\alpha}_{\leq \mathrm{cost}(D) \leq 2\,\mathrm{cost}(r_{opt})} + \underbrace{\sum_{D_i \in D^*} (r_i + r'_i + r_{opt}(B_i))^\alpha}_{(**)}$$

Before we bound the second term, note that a disk appears as a target disk only once in the process of updating the disk radii since once its center point is covered it is never

considered as a target disk again. Thus each r'_i in the above sum can also appear only once. Thus,

$$\begin{aligned}(**) &\leq c_\alpha\Big[\sum_{D_i\in D^*} r_i^\alpha + \sum_{D_i\in D^*} r_i'^\alpha + \sum_{D_i\in D^*} r_{opt}(B_i)^\alpha\Big]\\ &\leq c_\alpha\Big[\underbrace{2\sum_{D_i\in D} r_i^\alpha}_{\leq 4\,\mathrm{cost}(r_{opt})} + \underbrace{\sum_{D_i\in D} r_{opt}(B_i)^\alpha}_{=\mathrm{cost}(r_{opt})}\Big]\\ &\leq 5\cdot c_\alpha\cdot \mathrm{cost}(r_{opt})\end{aligned}$$

where the constant c_α can be bounded by $3^{\alpha-1}$. Thus there exists a broadcast over the center points C with total cost upper bounded by

$$\begin{aligned}&2\,\mathrm{cost}(r_{opt}) + 5\cdot c_\alpha\cdot\mathrm{cost}(r_{opt})\\ \leq\ & 6\cdot c_\alpha\cdot\mathrm{cost}(r_{opt})\end{aligned}$$

Since we use a 6-approximate broadcast, the algorithm has an approximation ratio of $36c_\alpha$. □

Theorem 2.9 *There exists a constant factor approximation algorithm for the k-set broadcast problem over n points in the Euclidean plane that runs in $O(n^{c'_\alpha})$.*

The theorem can be further improved by using the results of the previous section. By setting ϵ to 1 we obtain a coreset of size k^2. Using Theorem 2.9 we obtain directly a constant factor approximation algorithm whose running time is only linear in n and polynomial in k:

Theorem 2.10 *There exists a constant factor approximation algorithm for the k-set broadcast problem over n points in the Euclidean plane that runs in time linear in n and polynomial in k, i.e. in $O(n + k^{2\cdot c'_\alpha})$.*

2.6 TSP Under Squared Euclidean Distance

In this section we consider the problem of finding energy-optimal TSP tours. The challenge here is that the edge weights induced by the energy costs do not define a metric anymore; a simple example shows that an optimal solution to the Euclidean TSP can be a factor $\Omega(n)$ off the optimum solution. We present an $O(1)$-approximation for the TSP problem with powers α of the Euclidean distance as edge weights. For small α we improve upon previous work by Andreae [And01] and Bender and Chekuri [BC00].

Definition 2.9 (TSP under squared Euclidean distance) *Given a set S of n stations, determine a permutation $p_0, p_1, \ldots p_{n-1}$ of the nodes such that*

$$\sum_{i=0}^{n-1} |p_i p_{(i+1) \bmod n}|^\alpha$$

is minimized.

2.6.1 Our Contribution

We present a constant factor approximation algorithm that improves the result of Andrea [And01] and Bender and Chekuri [BC00] for small α, i.e. for $2 \leq \alpha \leq 2.7$.

2.6.2 Related Work

The classical travelling salesperson problem is NP-complete for arbitrary, non-metric distance functions (see [OM]). However, if the metric satisfies the relaxed triangle inequality $\text{dist}(x, y) \leq \tau(\text{dist}(x, z) + \text{dist}(z, y))$ for $\tau \geq 1$ and every triple of points x, y and z as in our case, then a 4τ approximation exists [BC00].

2.6.3 Why Euclidean TSP Does Not Work

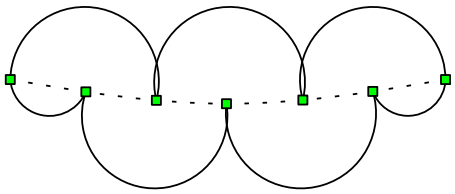


Figure 2.7: An optimal energy-minimal tour for points on a line

Simply computing an optimal tour for the underlying Euclidean instance does not work. The cost for such a tour can be a factor $\Omega(n)$ off from the optimal solution for the energy-minimal tour. Consider the example where n points lie on a slightly bent line and each point having distance 1 to its right and left neighbor. An optimal Euclidean tour would visit the points in their linear order and the go back to the first point. Omitting the fact that the line is slightly bent this tour would have a cost of $(n-1) \cdot 1^2 + (n-1)^2 = n(n-1)$ if the edge weights are squared Euclidean distances. However, an optimal energy-minimal tour would have a cost of $(n-2) \cdot 2^2 + 2 \cdot 1^2 = 4(n-1) + 2$. This tour would first visit every second point on the line and on the way back all remaining points as in Figure 2.7.

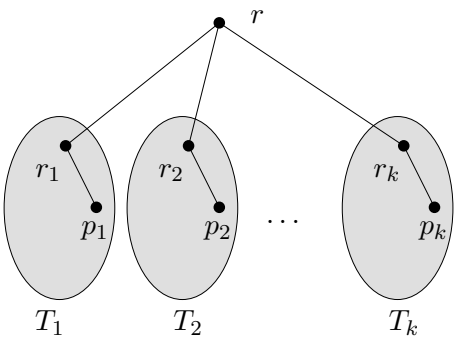


Figure 2.8: Tree T and its child trees $T_1, T_2, \ldots, T_k$

2.6.4 A 6-Approximation Algorithm

In this section we will describe an algorithm which computes a 6-approximation for the TSP under squared Euclidean distance. Obviously, the cost of a minimum spanning tree is a lower bound for the optimal value OPT of the tour. Consider a non-trivial minimum spanning tree T for a graph with node set V and squared Euclidean edge weights. We denote the cost of such a tree by $\mathrm{MST}(T)$. Let r be the root of T and p be one child of T.

We define two Hamiltonian paths $\pi^a(T)$ and $\pi^b(T)$ as follows. Let $\pi^a(T)$ be a path starting at r, finishing at p that visits all nodes of T and the cost of this path is at most $6\mathrm{MST}(T) - 3\|rp\|^2$. Let $\pi^b(T)$ be defined in the same way but in opposite direction, i.e. it starts at p and finishes at r.

Now, if we have such a tour $\pi^a(T)$ for the original vertex set V we can construct a Hamilton tour by connecting r with p. The cost of this tour is clearly at most $6\mathrm{MST}(T) - 3\|rp\|^2 + \|rp\|^2 \leq 6\mathrm{MST}(T) \leq 6\,\mathrm{OPT}$. It remains to show how to construct such tours π^a and π^b. We will do this recursively.

For a tree T of height 1, i.e. a single node r, $\pi^a(T)$ and $\pi^b(T)$ both consist of just the single node. Conceptually, we identify p with r in this case. Obviously, the cost of both paths is trivially at most $6\mathrm{MST}(T) - 3\|rp\|^2$.

Now, let T be of height larger than 1 and let $T_1, \ldots, T_k$ be its children trees. Let r denote the root of T and r_i the root of T_i and p_i be a child of T_i as in figure 2.8. Then we set $\pi^a(T) = (r, \pi^b(T_1), \pi^b(T_2), \ldots, \pi^b(T_k))$.

The cost of the path $\pi^a(T)$ satisfies

$$\begin{aligned}
\mathrm{cost}(\pi^a(T)) &= \|rp_1\|^2 + \mathrm{cost}(\pi^b(T_1)) + \|r_1p_2\|^2 + \mathrm{cost}(\pi^b(T_2)) + \\
&\quad \ldots + \|r_{k-1}p_k\|^2 + \mathrm{cost}(\pi^b(T_k)) \\
&\leq (\|rr_1\| + \|r_1p_1\|)^2 + \mathrm{cost}(\pi^b(T_1)) \\
&\quad +(\|r_1r\| + \|rr_2\| + \|r_2p_2\|)^2 + \mathrm{cost}(\pi^b(T_2)) \\
&\quad \vdots \\
&\quad +(\|r_{k-1}r\| + \|rr_k\| + \|r_kp_k\|)^2 + \mathrm{cost}(\pi^b(T_k)) \\
&\leq 2\|rr_1\|^2 + 2\|r_1p_1\|^2 + \mathrm{cost}(\pi^b(T_1)) \\
&\quad +3\|r_1r\|^2 + 3\|rr_2\|^2 + 3\|r_2p_2\|^2 + \mathrm{cost}(\pi^b(T_2)) \\
&\quad \vdots \\
&\quad +3\|r_{k-1}r\|^2 + 3\|rr_k\|^2 + 3\|r_kp_k\|^2 + \mathrm{cost}(\pi^b(T_k))
\end{aligned}$$

Therefore

$$\begin{aligned}
\mathrm{cost}(\pi^a(T)) &\leq 6\sum_{i=1}^{k}\|rr_i\|^2 + 3\sum_{i=1}^{k}\|r_ip_i\|^2 + \sum_{i=1}^{k}\mathrm{cost}(\pi^b(T_i)) - 3\|rr_k\|^2 \\
&\leq 6\sum_{i=1}^{k}\|rr_i\|^2 + 6\sum_{i=1}^{k}\mathrm{MST}(T_i) - 3\|rr_k\|^2 \\
&= 6\mathrm{MST}(T) - 3\|rr_k\|^2.
\end{aligned}$$

In the above calculation we used the fact that $(\sum_{i=1}^{n} a_i)^\alpha \leq n^{\alpha-1} \cdot \sum_{i=1}^{n} a_i^\alpha$ for $a_i \geq 0$ and $\alpha \geq 1$. This follows directly from Jensen's inequality and the fact that the function $f : x \mapsto x^\alpha$ is convex. The path $\pi^b(T)$ is constructed analogously.

In fact, the very same construction and reasoning can be generalized to the following corollary.

Corollary 2.8 *There exists a $2 \cdot 3^{\alpha-1}$-approximation algorithm for the TSP if the edge weights are Euclidean edge weights to the power α.*

The metric with Euclidean edge weights to the power α satisfies the relaxed triangle inequality with $\tau = 2^{\alpha-1}$. A short computation shows that our algorithm improves previous algorithms [BC00, And01] for small α, i.e. for $2 \leq \alpha \leq 2.7$.

2.7 Conclusion

We have shown how to solve a variety of problems occurring in the design of wireless networks. Additionaly we have proposed new network design models like networks in which broadcast operations can be performed but in which only a few senders are actually sending. It is our believe that such a design can be of great benefit for the efficiency of wireless networks. The most important research direction to follow in the near future is to design simple and *distributed* algorithms for some of the problems like the k-SEMBC problem. One possible idea could be to construct distributedly a low-weight k-disk-cover of the network and then connect the components in some way such that the overall cost does not increase too much – very much in the spirit of our $O(1)$-approximation algorithm. One major drawback of the $(1+\epsilon)$-approximation algorithms presented is that it still requires time exponential in the number of senders k. It is unclear whether this exponential dependence could be removed.

Bibliography

[AAB+06] Helmut Alt, Esther M. Arkin, Hervé Brönnimann, Jeff Erickson, Sándor P. Fekete, Christian Knauer, Jonathan Lenchner, Joseph S. B. Mitchell, and Kim Whittlesey. Minimum-cost coverage of point sets by disks. In *Proceedings of the 22nd ACM Symposium on Computational Geometry (SCG), Sedona, Arizona, USA*, pages 449–458. ACM, 2006.

[ACI+04] Christoph Ambühl, Andrea E. F. Clementi, Miriam Di Ianni, Nissan Lev-Tov, Angelo Monti, David Peleg, Gianluca Rossi, and Riccardo Silvestri. Efficient algorithms for low-energy bounded-hop broadcast in ad-hoc wireless networks. In *Proceedings of the 21st Annual Symposium on Theoretical Aspects of Computer Science (STACS), Montpellier, France*, Lecture Notes in Computer Science, pages 418–427. Springer, 2004.

[Amb05] Christoph Ambühl. An optimal bound for the mst algorithm to compute energy-efficient broadcast trees in wireless networks. In *Automata, Languages and Programming, 32nd International Colloquium (ICALP), Lisbon, Portugal*, volume 3580 of *Lecture Notes in Computer Science*, pages 1139–1150. Springer, 2005.

[AN06] Ernst Althaus and Rouven Naujoks. Computing steiner minimal trees in hamming metric. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Miami, USA*, pages 172–181. ACM Press, 2006.

[AN08] Ernst Althaus and Rouven Naujoks. Reconstructing phylogenetic networks with one recombination. In *Experimental Algorithms, 7th International Workshop (WEA), Provincetown, MA, USA*, volume 5038 of *Lecture Notes in Computer Science*, pages 275–288. Springer, 2008.

[And01] Thomas Andreae. On the traveling salesman problem restricted to inputs satisfying a relaxed triangle inequality. In *Networks*, volume 38, pages 59–67. John Wiley & Sons, 2001.

[BC00] Michael A. Bender and Chandra Chekuri. Performance guarantees for the TSP with a parameterized triangle inequality. In *Information Processing*

Letters, volume 73, pages 17–21. Elsevier Science Publishers B.V. (North Holland), 2000.

[BCKK05] Vittorio Bilò, Ioannis Caragiannis, Christos Kaklamanis, and Panagiotis Kanellopoulos. Geometric clustering to minimize the sum of cluster sizes. In *Algorithms, 13th Annual European Symposium (ESA), Palma de Mallorca, Spain*, volume 3669, pages 460–471. Springer, 2005.

[CCP+01] Andrea E. F. Clementi, Pierluigi Crescenzi, Paolo Penna, Gianluca Rossi, and Paola Vocca. On the complexity of computing minimum energy consumption broadcast subgraphs. In *Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS), Dresden, Germany*, volume 2010 of *Lecture Notes in Computer Science*, pages 121–131. Springer, 2001.

[CGH03] E.R. Chare, E.A. Gould, and E.C. Holmes. Phylogenetic analysis reveals a low rate of homologous recombination in negative-sense rna viruses. In *Journal of General Virology*, volume 84, pages 2691–2703, 2003.

[CHP+02] Andrea E.F. Clementi, Gurvan Huiban, Paolo Penna, Gianluca Rossi, and Yann C. Verhoeven. Some recent theoretical advances and open questions on energy consumption in ad-hoc wireless networks. In *Proc. 3rd Workshop on Approximation and Randomization Algorithms in Communication Networks (ARACNE), pages 23-38.*, volume 15 of *Proceedings in Informatics*. Carleton Scientific, 2002.

[CHR+03] Andrea E. F. Clementi, Gurvan Huiban, Gianluca Rossi, Yann C. Verhoeven, and Paolo Penna. On the approximation ratio of the mst-based heuristic for the energy-efficient broadcast problem in static ad-hoc radio networks. In *17th International Parallel and Distributed Processing Symposium (IPDPS), Los Alamitos, CA, USA*, pages 222–222. IEEE Computer Society, 2003.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[EJS01] Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric graphs. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA), Washington, DC, USA*, pages 671–679. ACM Press, 2001.

[Fel04] J. Felsenstein. PHYLIP (phylogeny inference package) version 3.6, 2004.

[FG82] L.R. Foulds and R.L. Graham. The Steiner problem in phylogeny is NP-complete. In *Advances in Applied Mathematics*, volume 3, pages 43–49, 1982.

[Fit71] Walter M. Fitch. Toward defining the course of evolution: minimum change for a specified tree topology. In *Systematic Zoology*, volume 20. Taylor and Francis, Ltd., 1971.

[FL07] Stefan Funke and Sören Laue. Bounded-hop energy-efficient broadcast in low-dimensional metrics via coresets. In *24th Annual Symposium on Theoretical Aspects of Computer Science (STACS), Aachen, Germany*, volume 4393 of *Lecture Notes in Computer Science*, pages 272–283. Springer, 2007.

[FLN07] Stefan Funke, Sören Laue, and Rouven Naujoks. Minimum-energy broadcast with few senders. In *Distributed Computing in Sensor Systems (DCOSS), Third IEEE International Conference, Santa Fe, NM, USA*, volume 4549 of *Lecture Notes in Computer Science*, pages 404–416. Springer, 2007.

[FLNL08] Stefan Funke, Sören Laue, Rouven Naujoks, and Zvi Lotker. Power assignment problems in wireless communication: Covering points by disks, reaching few receivers quickly, and energy-efficient travelling salesman tours. In *Distributed Computing in Sensor Systems (DCOSS), 4th IEEE International Conference, Santorini Island, Greece*, volume 5067 of *Lecture Notes in Computer Science*, pages 282–295. Springer, 2008.

[FNKP04] Michele Flammini, Alfredo Navarra, Ralf Klasing, and Stéphane Pérennes. Improved approximation results for the minimum energy broadcasting problem. In *Proceedings of the DIALM-POMC Joint Workshop on Foundations of Mobile Computing, Philadelphia, PA, USA*, pages 85–91. ACM, 2004.

[GEL03] Dan Gusfield, Satish Eddhu, and Charles Langley. Efficient reconstruction of phylogenetic networks with constrained recombination. In *Proceedings of the IEEE Computer Society Conference on Bioinformatics*, pages 363–374. IEEE Computer Society, 2003.

[GK99] Sudipto Guha and Samir Khuller. Improved methods for approximating node weighted steiner trees and connected dominating sets. In *Information and Computation*, volume 150, pages 57–74, 1999.

[GR00] A. Zelikovsky G. Robins. Improved steiner tree approximation in graphs. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), San Francisco, CA, USA*, pages 770–779. ACM Press, 2000.

[Hei93] Jotun Hein. A heuristic method to reconstruct the history of sequences subject to recombination. *J. Mol. Evol.*, 36, 1993.

[HHPM05] B. R. Holland, K. T. Huber, D. Penny, and V. Moulton. The minmax squeeze: Guaranteeing a minimal tree for population data. *Molecular Biology and Evolution*, 22(2), 2005.

[HK05] Daniel H. Huson and Tobias H. Kloepper. Computing recombination networks from binary sequences. *Bioinformatics*, 21(2):159–165, 2005.

[HP82] M. D. Hendy and D. Penny. Branch and bound algorithms to determine minimal evolutionary trees. *Mathematical Biosciences*, 59, 1982.

[HP04] Sariel Har-Peled. Clustering motion. In *Discrete & Computational Geometry*, volume 31, pages 545–565, 2004.

[JNST05] Guohua Jin, Luay Nakhleh, Sagi Snir, and Tamir Tuller. Inferring phylogenetic networks by the maximum parsimony criterion: A case study. *Molecular Biology and Evolution*, 24(1):324–337, 2005.

[JNST07] Guohua Jin, Luay Nakhleh, Sagi Snir, and Tamir Tuller. Efficient parsimony-based methods for phylogenetic network reconstruction. *Bioinformatics*, 23(2):123–128, 2007.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[KG98] John Kececioglu and Dan Gusfield. Reconstructing a history of recombinations from a set of sequences. In *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science (DAMATH)*, volume 88, pages 239–260. Elsevier Science Publishers B.V. (North Holland), 1998.

[KKKP00] Lefteris M. Kirousis, Evangelos Kranakis, Danny Krizanc, and Andrzej Pelc. Power consumption in packet radio networks. In *Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science (STACS), Lübeck, Germany*, volume 243, pages 289–305. Elsevier Science B.V., 2000.

[KTND07] S. Kumar, K. Tamura, M. Nei, and J. Dudley. Mega4: Molecular evolutionary genetics analysis (MEGA) software version 4.0. In *Molecular Biology and Evolution*, volume 24, pages 1596–1599, 2007.

[Meh88] K. Mehlhorn. A faster approximation algorithm for the steiner problem in graphs. In *Information Processing Letters*, volume 27, pages 125–128. North-Holland Publishing Company, 1988.

[MKL06] V. Makarenkov, D. Kevorkov, and P. Legendre. Phylogenetic network reconstruction approaches. In *Applied Mycology and Biotechnology - Bioinformatics*, volume 6 of *International Elsevier Series*, pages 61–97. Oxford University Press, 2006.

[ML06] Jochen Maydt and Thomas Lengauer. Recco: recombination analysis using cost optimization. In *Bioinformatics*, volume 22, pages 1064–1071. Oxford University Press, 2006.

[MM06] M.M. Morin and B.M.E. Moret. NetGen: generating phylogenetic networks with diploid hybrids. In *Bioinformatics*, volume 22, pages 1921–1923, 2006.

[MN99] Kurt Mehlhorn and Stefan Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[MS07] Jens Maue and Peter Sanders. Engineering algorithms for approximate weighted matching. In *Experimental Algorithms, 6th International Workshop (WEA), Rome, Italy*, volume 4525 of *Lecture Notes in Computer Science*, pages 242–255. Springer, 2007.

[NJZMC05] Luay Nakhleh, Guohua Jin, Fengmei Zhao, and John Mellor-Crummey. Reconstructing phylogenetic networks using maximum parsimony. In *Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference, Stanford*, 2005.

[NK00] N. Nei and S. Kumar. *Molecular Evolution and Phylogenetics.* Oxford University Press, 2000.

[OM] Pekka Orponen and Heikki Mannila. On approximation preserving reductions: Complete problems and robust measures. Technical Report, University of Helsinki, 1990.

[PCH02] David Posada, Keith A. Crandall, and Edward C. Holmes. Recombination in evolutionary genomics. In *Annual Review of Genetics*, volume 36, pages 75–97, 2002.

[Pol03] Tobias Polzin. *Algorithms for the Steiner Problem in Networks.* PhD thesis, Universität des Saarlandes, 2003.

[RG97] Andrew Rambaut and N. C. Grassly. Seq-gen: an application for the monte carlo simulation of dna sequence evolution along phylogenetic trees. volume 13, pages 235–238, 1997.

[RG01] Vincent Ranwez and Olivier Gascuel. Quartet based phylogenetic inference: improvement and limits. In *Molecular Biology and Evolution*, volume 18, pages 1103–1116, 2001. http://www.lirmm.fr/ ranwez/.

[RN05] D. Ruths and L. Nakhleh. Recombination and phylogeny: Effects and detection. In *International Journal on Bioinformatics Research and Applications*, volume 1, pages 202–212, 2005.

[Sch86] Schreiber. Zur geschichte des sogenannten steiner-weber-problems. In *Wiss. Zeitschr. Ernst-Moritz-Arndt-Univ. Greifswald, Math.-Nat. Reihe Bd. 35 Heft 3, 53-58*, 1986.

[Ste05] Mike Steel. Should phylogenetic models be trying to 'fit an elephant'? *Trends in Genetics*, 21(6):307–309, 2005.

[Swo03] D. L Swofford. PAUP*. Phylogenetic Analysis Using Parsimony (*and other methods). Version 4., 2003.

[TD88] T.Feder and D.Greene. Optimal algorithms for approximate clustering. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC), Chicago, IL, USA*, pages 434–444. ACM Press, 1988.

[Vaz03] Vijay Vazirani. *Approximation Algorithms.* Springer, 2003.

[WCLF01] Peng-Jun Wan, Gruia Calinescu, Xiang-Yang Li, and Ophir Frieder. Minimum-energy broadcast routing in static ad hoc wireless networks. In *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Los Alamitos, CA, USA*, pages 1162–1171. IEEE Computer Society, 2001.

[WEBa] http://evolution.genetics.washington.edu/phylip/software.html.

[WEBb] http://www.bioinf.manchester.ac.uk/recombination/programs.shtml.

[WEBc] http://evolution.genetics.washington.edu/phylip/software.html#Recombinant.

[WNE00] Jeffrey E. Wieselthier, Gam D. Nguyen, and Anthony Ephremides. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *Proceedings of the 2000 IEEE Computer and Communications Societies Conference on Computer Communications (INFOCOM), Tel-Aviv, Israel*, pages 585–594. IEEE Computer Society, 2000.

[WWZ00] D. M. Warme, P. Winter, and M Zachariasen. Exact algorithms for plane steiner tree problems: A computational study. In *Advances in Steiner Trees*, pages 81–116. Kluwer Academic Publishers, 2000.

[WZZ01] Lusheng Wang, Kaizhong Zhang, and Louxin Zhang. Perfect phylogenetic networks with recombination. In *Journal of Computational Biology*, volume 8, pages 69–78, 2001.

[Zac99] Zachariasen. Rectilinear full steiner tree generation. In *Networks*, volume 33, pages 125–143. John Wiley & Sons, 1999.