

INAUGURAL-DISSERTATION
submitted
to the
Combined Faculty of Mathematics, Engineering and Natural Sciences
of
Ruprecht–Karls–University, Heidelberg
for the degree of
Doctor of Natural Sciences

Put forward by

M.Sc. Enrique Fita Sanmartín

Born in: Quart de Poblet, Spain

Oral examination:

Exploring Graph Structures: Learning and Analysis

Advisor: Prof. Dr. Fred A. Hamprecht
Prof. Dr. Christoph Schnörr

This work is licensed under a Creative Commons “Attribution-NonCommercial-NoDerivs 3.0 Unported” license.



Abstract

Graphs are versatile data structures with great flexibility to represent various kinds of information due to their capability to model relationships between entities. In this work, we explore diverse applications of graphs in machine learning, with a particular focus on spanning trees.

First, we introduce the “Directed Probabilistic Watershed” algorithm, a graph-based semi-supervised learning method for classification on directed graphs. This algorithm builds a probability distribution over directed forests to infer the label of a node. It does this by considering the probability of a directed forest connecting the node to one of the given labeled nodes within the same subtree.

Next, building on a similar probabilistic framework, we investigate the node degree distribution in a random spanning tree. We provide analytical expressions for both the expected value and variance of the degree of a node in a spanning tree.

Subsequently, we analyze different node metrics in graphs, focusing on distances computed based on the paths connecting nodes. We frame several popular distance metrics, namely the shortest path distance, the commute cost distance, the minimax distance and the potential distance, as instances of the “algebraic path problem”, which is a generalization of the shortest path problem. We introduce the “log-norm” distance, a node metric that interpolates between the aforementioned metrics. Furthermore, we establish certain conditions that are sufficient and necessary for an algebraic path problem to define a metric across any given graph.

Finally, returning to the study of spanning trees, we focus on the geometrical stability. We introduce the “central spanning tree,” a parameterized family of spanning trees embedded in Euclidean space. By conducting empirical tests, we showcase its resilience against perturbations such as noise in node coordinates. We formulate the central spanning tree as a minimization problem, establishing connections to other well-known spanning tree problems such as the minimum spanning tree or the Euclidean Steiner tree. Additionally, we explore a variant of the central spanning tree, referred to as “branched central spanning tree”, that allows for the inclusion of Steiner points. We demonstrate the efficacy of this variant in modeling the skeleton of 3D point clouds representing plants.

Zusammenfassung

Graphen sind vielseitige Datenstrukturen, die aufgrund ihrer Fähigkeit, Beziehungen zwischen Entitäten zu modellieren, große Flexibilität bei der Darstellung verschiedener Arten von Informationen bieten. In dieser Arbeit werden verschiedene Anwendungen von Graphen im Bereich des maschinellen Lernens untersucht, wobei ein besonderer Schwerpunkt auf Spannbäumen liegt.

Zuerst stellen wir den “Directed Probabilistic Watershed” Algorithmus vor, ein graphenbasiertes, teilüberwachtes Lernverfahren für die Klassifizierung von gerichteten Graphen. Dieser Algorithmus baut eine Wahrscheinlichkeitsverteilung über gerichtete Wälder auf, um das Label eines Knotens abzuleiten. Er tut dies, indem er die Wahrscheinlichkeit eines gerichteten Waldes berücksichtigt, der den Knoten mit einem der gegebenen gelabelten Knoten innerhalb desselben Teilbaums verbindet.

Als Nächstes untersuchen wir, aufbauend auf einem ähnlichen probabilistischen Rahmen, die Knotengradverteilung in einem zufälligen Spannbaum. Wir liefern analytische Ausdrücke sowohl für den Erwartungswert als auch für die Varianz des Grades eines Knotens in einem Spannbaum.

Anschließend analysieren wir verschiedene Knotenmetriken in Graphen, wobei wir uns auf Entfernungen konzentrieren, die auf der Grundlage der Verbindungswege zwischen Knoten berechnet werden. Wir betrachten mehrere populäre Distanzmetriken, nämlich die kürzeste Pfaddistanz, die “Commuter Cost”-Distanz, die Minimax-Distanz und die Potentiality, als Instanzen des algebraischen Pfadproblems, das eine Verallgemeinerung des Problems des kürzesten Pfades ist. Wir führen die “Log-Norm” Distanz ein, eine umfassende Knotenmetrik, die zwischen den oben genannten Metriken interpoliert. Darüber hinaus legen wir bestimmte Bedingungen fest, die hinreichend und notwendig sind, damit ein Algebraisches Pfadproblem eine Metrik über einen beliebigen gegebenen Graphen definiert.

Schließlich kehren wir zur Untersuchung von Spannbäumen zurück und konzentrieren uns auf die geometrische Stabilität. Wir stellen den “zentralen Spannbaum” vor, eine parametrisierte Familie von Spannbäumen, die in den euklidischen Raum eingebettet sind. Durch die Durchführung empirischer Tests zeigen wir seine Widerstandsfähigkeit gegenüber Störungen wie Rauschen in den Knotenkoordinaten. Wir formulieren den zentralen Spannbaum als

ein Minimierungsproblem und stellen Verbindungen zu anderen bekannten Spannbaumproblemen wie dem minimalen Spannbaum oder dem euklidischen Steiner-Baum her. Zusätzlich untersuchen wir eine Variante des zentralen Spannbaums, den so genannten “Verzweigten zentralen Spannbaum”, die das Hinzufügen von Steiner-Punkten erlaubt. Wir demonstrieren die Wirksamkeit dieser Variante bei der Modellierung des Skeletts von 3D-Punktwolken, die Pflanzen darstellen.

Acknowledgements

Firstly, I express my gratitude to my supervisor Fred Hamprecht for offering me the opportunity to pursue my Ph.D. within his research group. Throughout my time in the lab, I am truly thankful for his guidance, unwavering support, and mentorship. The discussions with him and the advice he provided were consistently inspiring and fruitful, enabling me to advance in my research. I also appreciate his wide range of interests and the autonomy he provided in the choice of research topics, which allowed me to delve more into the theoretical aspects that interested me the most.

I likewise extend my gratitude to my second supervisor, Christoph Schnörr, for his invaluable support, expertise, and insightful idea exchanges, which significantly contributed to the advancement of my research.

Being part of the Scientific AI Lab, formerly known as the Image Analysis and Learning group, has been a pleasure. I have had the opportunity to collaborate with brilliant colleagues knowledgeable in various fields. Overall, I am thankful to all current and former group members who provided a great working environment and were always ready to offer their help and sympathy. I would like to highlight the engaging lunch conversations and event nights. Special thanks go to Sebastian Damrich for the insightful conversations that significantly influenced my work, to Peter Lippmann and Fabian Egersdörfer for the discussions regarding the central spanning tree problem, and to Ocima Kamboj for proofreading parts of this thesis.

I want to extend my gratitude to Barbara Werner, whose knowledge about the bureaucracy intricacies simplified all administrative formalities.

I am also grateful to all the friends I have made in Heidelberg, specially to Ashis, Abhinav, Sreedev, Ram, Shreya and Ramia for all the dinners and time we have spent together.

Le doy las gracias a Lea por su apoyo y por fregar los platos en los momentos que tenía mucho trabajo. Mein Dank gilt auch ihrer Familie, die mich mit Essen versorgte, damit ich bei Kräften bleibe.

A los amigos de Quart de Poblet, quienes, aunque no tengan una idea precisa de lo que hago en Alemania, cada vez que los he visitado, ha sido como si nunca me hubiera ido. Especial mención a Ferran, que constantemente me ha hecho de relaciones públicas,

permitiendo así que pudiera seguir ignorando el Whatsapp.

Per últim, vull donar-li les gràcies a la meua família, ja que sense ells no estaria ací. Al meu germà Nofre per sempre recordar-me, a la seua manera, que hi ha espai per millorar. A mon tio Agustí pels estudis a l'estiu, el suport tècnic i més coses que farien una llista molt llarga. A mons tios Pascual i Pili per les paelles. A mons pares Onofre i Loles per tot.

Contents

Abstract	vii
Zusammenfassung	ix
Acknowledgements	xi
Contents	xiii
1 Introduction	1
1.1 Semi-Supervised Learning on Graphs	2
1.2 Spanning Trees	4
1.3 Graph Node Distances	7
1.4 Thesis Outline	8
1.5 List of Publications	9
2 Directed Probabilistic Watershed	11
2.1 Introduction	11
2.1.1 Related Work	13
2.2 Background	14
2.2.1 Notation and Terminology	14
2.2.2 Matrix Tree Theorem	15
2.2.3 Random Walker and Power Watershed	16
2.2.4 Probabilistic Watershed Review	17
2.3 Directed Probabilistic Watershed	18
2.3.1 Gibbs Probability Distribution	18
2.3.2 Computation of the Directed Probabilistic Watershed Probabilities .	19
2.4 Equivalence of DProbWS and the Directed Random Walker	21
2.4.1 Teleporting Random Walker	23
2.5 Directed Power Watershed	25
2.6 Experiments	26

2.7	Conclusion	27
3	Expected Degree and Variance in Random Spanning Trees	29
3.1	Introduction	29
3.2	Notation	31
3.3	Polynomial-Based Computation of Expectation and Variance of Node Degree in Spanning Trees	33
3.4	Laplacian-Based Computation of Expectation and Variance of Node Degree in Spanning Trees	35
3.5	Relation Between Edge Probability and Expected Node Degree in Spanning Trees	40
3.6	Extension to Directed Graphs	43
3.7	Toy Examples	43
3.8	Conclusion	45
4	Algebraic Path Problem for Graph Metrics	47
4.1	Introduction	47
4.1.1	Related Work	50
4.2	Preliminaries	50
4.2.1	Semirings	50
4.2.2	Graph Notation	51
4.2.3	Algebraic Path Problem	52
4.2.4	Semiring Distances	53
4.3	Log-Norm Distances	54
4.3.1	Shortest Path and Minimax Distance Interpolation	54
4.3.2	Commute Cost and Shortest Path Distance Interpolation	55
4.3.3	The Family of Log-Norm Distances	56
4.4	When Does a Semiring Define a Distance?	59
4.4.1	Identity of Indiscernibles	60
4.4.2	Triangle Inequality	60
4.5	Conclusion	64
5	Central Spanning Tree	65
5.1	Introduction	65
5.2	Limit Cases of the CST/BCST Problems Beyond the Range $\alpha \in [0, 1]$	69
5.2.1	Limit Cases Where the Optimum (B)CST Transforms into a Star-Tree	69
5.2.2	Limit Cases Where the Optimum (B)CST Transforms into a Path-Tree	72
5.3	Stability of the CST Problem	75

5.3.1	Toy Data	75
5.3.2	Real-World Data	76
5.4	Correspondence Between the BCST and CST Topologies	78
5.5	Geometry of Optimal BCST Topologies	80
5.5.1	Branching Angles at the Steiner Points	81
5.5.2	Infeasibility of Degree-4 Steiner Points in the Plane	83
5.6	CST and BCST Optimization Algorithm	87
5.6.1	Geometry Optimization	87
5.6.2	Heuristic Optimizer for the (B)CST Problem	88
5.7	Benchmark	90
5.7.1	Brute Force Benchmark	90
5.7.2	Steiner and MRCT Benchmark	91
5.7.3	Comparing GRASP_PR and mSTreg for the CST Problem	92
5.8	Conclusion	93
6	BCST-Based Skeletonization with BCST	97
6.1	Introduction	97
6.2	Skeletonization Method	100
6.2.1	Pruning Optimization Problem: Formulating the Objective	100
6.2.2	Pruning Algorithm Based on the Prize Collecting Steiner Tree Problem	102
6.2.3	Implementation Details	105
6.3	Experiments	107
6.3.1	Metrics	108
6.3.2	Experimental Details and Results	109
6.4	Conclusion	113
7	Conclusion	115
A	Directed Probabilistic Watershed	119
A.1	Directed Random Walker	119
A.2	Efficient Computation of the DProbWStrw Probabilities	122
A.3	Further Experiment Details	125
A.3.1	Reference Methods	125
A.3.2	Datasets	126
A.3.3	k-Nearest Neighbor Graph Construction	127
B	Expected Degree and Variance in Random Spanning Trees	129
B.1	Proof of Theorem 3.5.2	129

C Algebraic Path Problem for Graph Metrics	131
C.1 Min-Norm Semiring	131
C.2 APP of the Eisner Semiring Recovers the First Hitting Cost	132
C.3 APP of the Log-Semiring Recovers the Potential Distance	132
C.4 Log-Norm Strong Bimonoid	133
C.5 Log-Norm Metric Limits	136
C.6 Proofs of Section 4.4	137
C.6.1 Proof Lemma 4.4.2	137
C.6.2 Proof Theorem 4.4.6	138
C.7 Use Case of the Results in Section 4.4	139
C.8 Log-Norm Distance	143
C.9 Log-Norm Distance and the Randomized Shortest Paths	145
C.10 Exp-Max and Log-Max Metric Computation	147
D Central Spanning Tree	151
D.1 Stability Examples	151
D.2 Reinterpreting CST as a Minimum Concave Cost Flow	152
D.2.1 Relation to the Branched Optimal Transport Problem	155
D.3 Equivalence of the CST Problem with $\alpha = 1$ and the Minimum Routing Cost Tree Problem	156
D.4 Limit Cases of the CST/BCST Problems Beyond the Range $\alpha \in [0, 1]$	157
D.4.1 Proof Theorem 5.2.2	157
D.4.2 Proof $h_1(\ell, N, \alpha) > 1$ as N Approaches Infinity, for $\alpha > 1$	160
D.4.3 Computation $\alpha^*(N)$	161
D.4.4 Proof Theorem 5.2.5	161
D.5 Exploring the Number of Derivable Topologies from CST and BCST Topolo- gies	166
D.5.1 Number of BCST Topologies Derivable from a CST Topology	166
D.5.2 Number of CST Topologies Derivable from a BCST Topology	166
D.6 Branching Angles at the Steiner Points in the BCST Problem	168
D.6.1 Steiner Point b Does Not Collapse with a Terminal	169
D.6.2 Steiner Point b Collapses with a Terminal	169
D.7 Infeasibility of Degree-4 Steiner Points in the Plane for $\alpha = 1$	171
D.8 Iteratively Reweighted Least Square for the Geometric Optimization of the Steiner Points	174
D.9 Complexity mSTreg Heuristic	175
D.10 Effect of Additional Intermediate Points in the mSTreg Heuristic	177

D.11 Strategies to Transform a Full Tree Topology into a CST Topology	178
D.12 Further Details on the Brute Force Experiment	179
D.13 Selection of α	179
D.14 Implementation Details	183
E BCST-Based 3D Plant Skeletonization	185
E.1 Additional Skeletonization Results	185
Bibliography	191
List of Figures	209
List of Tables	211
List of Algorithms	213

Chapter 1

Introduction

In recent years, machine learning has experienced an unprecedented growth, transitioning from a tool primarily utilized in lab experiments to becoming a fundamental component of myriad pipelines spanning across academic and industrial domains. While the origins of machine learning date back to the 1950s, its recent success can not be fully understood without the advancements in computational power and data storage technology. These advancements foster the acquisition and processing of data, leading to more efficient machine learning algorithms.

Data comes in various modalities and formats, including images, text, tables, and more. However, machine learning algorithms do not directly perceive images or read text directly; instead, they rely on representations of the data that are compatible with their processing mechanisms. One of these data representations are graphs.

Formally, a graph is defined as a pair of sets (V, E) , where V is a set of vertices (also known as nodes) and $E \subseteq V \times V$ is a set of edges. Each edge of E represents a connection between two vertices of V . By linking vertices through edges, graphs are capable of capturing pairwise relations between different entities or objects. The inception of graph theory can be attributed to Euler [55], who laid its foundations while addressing the conundrum of the seven bridges of Königsberg. The riddle centered on the existence of a route that traversed each bridge once and only once, to which he provided a negative response.

Today, graphs are used not only to represent spatial data, but also to model various types of information. The ability to model one-to-one connections between entities, represented by nodes, is what makes graphs powerful and flexible in data representation. Indeed, data often comprises individual components that are related to each other in some manner. For instance, both sentences and images are composed of interconnected elements—words and pixels, respectively—that collectively convey meaning. Similarly, social networks consist of individuals linked by relationships like friendship, and molecules are assemblies of atoms bonded together (see Figure 1.1).

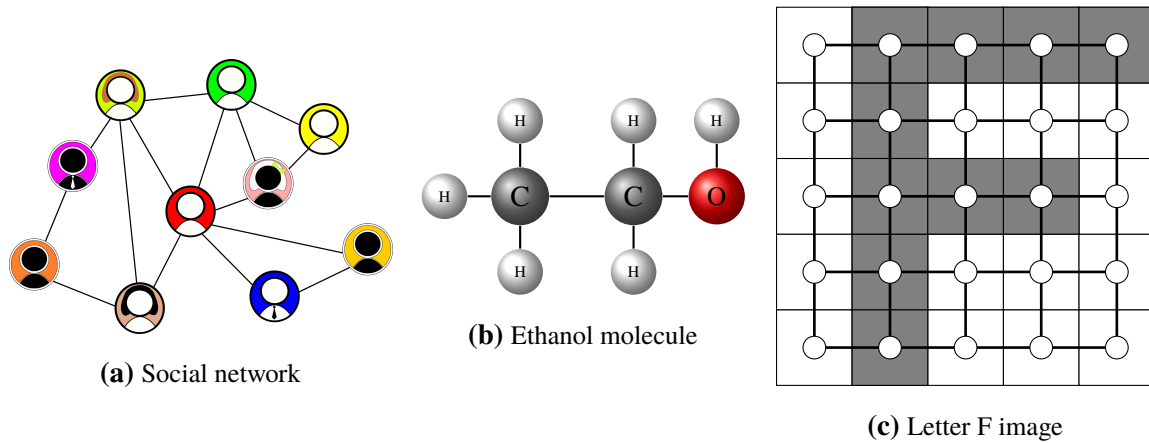


Figure 1.1. Data Represented as Graphs: Various data types can be modeled as graphs. **(a)** Social network: nodes represent people, and edges represent friendships. **(b)** Ethanol molecule: nodes represent atoms, and edges represent chemical bonds. **(c)** Letter F image: nodes represent pixels, and edges connect neighboring pixels.

The representation capability of graphs increases when additional attributes are incorporated. For instance, weighted graphs assign weights or costs to the edges, which can model the intensity or penalty associated with each edge. Directed graphs show unidirectional connections from one vertex to another, enabling the modeling of directed relations, such as those found in citation networks. Node-attributed graphs associate features to each node, enriching the representation with additional information about the entities portrayed by the nodes. Moreover, graphs are advantageous due to their often sparse structure, which allows for efficient processing.

Graphs' flexibility in modeling various forms of data has made them ubiquitous in machine learning, leading to the development of algorithms across different subareas including clustering [21, 56], classification [31, 66], dimensionality reduction [4, 14] and many more [181]. Additionally, specialized artificial neural networks, known as graph neural networks, have emerged as a prominent tool for processing graphs [180, 195].

Through this thesis we will touch on the use of graphs for semi-supervised learning (Chapter 2), node metrics on graphs (Chapter 4) and data skeletonization (Chapters 5 and 6).

1.1 Semi-Supervised Learning on Graphs

Machine learning is often divided into supervised and unsupervised learning. Supervised learning involves learning from labeled data, denoted as $\{(x_i, y_i)\}_{i=1}^n$. In this setup, each data point x_i in the training set is paired with a label y_i that corresponds to the desired output. This pairing assists the model in predicting accurate responses for new, unseen data.

In contrast to supervised learning, unsupervised learning operates without access to labels that guide the learning process. Instead, it focuses on uncovering the underlying structure of the data and extracting meaningful information from it.

Semi-supervised learning has characteristics of both, supervised and unsupervised learning. Similar to supervised learning, semi-supervised learning utilizes data with labels. However, a distinctive feature is that only a small subset of the data is labeled. That is, the data takes the form of $\{(x_i, y_i)\}_{i=1}^{n_l}, \{x_i\}_{i=1}^{n_u}\}$, where n_l is much smaller than n_u . Therefore, semi-supervised learning leverages the labeled data with the unlabeled one to generate appropriate responses.

While the availability of data has increased in recent years, labeled data remains scarce in specialized fields such as healthcare or bioinformatics, where only experts can effectively annotate the data. Therefore, semi-supervised learning continues to be a relevant field with numerous applications [53, 145].

In graph semi-supervised learning, algorithms exploit the structure of the graph along with the labeled data. They typically operate under the assumption of homophily, where connected nodes are more likely to share similarities and consequently have the same label. Different methods exploit homophily in various ways, including regularization techniques [24, 66, 192], embedding or low-rank representation techniques [13, 68, 178, 185], and Graph Neural Network (GNN) based models [93, 185]. We refer to [39, 158] for a more complete overview.

Within graph semi-supervised learning, two main subcategories are distinguished: transductive and inductive semi-supervised learning. In a transductive setting, the objective is to learn a function f that predicts labels exclusively for the unlabeled data $\{x_i\}_{i=1}^{n_u}$ within the provided graph. Consequently, the algorithm is confined to generalizing solely within the original graph and cannot extend to data outside its scope. In contrast, inductive semi-supervised learning, aims to learn a function f capable of predicting labels for nodes of graphs not present in the training data. This allows the algorithm to generalize beyond the original graph structure (see Figure 1.2).

Most classic graph-based semi-supervised learning algorithms belong to the transductive category [192, 193]. However, there are also a few inductive ones [85]. Recent models leveraging Graph Neural Networks (GNNs) tend to fall into the inductive category as well [93].

In Chapter 2, we propose a transductive graph-based semi-supervised learning algorithm for node classification in directed graphs.

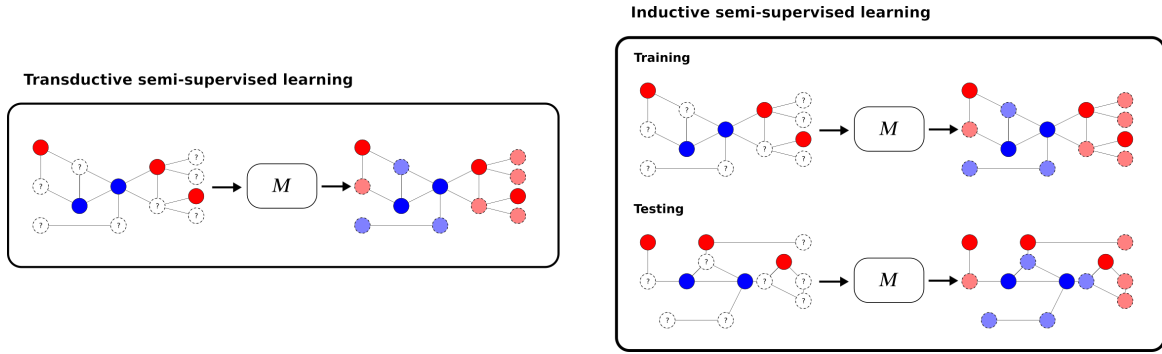


Figure 1.2. Comparison Between Transductive and Inductive Semi-Supervised Learning. In the transductive setting, only the labels of unlabeled nodes in the training dataset are inferred, whereas in the inductive setting, the trained model M can predict the labels of unseen nodes, allowing its application to previously unseen graphs.

1.2 Spanning Trees

Trees are a very special type of undirected graphs. To define a tree, it is essential to understand what a connected graph and a cycle are. We say that a graph is connected, if there exists a path φ connecting any pair of nodes s, t . In other words, there exists a sequence of nodes $\varphi = (v_0, v_1, \dots, v_k)$ such that $(v_i, v_{i+1}) \in E$, where $v_0 = s$ and $v_k = t$. A path is simple if no node is repeated ($v_i \neq v_j$ for all $i \neq j$). A cycle is a simple path with more than one node where the starting and end nodes are equal, that is, $v_0 = v_k$.

Definition 1. A **tree**, denoted as T , is an undirected graph that satisfies **one** of the following conditions:¹

1. T is connected and acyclic (does not contain cycles).
2. T is acyclic and the addition of a single edge generates a cycle.
3. T is connected and the removal of an edge disconnects the tree.
4. Any two vertices of T are connected by a unique simple path, that is a path where every node is visited once.
5. T is connected and has $n - 1$ edges.
6. T is acyclic and has $n - 1$ edges.

The equivalence of these definitions is proven in Theorem 9 of [173]. When a graph is disconnected but lacks cycles, it is referred to as a forest.

Given a graph $G = (V_G, E_G)$, we say that a tree $T = (V_T, E_T)$ is a spanning subtree of G if $V_T = V_G$ and $E_T \subset E_G$. A spanning forest is defined accordingly. One of the most well-known spanning trees is the maximum/minimum spanning tree, which, for a weighted graph,

¹We assume a finite graph, that is with a finite number of nodes. If the graph has an infinite number of nodes then definition 5 and 6 are not equivalent to the previous ones.

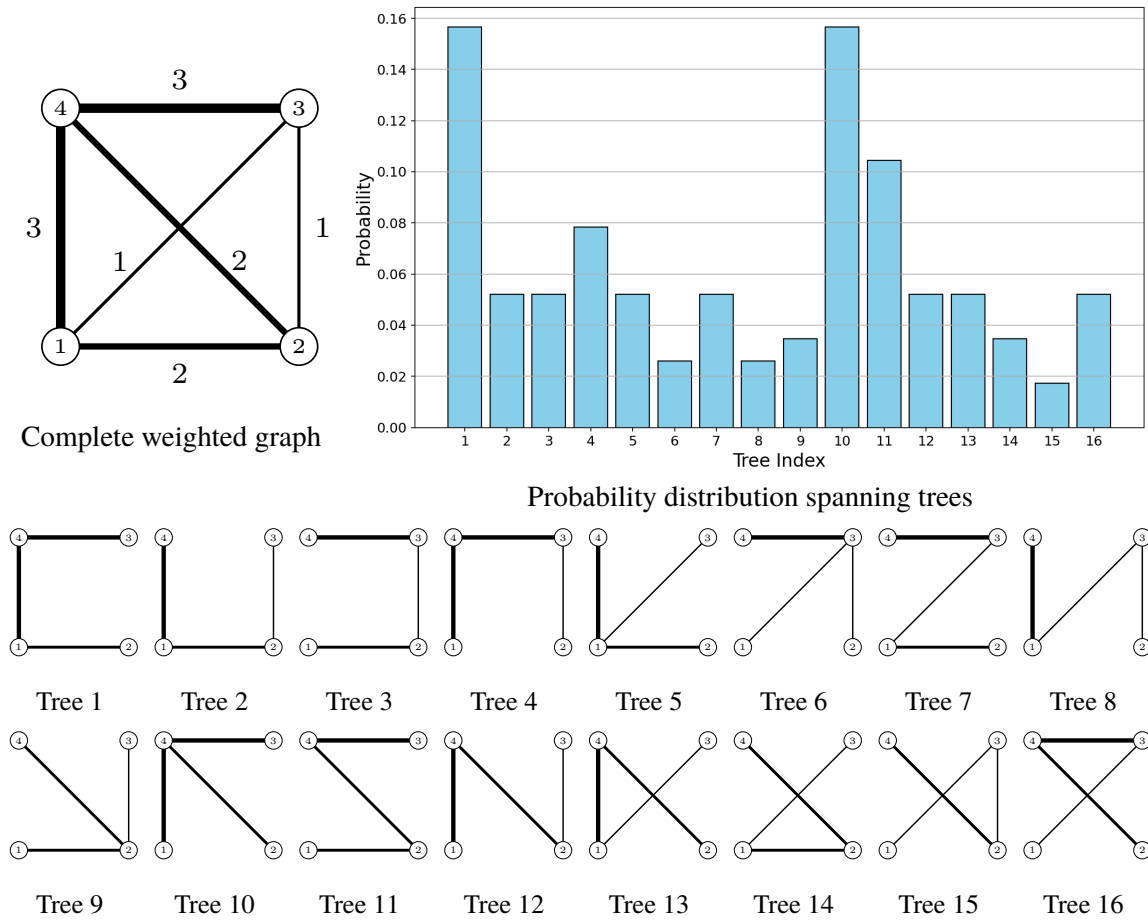


Figure 1.3. Probability Distribution over Spanning Trees of a Complete Graph with 4 Nodes. **Top left:** Complete weighted graph where the edge width is proportional to its weight, indicated next to the edge. **Top right:** Probability distribution over trees where the probability of a tree is proportional to the product of the weights of the edges composing the tree. **Bottom rows:** List of all spanning trees. The trees with higher probabilities correspond to Tree 1 and 10, as they contain the highest combinations of edge weights. Conversely, Tree 15 has the lowest probability because it is composed of the lowest edge weights.

maximizes/minimizes the sum of the edge weights [124]. When the aim is to minimize, the edge weights are often referred to as edge costs.

Trees can be regarded as the most basic form of connected graphs, as they characterize the minimal properties required for connectivity. Due to their fundamental nature, they serve as effective tools for simplifying complex graphs while maintaining connectivity. This property finds applications in diverse fields such as network design and communication networks [8, 18, 188], clustering [65, 176], classification [41, 190], pathfinding algorithms [47, 164] and skeletonization [19, 46] among others.

In many applications, such as network design, it is often desirable for the crafted graph to be robust to perturbations, meaning that small changes in the data result in small changes

in the graph. The robustness of an algorithm can be measured under different criteria [62, 84, 90, 110, 131, 134]. In Chapter 5, we propose a spanning tree of a graph embedded in a Euclidean space, whose geometric structure is robust to perturbations such as noise on the coordinates of the nodes.

Occasionally, it may be beneficial to consider all spanning trees of a graph instead of focusing on a single one. If the graph is edge-weighted, we can assign a weight to each spanning tree, T , given by the product of its edge weights, $w(T) := \prod_{e \in E_T} w(e)$. This enables us to establish a probability distribution over all spanning trees, where each tree has a probability proportional to its weight, i.e., $\Pr(T) \propto w(T)$. It's important to note that if the graph is unweighted, the distribution is uniformly random since all trees are equally probable. Figure 1.3 illustrates a distribution over the spanning trees of a weighted graph.

In 1847, Kirchhoff introduced the Matrix Tree Theorem, which stated a formula to compute the total number of spanning trees of a graph G using the determinant of the Laplacian matrix of G . The Laplacian matrix of a graph is defined as follows:

$$L = D - A,$$

where A is the adjacency matrix with A_{ij} representing the weight value of edge (i, j) , and D is the diagonal degree matrix, with entries defined as $D_{ii} = \sum_j A_{ij}$. Thus, the Matrix Tree Theorem enables us to compute the normalization factor of the probability distribution of the trees. Formally, the theorem states:

Theorem 1.2.1 (Matrix Tree Theorem, [94, 168]). Given an edge-weighted undirected graph $G = (V, E, w)$, let \mathcal{T} represent the set of all spanning trees of G . Then, the total weight of these spanning trees, denoted as $w(\mathcal{T})$, can be expressed as:

$$w(\mathcal{T}) = \sum_{T \in \mathcal{T}} w(T) = \sum_{T \in \mathcal{T}} \prod_{e \in E_T} w(e) = \det(L^{[v]}),$$

where v is an arbitrary but fixed node, and $L^{[v]}$ represents the Laplacian matrix after removing the row and column indexed by v .

In other words, the Matrix Tree Theorem states that all the cofactors of the Laplacian matrix are equal and coincide with $w(\mathcal{T})$. Note that when G is unweighted, i.e., $w(e) = 1$ for all edges e , then the theorem states that $\det(L^{[v]})$ is equal to the total number of spanning trees of G .

This theorem is pivotal in the thesis because its generalization to directed graphs allows us to explore in Chapter 2 a distribution over a subset of directed spanning forests. We utilize this distribution to infer the labeling of a graph based on the probability of being connected to a labeled node in a forest. Additionally, in Chapter 3, we use it to compute the expected degree and variance of a node in a random spanning tree sampled from the probability distribution of spanning trees of a graph G .

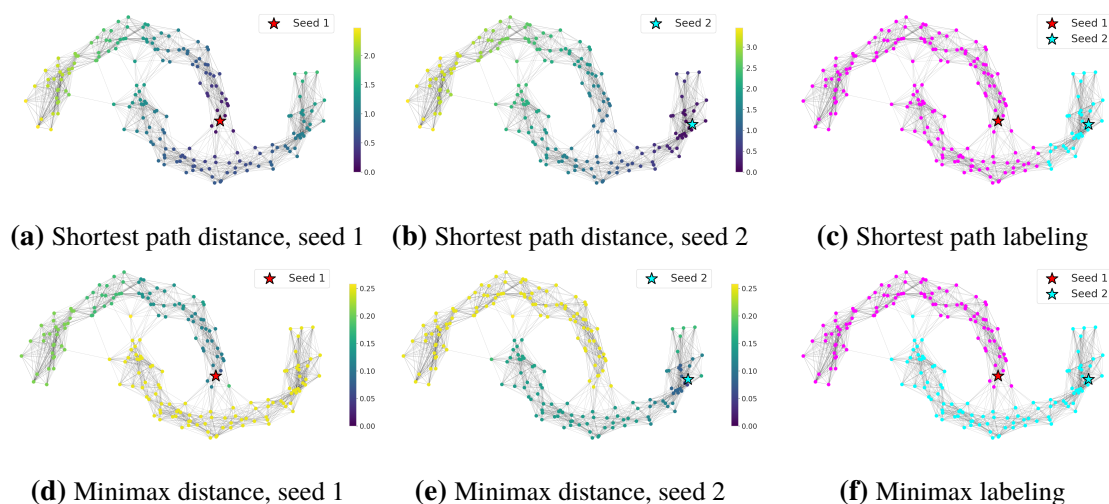


Figure 1.4. Impact of graph node distance in semi-supervised classification: Figures 1.4a) and 1.4b) illustrate the shortest path distance from all nodes to two seed nodes, while Figures 1.4d) and 1.4e) depict the minimax path distance. For a definition of these metrics, refer to Chapter 4. In Figures 1.4e) and 1.4f), the labeling is determined by the proximity of each seed node according to the corresponding distance metric, resulting in different labelings based on the graph node distances considered.

1.3 Graph Node Distances

Similarity/dissimilarity measures are crucial in machine learning, enabling the comparison of objects to discern commonalities and differences, thereby aiding in the classification and categorization of diverse entities. These measures quantify such similarities and differences based on specific criteria determined by the measures themselves. They find multiple uses, including visualization and dimensionality reduction techniques [37, 119, 126, 128], clustering [112, 136], classification [58, 75] and information retrieval [9, 51], among others. In the context of graphs, similarity/dissimilarity measures between vertices are relevant, allowing the generalization of pairwise relations encoded in the graph and aiding in the prediction of missing links, which is relevant for applications such as recommender systems [73, 161].

Different criteria exist to determine if two nodes are similar or dissimilar in a graph. These criteria can be based on various factors such as proximity (how close they are) [121], connectivity (how well they are connected by undirected paths) [97], substructure resemblance (sharing common substructures) [189], network role (similar influence or role in the network) [123], and more. Dissimilarity measures, which also function as distances, can be utilized for this purpose. A dissimilarity measure d is deemed a distance if it meets the following criteria:

$$1) d(u, v) = 0 \iff u = v \quad 2) d(u, v) = d(v, u) \quad 3) d(u, v) \leq d(u, w) + d(w, v)$$

for all triplets of nodes u, v, w in the graph. Node metrics in graphs are mostly based on connectivity and proximity criteria [34, 36, 60, 61].

Figure 1.4 illustrates how different node distances can impact classification in a semi-supervised setting. It compares the shortest path distance (the path with the minimum sum of edge costs) and the minimax distance (the path that minimizes the maximum edge cost), where classes are inferred by proximity to labeled nodes, referred to as seeds. In this specific example, the minimax distance proves to be more appropriate. The choice of metric can be crucial depending on the task, as it needs to adapt to the specific characteristics of the application. In Chapter 4, we study a series of node metrics solely based on the paths connecting two nodes and unify them under a common framework.

1.4 Thesis Outline

This section provides an overview of the thesis while highlighting the contributions of the thesis.

Chapter 2 proposes a graph-based semi-supervised learning algorithm, named Directed Probabilistic Watershed (DProbWS), designed for node classification on edge-weighted directed graphs. The method builds on the Probabilistic Watershed method [57] extending its capabilities to directed graphs. The algorithm establishes a Gibbs probability distribution over all spanning directed forests that separate the labeled nodes in different components. Utilizing this distribution, DProbWS infers the class for unlabeled nodes based on the probability of connection to a labeled node within a randomly sampled directed forest. We demonstrate that this probability can be analytically computed by solving a linear system. Furthermore, we establish its equivalence to the absorption probability of a random walker reaching one of the labeled nodes.

Chapter 3 also explores the application of a probability distribution, this time over the set of spanning trees within a given graph. Within this framework, we first explain the distinction between two types of degrees: weighted degree, which sums the weights of the edges incident to the node, and unweighted degree, which counts the number of neighbors of the node. Subsequently, we derive analytical formulas for both the expected degree and variance of a fixed node within a randomly sampled spanning tree.

Chapter 4 delves into the study of various metrics defined over the nodes of a graph from the perspective of the algebraic path problem. This problem is a generalization of the shortest path problem, replacing the min and sum operations with arbitrary operations that together define a semiring. We pose many popular path-based metrics as instances of the algebraic path problem while unifying them under a novel family of distances named the “log-norm distance”. Additionally, we investigate the inverse problem, determining under

which circumstances an instance of the algebraic path problem defines a proper metric over a graph, providing both sufficient and necessary conditions for such cases.

Chapter 5 introduces a family of robust spanning trees embedded in Euclidean space, named central spanning tree (CST), whose geometrical structure is resilient against perturbations such as noise on the coordinates of the nodes. Two variants of the problem are explored: one permitting the inclusion of Steiner points (referred to as branched central spanning tree or BCST), and another that does not. The family of trees is defined through a parameterized NP-hard minimization problem over the edge lengths, with specific instances including the minimum spanning tree or the Euclidean Steiner tree. The minimization problem weighs the length of the edges by their tree edge-centralities, which are regulated by a parameter α . The effect of α on the tree robustness is empirically analyzed, and a heuristic for approximating the optimal solution is proposed. Geometrical properties of the tree and its behavior in limit cases as $\alpha \rightarrow \pm\infty$ are also studied.

Chapter 6 applies the BCST to the skeletonization of 3D point clouds of plants. A pruning algorithm based on the Prize Collecting Steiner tree problem [87, 111] is defined to remove spurious branches generated by the BCST. The quality of the skeleton is evaluated qualitatively and quantitatively by comparison with other methods.

1.5 List of Publications

This thesis is based on the following publications/preprints:

- E. Fita Sanmartin, S. Damrich, and F. A. Hamprecht. “Directed Probabilistic Watershed”. *Advances in Neural Information Processing Systems*. 2021.
- E. Fita Sanmartin, S. Damrich, and F. A. Hamprecht. “The Algebraic Path Problem for Graph Metrics.” In *International Conference on Machine Learning*. 2022.
- E. Fita Sanmartin, C. Schnörr, and F. A. Hamprecht. “The Central Spanning Tree Problem.” 2023.

Complete list of publications to which the author contributed.

- J. Erik, E. Fita Sanmartin, and F. A. Hamprecht. “Extensions of Karger’s Algorithm: Why They Fail in Theory and How They Are Useful in Practice.” In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021.
- P. Lippmann, E. Fita Sanmartin, and F. A. Hamprecht. “Theory and Approximate Solvers for Branched Optimal Transport with Multiple Sources.” *Advances in Neural Information Processing Systems*. 2022.

Chapter 2

Directed Probabilistic Watershed

The Probabilistic Watershed is a semi-supervised learning algorithm applied on undirected graphs. Given a set of labeled nodes (seeds), it defines a Gibbs probability distribution over all possible spanning forests disconnecting the seeds. It calculates, for every node, the probability of sampling a forest connecting a certain seed with the considered node. We propose the "Directed Probabilistic Watershed", an extension of the Probabilistic Watershed algorithm to directed graphs. Building on the Probabilistic Watershed, we apply the Matrix Tree Theorem for directed graphs and define a Gibbs probability distribution over all incoming directed forests rooted at the seeds. Similar to the undirected case, this turns out to be equivalent to the Directed Random Walker. Furthermore, we show that in the limit case in which the Gibbs distribution has infinitely low temperature, the labeling of the Directed Probabilistic Watershed is equal to the one induced by the incoming directed forest of minimum cost. Finally, for illustration, we compare the empirical performance of the proposed method with other semi-supervised classification methods for directed graphs.

2.1 Introduction

Semi-supervised learning is the subfield of machine learning that utilizes both labeled and unlabeled data. It permits exploiting the large amount of unlabeled data available in many use cases jointly with frequently smaller sets of labeled data. When data is encoded as a network, graph-based semi-supervised learning aims to assign a label or class to each of the nodes of a network based on its topology and a given set of labeled nodes / seeds. Graph-based semi-supervised learning has been applied in multiple domains like computer vision [41, 66, 74], NLP [7, 107, 129], social networks [5, 184] and biomedical science[101]. Most graph-based semi-supervised learning algorithms focus on undirected weighted graphs [39, 158], though directed graphs appear naturally in many cases like in k-Nearest Neighbors graphs or citation and recommendation networks among others. Instances of methods

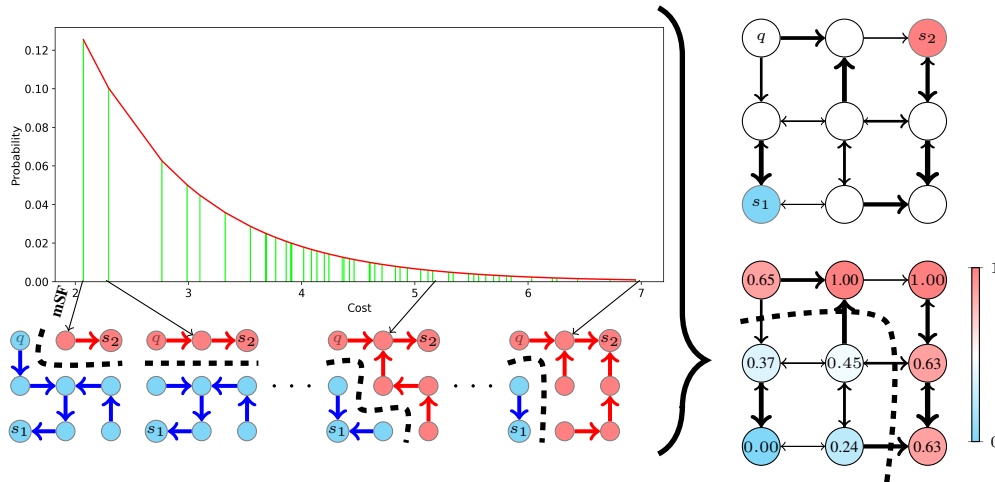


Figure 2.1. Overview DProbWS. The DProbWS computes the expected seed assignment of every node for a Gibbs distribution over all exponentially many spanning in-forests in closed-form. It thus avoids the winner-takes-all behaviour of the minimum cost spanning in-forest (mSF). **Top right:** Two-seeded directed graph with edge costs represented by the widths of the arrows. **Bottom left:** The mSF and other, higher cost in-forests. The mSF assigns the query node q to s_1 . Other in-forests of low cost might however induce different labelings. **Top left:** We therefore consider a Gibbs distribution over all spanning in-forests with respect to their cost (see equation (2.6)). Each green bar corresponds to the cost of one of the 60 possible spanning in-forests. **Bottom right:** DProbWS probabilities for assigning a node to s_2 . The dashed lines indicate the cut of the assignment. Query q is now assigned to s_2 . Considering a distribution over all spanning in-forests gives an uncertainty measure and can yield a different assignment from the mSF’s. In contrast to the 60 forests in this toy graph, for real-life graphs the number of in-forests increases exponentially with the number of edges and nodes.

proposed for directed graphs are [45, 54, 169, 193, 194].

Recently, [57] proposed the transductive graph-based semi-supervised learning algorithm Probabilistic Watershed (ProbWS). It envisages a Gibbs distribution over the labeled nodes separating forests (forests whose trees contain a unique labeled node) that span a weighted undirected graph. The probability of each forest is proportional to its weight. By means of the Matrix Tree Theorem [94, 168], the ProbWS computes the probability of sampling a seed separating forest such that a query node is connected to a single labeled node / seed. This approach turns out to be equivalent computationally and by result to the Random Walker / Harmonic energy minimization [15, 66, 192, 196].

We propose the “Directed Probabilistic Watershed” (DProbWS), an extension of ProbWS that can be applied to directed graphs through the extension of the Matrix Tree Theorem to directed graphs [102]. Instead of considering seed separating spanning forests, we deal

with incoming directed forests rooted at the seed nodes (see Figure 2.1). We analyze the DProbWS from a theoretical point of view and generalize the results presented in [57] to directed graphs. Building on the arguments used in the original ProbWS paper, our contributions include:

1. Analytically computing the probability that a graph node is assigned to a particular seed in an ensemble of Gibbs distributed incoming directed spanning forests rooted at the seeds (Section 2.3).
2. Demonstrating the equivalence of the Directed Probabilistic Watershed and the directed version of the Random Walker presented in [66] (Section 2.4).
3. Proposing a natural extension of the Power Watershed [41] to directed graphs (Section 2.5).
4. Conducting an illustrative experiment to showcase the empirical performance of the Directed Probabilistic Watershed and compare it with other semi-supervised transductive methods proposed in [45, 54, 194] (Section 2.6).

2.1.1 Related Work

Semi-supervised learning is an important subfield of machine learning due to its ability to exploit both labeled and unlabeled data [53]. If data is represented as a graph, graph based semi-supervised learning exploits, in addition to the labeled samples, the topology of the graph to infer the class of the unlabeled nodes. Undirected graphs have received more attention in this field than the directed ones [39, 158]. Nonetheless, there have been methods proposed for directed graphs. In [193, 194] the labels are inferred by minimizing a regularized function that forces highly connected subgraphs to have the same label. The paper [169] builds a symmetric pairwise similarity via co-linkage analysis from the directed graph in order to apply algorithms for undirected graphs. The work presented in [54] proposes a method based on game theory, where the game's Nash equilibrium defines the labeling of the data. Related to our work by the use of absorbing random walks is the one proposed in [45]. Their algorithm assigns the label of the seed that maximizes the accumulated expected number of visits from the unlabeled query node before being absorbed by an artificially added metanode. We, instead, prove that the DProbWS turns out to assign the label of the seed that maximizes the probability of absorption of a random walker.

Our method extends the Probabilistic Watershed (ProbWS) [57], which, in turn, is based on the Watershed algorithm [42]. The ProbWS paper established a link between the Watershed [42] and the Random Walker [15, 66, 192, 196]. Another such connection was made by the Power Watershed paper [41]. We extend these relations to the directed graph framework.

We smooth the combinatorial minimum spanning in-forest via entropic regularization.

More specifically, we consider a Gibbs distribution over all spanning in-forests, as was proposed in the original ProbWS paper [57]. Entropic regularization has been applied in other situations like the randomized shortest path framework [61, 95] or optimal transport [43]. If applied in conjunction with deep networks, it allows end-to-end training [130].

The Matrix Tree Theorem (MTT) is key to the theory developed in our work. The directed version of the MTT [102] allows us to compute in closed form the weight of a set of incoming forests. The MTT has been also applied in NLP [98], biology [166] and network analysis [165, 167]. A generalization of the MTT, the Matrix Forest Theorem (MFT), was used by [34] to define a distance between the nodes of a graph. Similar to our approach, [153] defines a Gibbs distribution over the forests using the MFT.

2.2 Background

2.2.1 Notation and Terminology

In this section, we introduce the main definitions and notation used in the paper. Most of them have been borrowed from [57, 102]. Let $G = (V, E, w, c)$ be a directed graph where V denotes the set of nodes, E the set of edges and w and c are functions that assign a weight $w(e) \in \mathbb{R}_{\geq 0}$ and a cost $c(e) \in \mathbb{R}$ to each edge $e \in E$. The edge $e = (i, j)$ indicates that there is a directed edge from i to j . If $(i, j) \notin E$, we set $c((i, j)) = \infty$ and $w((i, j)) = 0$. The set $S = \{s_1, s_2\} \subset V$ will denote the set of seed / labeled nodes and $U = V \setminus S$ the unlabeled ones. In order to ease the exposition we only consider the 2-seed scenario, although the method can be generalized to the n -seed scenario. We will use the terms seed and labeled node interchangeably.

Definition 2. Let $T = (V_T, E_T)$ be a graph. We say that T is an **incoming directed spanning tree rooted at $r \in V_T$** (in-tree for short) if:

1. every vertex $u \neq r$ has one and only one outgoing edge;
2. the root node r does not have any outgoing edges;
3. T does not have directed cycles, i.e., there does not exist a directed path in T such that the initial and final vertex are the same.

Note that condition 3), in conjunction with the other conditions, implies that no cycles (directed or undirected) will be formed. The literature refer to in-trees also as arborescences.

Equivalently, one can define an in-tree as a graph in which, for any vertex $u \in V_T$ there is exactly one directed path from u to the root r , and the root does not have any out-going edges.¹ Note that an in-tree becomes a tree in the classical sense if the directions of the edges

¹One can define analogously an outgoing tree (out-tree) rooted at r as a tree where there is exactly one directed path from the root r to any other node, and the root does not have any incoming edges.

are ignored. We say that an in-tree is spanning on $G = (V, E)$ if T is a subgraph of G and $V_T = V$. The set of spanning in-trees of G rooted at r will be denoted by $\mathcal{T}^{\vec{r}}$.

Analogously, the set $\mathcal{F}_{\vec{u}}$ represents the set of 2-in-trees spanning forests rooted at u and v , i.e., the spanning graphs of G consisting of two disjoint in-trees, such that u and v are the respective roots of the in-trees. Furthermore, if we consider a third node q , we define $\mathcal{F}_{\vec{u},q} \subseteq \mathcal{F}_{\vec{u}}$, as the set of all 2-in-trees spanning forests where q is connected to u by a directed path. In order to shorten the notation we will refer to 2-in-trees spanning forests simply as 2-in-forests or in-forests.

We define the weight of an arbitrary graph (e.g. in-forests) as the product of the weights of all its edges, $w(G) = \prod_{e \in E} w(e)$. The weight of a set of graphs, $w(\{G_i\}_{i=0}^n)$ is the sum of the weights of the graphs G_i . In a similar manner, we define the cost of a graph as the sum of the costs of all its edges, $c(G) = \sum_{e \in E} c(e)$. As in the Probabilistic Watershed [57], we consider $w(e) = \exp(-\mu c(e))$, $\mu \geq 0$, which will be a consequence of the definition of a Gibbs distribution over the 2-in-forests in $\mathcal{F}_{\vec{u}}$. Thus, a low edge-cost corresponds to a large edge-weight, and a minimum edge-cost spanning in-forest (mSF) is equivalent to a maximum edge-weight spanning in-forest (MSF). Depending on the context, the abbreviations mSF and MSF will be also used for the undirected versions of the minimum/maximum spanning forests. Via the Directed Matrix Tree Theorem [102], we aim to compute the weight of the set of the 2-in-forests rooted at the seeds. The theorem makes use of the out-Laplacian matrix which we define next.

Definition 3. Given a weighted directed graph $G = (V, E, w)$ we define the **out-Laplacian** of G as

$$L := D - A^T, \quad (2.1)$$

where $A \in \mathbb{R}^{|V| \times |V|}$ is the vertex-adjacency matrix of G represented entry-wise as $A_{ij} = w((i, j))$ and D denotes the diagonal matrix defined as $D_{ii} = \sum_{j \in V} A_{ij}$, i.e., D_{ii} is the out-degree of vertex i .² For brevity, we will refer to the out-Laplacian just as Laplacian. For any $v \in V$, $L^{[v]}$ will stand for the Laplacian after removing the row and column indexed by v .

2.2.2 Matrix Tree Theorem

Theorem 2.2.1 (MTTdir, [102]). For any edge-weighted directed graph G with an arbitrary fixed node $r \in V$ the weight of the set of incoming directed spanning trees rooted at r ,

²Analogously, the in-Laplacian is defined as $L_{in} = D_{in} - A$, where D_{in} is the diagonal matrix with the in-degrees of the nodes.

$w(\mathcal{T}^{\vec{r}})$, is equal to³

$$w(\mathcal{T}^{\vec{r}}) := \sum_{\mathbb{T} \in \mathcal{T}^{\vec{r}}} w(\mathbb{T}) = \sum_{\mathbb{T} \in \mathcal{T}^{\vec{r}}} \prod_{e \in E_{\mathbb{T}}} w(e) = \det(L^{[r]}).$$

Note that Theorem 2.2.1 generalizes the original Matrix Tree Theorem [94, 168] if we consider an undirected graph as a directed graph with both directions present for each edge. In this case, any spanning tree in the undirected graph can be interpreted as an in-tree rooted at an arbitrary root r once the direction of the edges has been set appropriately. Hence, for an undirected graph the choice of the root r is irrelevant and we retrieve the original MTT.

We propose to follow the approach taken in [57], but with the generalization of the Matrix Tree Theorem to directed graphs and thus obtain a directed version of the ProbWS.

2.2.3 Random Walker and Power Watershed

In this section, we summarize the Random Walker [66] and Power Watershed [41], two semi-supervised graph-based algorithms which are directly connected to the Probabilistic Watershed. Both algorithms consider an undirected graph, G , with a set of seeds $S = \{s_1, s_2\}$, i.e., labeled nodes.

The Random Walker paper [66] addresses the following problem: *What is the probability that a random walker starting at node q reaches seed s_1 before reaching seed s_2 ?* The solution to this question can be obtained by solving the combinatorial Dirichlet problem, which consists of finding the minimizer of

$$\frac{1}{2} x^\top L x = \frac{1}{2} \sum_{e=\{u,v\} \in E} w(e)(x_u - x_v)^2, \text{ s.t. } x_{s_1} = 1, x_{s_2} = 0, \quad (2.2)$$

where L denotes the undirected Laplacian of G . The minimizer of equation (2.2) is the solution of the following linear system

$$L_U x_U^{s_i} = -B_{s_i}^\top, \quad (2.3)$$

where $x_U^{s_i}$ represents the probability that the unlabeled nodes in U are absorbed by s_i , L_U is the square submatrix of L indexed by the elements in U and $B_{s_i}^\top$ is the row s_i of L without the seeds.

The Power Watershed generalizes the framework of the Random Walker and minimizes the following objective function

$$\sum_{e=\{u,v\} \in E} w^\alpha(e)(x_u - x_v)^\beta, \text{ s.t. } x_{s_1} = 1, x_{s_2} = 0, \quad (2.4)$$

³The weight of the out-trees can be calculated using the in-Laplacian instead of the out-Laplacian.

for $\alpha, \beta \geq 0$. For instance, $\alpha = 1$ and $\beta = 2$ define the Random Walker’s objective function. The Power Watershed analyzes the case $\alpha \rightarrow \infty$ and $\beta = 2$, which boils down to computing a mSF (minimum cost Spanning Forest) and applying the Random Walker in the plateaus (connected subgraphs with constant edge weight), resolving the ambiguity of which mSF is sampled when there is more than one.

The generalization of these algorithms to directed graphs is not straightforward if one takes the objective function approach. Solving (2.2) directly is equivalent to solve the problem with an undirected graph whose edge-weights are equal to the sum of the edge-weights in both directions of the original directed graph. The method proposed in [156] generalizes the combinatorial Dirichlet problem (2.2) by interpreting the graph as an electrical network with diodes. This approach is not equivalent to the random walk on a directed graph anymore.

The Random Walker can also be generalized to the directed case if we use the intuitive approach instead, i.e., we consider the probability of a random walker (constrained to follow the directions of the edges) being absorbed by a certain seed. In AppendixA.1, we give a proof of how these probabilities can be computed in the directed case. We demonstrate that the solution of the linear system (2.3) with the Laplacian transposed provides the desired probabilities. In section 2.5, we show how the Power Watershed can be generalized to directed graphs by means of the DProbWS.

2.2.4 Probabilistic Watershed Review

The Probabilistic Watershed (ProbWS) [57] is based on the Watershed algorithm. The Watershed algorithm calculates a minimum cost spanning forest, mSF, such that the seeds belong to different components [42]. A query node inherits the label of the seed to which it is connected in the mSF. The ProbWS instead, considers all possible seed-separating spanning forests. It defines a Gibbs probability distribution over the forests and analytically computes, for every node, the probability of sampling a forest connecting a certain seed with that node via the application of the Matrix Tree Theorem (for undirected graphs).

In [57] it was shown that the ProbWS turns out to be equivalent to the Random Walker algorithm proposed in [66]. Additionally, it was shown that the Power Watershed potentials [41] were given by the ProbWS probabilities when the latter is restricted to mSFs instead of all spanning forests. This restriction manifests when the entropy of the Gibbs distribution of forests is minimized.

In this work, we extend the ProbWS framework to directed graphs and call it Directed Probabilistic Wasterhsed. In this case, the spanning forests considered are formed by in-forests rooted at the seeds, i.e., directed trees with a unique path from every node to the

seed. Inspired by [57], we define a Gibbs Probability distribution over the in-forests and we apply the Matrix Tree Theorem for directed graphs to compute the probability of sampling a forest connecting a certain seed with a query node in closed-form. Next, we demonstrate the equivalence to the Directed Random Walker and finally we propose an extension of the Power Watershed for directed graphs.

2.3 Directed Probabilistic Watershed

2.3.1 Gibbs Probability Distribution

Similarly to [57], we define a Gibbs probability distribution over 2-in-forests with given entropy J , such that the 2-in-forests with a lower cost have a higher probability mass. Formally, the 2-in-forests are sampled from the distribution which minimizes

$$\min_P \sum_{\mathbf{F} \in \mathcal{F}_{s_1}^{\vec{s}_2}} P(\mathbf{F})c(\mathbf{F}), \quad \text{s.t.} \quad \sum_{\mathbf{F} \in \mathcal{F}_{s_1}^{\vec{s}_2}} P(\mathbf{F}) = 1 \quad \text{and} \quad \mathcal{H}(P) = J, \quad (2.5)$$

where $\mathcal{H}(P)$ is the entropy of P . The lower the entropy, the more probability mass is given to the 2-in-forests of lowest cost. The minimizing distribution is the Gibbs distribution ([see 174, 3.2]):

$$P(\mathbf{F}) = \frac{\exp(-\mu c(\mathbf{F}))}{\sum_{\mathbf{F}' \in \mathcal{F}_{s_1}^{\vec{s}_2}} \exp(-\mu c(\mathbf{F}'))} = \frac{\prod_{e \in E_{\mathbf{F}}} \exp(-\mu c(e))}{\sum_{\mathbf{F}' \in \mathcal{F}_{s_1}^{\vec{s}_2}} \prod_{e' \in E_{\mathbf{F}'}} \exp(-\mu c(e'))} = \frac{w(\mathbf{F})}{\sum_{\mathbf{F}' \in \mathcal{F}_{s_1}^{\vec{s}_2}} w(\mathbf{F}')}, \quad (2.6)$$

where μ can be interpreted as the inverse temperature of the Gibbs distribution. The parameter μ implicitly determines the entropy. A higher μ (lower temperature) implies a lower entropy. Equation (2.6) motivates our choice of edge weights $w(e) = \exp(-\mu c(e))$.

Definition 4 (Probabilities of the Directed Probabilistic Watershed). Given two seeds s_1 and s_2 and a query node q , we define the Directed Probabilistic Watershed's probability that q and s_1 have the same label as the probability of sampling a 2-in-forest from $\mathcal{F}_{s_1}^{\vec{s}_2}$ that connects q to s_1 by a directed path, i.e., they belong to the same in-tree rooted at s_1 :

$$P(q \sim s_1) := \sum_{\mathbf{F} \in \mathcal{F}_{s_1, q}^{\vec{s}_2}} P(\mathbf{F}) = \frac{\sum_{\mathbf{F} \in \mathcal{F}_{s_1, q}^{\vec{s}_2}} w(\mathbf{F})}{\sum_{\mathbf{F}' \in \mathcal{F}_{s_1}^{\vec{s}_2}} w(\mathbf{F}')} = \frac{w\left(\mathcal{F}_{s_1, q}^{\vec{s}_2}\right)}{w\left(\mathcal{F}_{s_1}^{\vec{s}_2}\right)}. \quad (2.7)$$

The Directed Probabilistic Watershed (DProbWS) takes all spanning 2-in-forests into account according to their cost (see Figure2.1). The resulting assignment probability of each node provides an uncertainty measure. Assigning each node to the seed for which

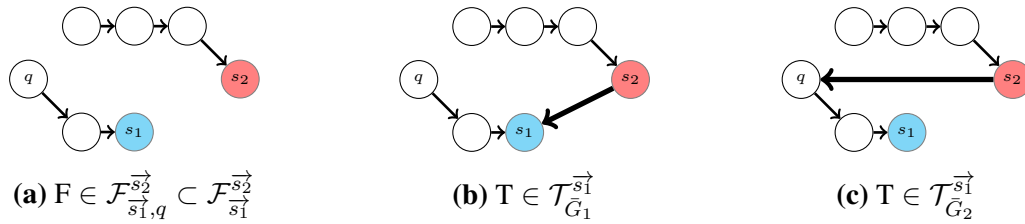


Figure 2.2. Procedure to Transform an In-Forest in $\mathcal{F}_{\vec{u}}^{\vec{v}}[s_1][s_2][q]$ to a Spanning In-Tree in \bar{G}_1 and \bar{G}_2 as Defined in Lemma 2.3.3. 2.2a) In-forest $F \in \mathcal{F}_{s_1, q}^{\vec{s}_2} \subseteq \mathcal{F}_{s_1}^{\vec{s}_2}$. 2.2b) Transformation of F into an in-tree $T \in \mathcal{T}_{\bar{G}_1}^{\vec{s}_1}$ after adding the edge (s_2, s_1) (thick edge). 2.2c) Transformation of F into an in-tree $T \in \mathcal{T}_{\bar{G}_2}^{\vec{s}_1}$ after adding the edge (s_2, q) (thick edge).

it has the highest probability can yield a labeling different from the in-forest with highest individual probability, the mSF.

Remark 2.3.1. Note that the probability given by the DProbWS is well defined if and only if $w(\mathcal{F}_{s_1}^{\vec{s}_2}) \neq 0$, that is if and only if there exists at least one in-forest rooted at the seeds. As far as any node is connected by a directed path to at least one of the seeds, such an in-forest will exist. If the seeds were not reachable from a node q , this node would have no connection with the seeds and therefore we could not infer any label. These nodes are termed zero-knowledge nodes. We can still use the proposed method by removing all nodes in the zero-knowledge components [120]. In the remaining we will assume $w(\mathcal{F}_{s_1}^{\vec{s}_2}) \neq 0$.

Remark 2.3.2. Note that the DProbWS probabilities have been based on the concept of incoming forests. The same reasoning can be extended for outgoing forests just by reversing the edge directions.

2.3.2 Computation of the Directed Probabilistic Watershed Probabilities

In this section, we explain how to compute the probabilities defined by the DProbWS. While for undirected graphs it is possible to compute $w(\mathcal{F}_{s_1}^{s_2})$ as a difference of the weight of the set of spanning trees of the graph G and the weight of the set of spanning trees of G after adding an edge connecting the seeds, this is not possible for $w(\mathcal{F}_{s_1, q}^{s_2})$.⁴ In [57], $w(\mathcal{F}_{s_1, q}^{s_2})$ is obtained by solving a linear system involving $w(\mathcal{F}_{s_1}^{s_2})$, $w(\mathcal{F}_{s_2}^q)$ and $w(\mathcal{F}_{s_1}^q)$. In contrast to the undirected case, we can bypass the auxiliary linear system due to the directions of the edges which restrict the number of possible directed forests. This permits us to express $w(\mathcal{F}_{s_1, q}^{\vec{s}_2})$ and $w(\mathcal{F}_{s_1, q}^{\vec{q}})$ in terms of the weight of the set of in-trees of an augmented graph as it is shown in the following lemma.

⁴ $\mathcal{F}_{s_1, q}^{s_2}$, $\mathcal{F}_{s_1}^{s_2}$, $\mathcal{F}_{s_2}^q$ and $\mathcal{F}_{s_1}^q$ are equivalents to $\mathcal{F}_{s_1, q}^{\vec{s}_2}$, $\mathcal{F}_{s_1}^{\vec{s}_2}$, $\mathcal{F}_{s_2}^{\vec{q}}$ and $\mathcal{F}_{s_1}^{\vec{q}}$ but defined in an undirected graph.

Lemma 2.3.3. Let $G = (V, E, w)$ be a directed graph. Given $s_1, s_2, q \in V$, define the graphs $\bar{G}_1 = (V, E \cup (s_2, s_1), \bar{w}_1)$ and $\bar{G}_2 = (V, E \cup (s_2, q), \bar{w}_2)$ where $\bar{w}_1(e) = \bar{w}_2(e) = w(e) \forall e \in E$ and $\bar{w}_1((s_2, s_1)) = \bar{w}_2((s_2, q)) = 1$. Then

$$\mathbf{a)} \ w \left(\mathcal{F}_{s_1}^{\vec{s}_2} \right) = w \left(\mathcal{T}_{\bar{G}_1}^{\vec{s}_1} \right) - w \left(\mathcal{T}_G^{\vec{s}_1} \right) \quad \mathbf{b)} \ w \left(\mathcal{F}_{s_1, q}^{\vec{s}_2} \right) = w \left(\mathcal{T}_{\bar{G}_2}^{\vec{s}_1} \right) - w \left(\mathcal{T}_G^{\vec{s}_1} \right).$$

Proof: We will only prove **a)**. Note that $w \left(\mathcal{T}_{\bar{G}_1}^{\vec{s}_1} \right) - w \left(\mathcal{T}_G^{\vec{s}_1} \right)$ is the weight of all the spanning trees of \bar{G}_1 rooted at s_1 containing the edge (s_2, s_1) . It is easy to see that for any $F \in \mathcal{F}_{s_1}^{\vec{s}_2}$, there exists a unique $T \in \mathcal{T}_{\bar{G}_1}^{\vec{s}_1}$ containing (s_2, s_1) such that $w(T) = w(F)$. Indeed, if we add the edge (s_2, s_1) to F we obtain a tree $T \in \mathcal{T}_{\bar{G}_1}^{\vec{s}_1}$ since for every node in V , there exists a unique path to s_1 (see Figure 2.2b). Conversely, any tree in $T \in \mathcal{T}_{\bar{G}_1}^{\vec{s}_1}$ containing the edge (s_2, s_1) will be transformed into a forest $F \in \mathcal{F}_{s_1}^{\vec{s}_2}$ after the removal of the edge of (s_2, s_1) . Due to $\bar{w}_1((s_2, s_1)) = 1$ the equality of weights $w(F) = w(T)$ follows. The argument for **b)** is analogous to the previous case (see Figure 2.2c). \square

The following result (Lemma 2.3.5) provides a closed formula for $w \left(\mathcal{F}_{s_1}^{\vec{s}_2} \right)$ and $w \left(\mathcal{F}_{s_1, q}^{\vec{s}_2} \right)$ in terms of the in-trees of G rooted at the seeds. The proof makes use of the MTTdir (Theorem 2.2.1), Lemma 2.3.3 and the well-known Determinant Lemma [72] which we state without proof.

Lemma 2.3.4 (Determinant Lemma [72]). Given an invertible matrix $A \in \mathbb{R}^{m \times m}$ and $u, v \in \mathbb{R}^m$ then we have:

$$\det(A + uv^\top) = \det(A)(1 + v^\top A^{-1}u)$$

Lemma 2.3.5. Let $G = (V, E, w)$ be a directed graph. Let $l_{ij}^{-1, [v]}$ represent the entry ij of the matrix $(L^{[v]})^{-1}$ for any $v \in V$. Given s_1, s_2 and q :

$$\mathbf{a)} \ w \left(\mathcal{F}_{s_1}^{\vec{s}_2} \right) = w(\mathcal{T}_{s_2 s_2}^{\vec{s}_1}) l_{s_2 s_2}^{-1, [s_1]} = w(\mathcal{T}_{s_1 s_1}^{\vec{s}_2}) l_{s_1 s_1}^{-1, [s_2]}, \quad \mathbf{b)} \ w \left(\mathcal{F}_{s_1, q}^{\vec{s}_2} \right) = w(\mathcal{T}_{s_2 s_2}^{\vec{s}_1}) (l_{s_2 s_2}^{-1, [s_1]} - l_{s_2 q}^{-1, [s_1]}).$$

Proof: Let $\mathbf{1}_j$ denote the column j of the identity matrix I . Given a column vector $v \in \mathbb{R}^{|V|}$, $v^{[s_1]}$ denotes the vector v after removing the entry indexed by s_1 .

1. Let $\bar{G}_1 = (V, E \cup (s_2, s_1), \bar{w})$ be defined as in Lemma 2.3.3. We can compute

$$\begin{aligned} w \left(\mathcal{F}_{s_1}^{\vec{s}_2} \right) &\stackrel{\text{Lemma 2.3.3}}{=} w \left(\mathcal{T}_{\bar{G}_1}^{\vec{s}_1} \right) - w \left(\mathcal{T}_G^{\vec{s}_1} \right) \stackrel{\text{MTTdir}}{=} \det \left(L_{\bar{G}_1}^{[s_1]} \right) - \det \left(L_G^{[s_1]} \right) \\ &= \det \left(L_G^{[s_1]} + \mathbf{1}_{s_2}^{[s_1]} \left(\mathbf{1}_{s_2}^{[s_1]} \right)^\top \right) - \det \left(L_G^{[s_1]} \right) \\ &\stackrel{\text{Lemma 2.3.4}}{=} \det \left(L_G^{[s_1]} \right) \left(1 + \left(\mathbf{1}_{s_2}^{[s_1]} \right)^\top \left(L_G^{[s_1]} \right)^{-1} \mathbf{1}_{s_2}^{[s_1]} \right) - \det \left(L_G^{[s_1]} \right) \\ &= \det \left(L_G^{[s_1]} \right) \ell_{s_2 s_2}^{-1, [s_1]} \stackrel{\text{MTTdir}}{=} w \left(\mathcal{T}_G^{\vec{s}_1} \right) \ell_{s_2 s_2}^{-1, [s_1]}. \end{aligned} \tag{2.8}$$

To prove $w \left(\mathcal{F}_{s_1}^{\vec{s}_2} \right) = w(\mathcal{T}_{s_1 s_1}^{\vec{s}_2}) l_{s_1 s_1}^{-1, [s_2]}$ we just need to exchange the role of s_1 and s_2 in (2.8) and follow the same steps as above.

2. Let $b = \mathbf{1}_{s_2} - \mathbf{1}_q$ and \bar{G}_2 be defined as in Lemma 2.3.3. Then

$$\begin{aligned}
w\left(\mathcal{F}_{\bar{s}_1, q}^{\bar{s}_2}\right) &\stackrel{\text{Lemma 2.3.3}}{=} w\left(\mathcal{T}_{\bar{G}_2}^{\bar{s}_1}\right) - w\left(\mathcal{T}_G^{\bar{s}_1}\right) \stackrel{\text{MTTdir}}{=} \det\left(L_{\bar{G}_2}^{[s_1]}\right) - \det\left(L_G^{[s_1]}\right) \\
&= \det\left(L_G^{[s_1]} + b^{[s_1]} \left(\mathbf{1}_{s_2}^{[s_1]}\right)^\top\right) - \det\left(L_G^{[s_1]}\right) \\
&\stackrel{\text{Lemma 2.3.4}}{=} \det\left(L_G^{[s_1]}\right) \left(1 + \left(\mathbf{1}_{s_2}^{[s_1]}\right)^\top \left(L_G^{[s_1]}\right)^{-1} b^{[s_1]}\right) - \det\left(L_G^{[s_1]}\right) \\
&= \det\left(L_G^{[s_1]}\right) \left(l_{s_2 s_2}^{-1, [s_1]} - l_{s_2 q}^{-1, [s_1]}\right) \stackrel{\text{MTTdir}}{=} w\left(\mathcal{T}^{\bar{s}_1}\right) \left(l_{s_2 s_2}^{-1, [s_1]} - l_{s_2 q}^{-1, [s_1]}\right).
\end{aligned} \tag{2.9}$$

□

As a consequence of Lemma 2.3.5, we can conveniently find a closed form for the probabilities of the DProbWS (Definition 4):

Theorem 2.3.6. The Directed Probabilistic Watershed probabilities are equal to

$$\Pr(q \sim s_1) = \frac{l_{s_2 s_2}^{-1, [s_1]} - l_{s_2 q}^{-1, [s_1]}}{l_{s_2 s_2}^{-1, [s_1]}} \quad \text{and} \quad \Pr(q \sim s_2) = \frac{l_{s_2 q}^{-1, [s_1]}}{l_{s_2 s_2}^{-1, [s_1]}},$$

or equivalently

$$\Pr(q \sim s_2) = \frac{l_{s_1 s_1}^{-1, [s_2]} - l_{s_1 q}^{-1, [s_2]}}{l_{s_1 s_1}^{-1, [s_2]}} \quad \text{and} \quad \Pr(q \sim s_1) = \frac{l_{s_1 q}^{-1, [s_2]}}{l_{s_1 s_1}^{-1, [s_2]}}.$$

Our discussion was constrained to the case of two seeds only to ease our explanation. We can reduce the case of multiple seeds per label to the two seed case by merging all nodes seeded with the same label. Similarly, the case of more than two labels can be reduced to the two label scenario by using a ‘‘one versus all strategy’’: We choose one label and merge the seeds of other labels into one unique seed. In both cases we might add multiple edges between node pairs. While having formulated our arguments for simple graphs, they are also valid for multigraphs by the same arguments as in [57].

2.4 Equivalence of DProbWS and the Directed Random Walker

Resembling the ProbWS equivalence with the Random Walker, the DProbWS also turns out to be equivalent to the directed version of the Random Walker. Although multiple references exist on how to compute the absorbing probabilities of a random walker in terms of the Laplacian matrix for undirected graphs [60, 66], we could not find any suitable reference for the directed case.⁵ We sketch how the absorbing probabilities can be obtained for the

⁵However, there exist closed formulas to compute the absorbing probabilities of Markov chains in terms of the fundamental matrix [91].

directed case in terms of the Laplacian matrix and refer to the supplemental material for a complete derivation.

Theorem 2.4.1. The probability, $x_q^{s_1}$, that a random walker on a directed graph starting at node q first reaches s_1 before reaching s_2 is given by the solution of the following linear system

$$L_U^\top x_U^{s_1} = - [B_1^\top]_{s_1}, \quad (2.10)$$

where $x_U^{s_i}$ represents the probability that the nodes in U are absorbed by s_i , L_U is the square submatrix of L indexed by the elements in U and $[B_1^\top]_{s_1}$ is the column s_1 of L without the entries indexed by the seeds.

Proof sketch: Let P denote the transition probability matrix defined by the absorbing random walker. We note that the probability of being absorbed by s_1 at q can be obtained by computing the entry (q, s) of P^n when n tends to infinity, i.e., $x_q^{s_1} = \lim_{n \rightarrow \infty} [P^n]_{qs}$. By expressing the transition matrix in terms of the Laplacian, we find a closed formula of the matrix P^n by induction. By taking the limit to infinity, we obtain the desired result. See Appendix A.1 for further details. \square

Remark 2.4.2. Note that (2.10) is the transposed version of the linear system solved by the undirected Random Walker and equivalently the ProbWS (2.3), i.e., $(L_U) y_U = -B_1^\top$. Since [57, 66] consider an undirected graph $(L_U^\top) = L_U$, the transposition becomes irrelevant in the undirected case.

The next theorem states the equivalence between the DProbWS and the Directed Random Walker. The proof is analogous to Theorem 4.1 in [57].

Theorem 2.4.3. The probability, $x_q^{s_1}$, that a random walker on a directed graph starting at node q first reaches s_1 before reaching s_2 is equal to the Directed Probabilistic Watershed's probability (Definition 4)

$$x_q^{s_1} = P(q \sim s_1).$$

Proof: We write the probability of the DProbWS in terms of the inverse of $L^{[s_2]}$ (Theorem 2.3.6):

$$\Pr(q \sim s_1) = \ell_{s_1 q}^{-1, [s_2]} / \ell_{s_1 s_1}^{-1, [s_2]}. \quad (2.11)$$

Therefore, we only need to know the row s_1 of $(L^{[s_2]})^{-1}$ to calculate for each q the probability $P(q \sim s_1)$, which can be computed solving the following linear system

$$(L^{[s_2]})^\top y = \mathbf{1}_{s_1} / \ell_{s_1 s_1}^{-1, [s_2]} \iff y = \left((L^{[s_2]})^\top \right)^{-1} \mathbf{1}_{s_1} / \ell_{s_1 s_1}^{-1, [s_2]} = \left((L^{[s_2]})^\top \right)^{-1}_{\cdot, s_1} / \ell_{s_1 s_1}^{-1, [s_2]}. \quad (2.12)$$

Here $\mathbf{1}_{s_1}$ denotes the column s_1 of the identity matrix. Note that y is the vector formed by the elements in the right hand side of (2.11). Let us assume without loss of generality that

the row corresponding to the seed s_1 is the first one. Thus, we can express equation (2.12) block-wise :

$$\begin{pmatrix} L_{s_1 s_1} & [B_2^\top]_{s_1} \\ [B_1^\top]_{s_1} & L_U^\top \end{pmatrix} \begin{pmatrix} y_{s_1} \\ y_U \end{pmatrix} = \begin{pmatrix} L_{s_1 s_1} y_{s_1} + [B_2^\top]_{s_1} y_U \\ [B_1^\top]_{s_1} y_{s_1} + L_U^\top y_U \end{pmatrix} = \begin{pmatrix} 1/\ell_{s_1 s_1}^{-1, [s_2]} \\ 0 \end{pmatrix}, \quad (2.13)$$

where $L_{s_1 s_1}$ is the entry $s_1 s_1$ of the Laplacian, $[B_1]_{s_1}$ and $[B_2]_{s_1}$ are the row and column s_1 of the Laplacian without considering the element in the diagonal and L_U are the rows and columns of the unseeded vertices. Since $y_{s_1} = \Pr(q \sim s_1) = 1$, we obtain the following linear system of equations

$$L_U^\top y_U = - [B_1^\top]_{s_1}, \quad (2.14)$$

which is the same linear system that the Directed Random Walker solves (see Theorem 2.4.1 and Appendix A.1 for more details). Therefore $P(q \sim s_1) = y_q = x_q^{s_1}$ for all q . \square

The equivalence stated by Theorem 2.4.3 combined with Theorem 2.4.1 implies that in order to obtain the DProbWS probabilities defined in Theorem 2.3.6, it is just necessary to solve a linear system for each seed node (see Algorithm 1), as it is the case for the Random Walker algorithm.

Algorithm 1: Directed Probabilistic Watershed / Directed Random Walker

Input: $G = (V, E, w)$, seeds

Output: x // DProbWS probabilities

1 $U = V \setminus \text{seeds}$ // Unlabeled nodes

2 $L = \text{outLaplacian}(G)$

3 $L_U = L[U; U]$ // submatrix of L indexed by the unlabeled nodes

4 **for** $s \in \text{seeds}$ **do**

5 $B_s = L[U; s]$ // s column of L restricted to unlabeled nodes

6 $x[U; s] = \text{solve}(L_U^\top, -B_s)$ // Solves the linear system

$L_U^\top * x_U^s = -B_s$

2.4.1 Teleporting Random Walker

As pointed out in Remark 2.3.1, if a node cannot reach any seed via a directed path, the DProbWS method is not able to infer the label. Such nodes are known as zero-knowledge nodes. From the tree perspective view, the zero knowledge nodes do not belong to any in-tree rooted at the seeds and therefore the DProbWS probabilities are all equal to zero.

Inspired by [194] and leveraging the equivalence of the DProbWS with the Random Walker method, we remedy this by replacing the natural random walker by the so-called

teleporting random walker (TRW). In the TRW setting, a random walker jumps uniformly at random to any node with probability η , and with probability $(1 - \eta)$ takes a step of the natural random walker. This ensures that any node can reach any other node. Let P and P_{TRW} be the transition probability matrices of the random walker and the teleporting random walker. Formally, they are related as follows

$$P_{TRW} = (1 - \eta)P + \frac{\eta}{n - 1} (\mathbb{1}\mathbb{1}^\top - I), \quad (2.15)$$

where $\mathbb{1}$ is the column vector full of ones, I is the identity matrix of appropriate size and n is the number of nodes in the graph. Note that the Laplacian matrix L is related to the transition probability matrix P in the following way

$$L^\top = D - A = D(I - D^{-1}A) = D(I - P).$$

Thus, the linear system of equation (2.10) can be expressed in terms of the transition probability matrix P as

$$(I - P_U)x_U^{s_1} = -(D_U)^{-1} [B_1^\top]_{s_1} \quad (2.16)$$

Therefore, we can solve the linear system that determines the DProbWS probabilities in terms of P . Moreover, we can apply the TRW approach to the DProbWS by solving the linear system with respect to P_{TRW} . We refer to the variant that uses the TRW by DProbWStrw.

In practice, the TRW approach is equivalent to add to every node out-going edges towards the rest of nodes. By adding these out-going edges to each node the graph becomes a complete graph. Hence, a priori, one cannot exploit the sparsity of the Laplacian when solving the DProbWS linear systems. However, in Appendix A.2, we present an efficient computation of the DProbWStrw probabilities, allowing the use of sparse linear solvers at the cost of solving an additional linear system. Specifically, we demonstrate the following:

Theorem 2.4.4. Given a weighted graph G with n nodes, where s is a seed node and U is the set of unlabeled nodes, let x_U^s represent the DProbWS/absorption probabilities associated with the transition probability matrix P of G . Similarly, let \tilde{x}_U^s represent the DProbWStrw/absorption probabilities associated with the TRW transition probability matrix P_{TRW} with parameter η as defined in (2.15). Then, both probability vectors are related as follows:

$$\tilde{x}_U^s = x_U^s - \frac{\eta(\eta + 1)}{1 + \eta\mathbb{1}^\top y} y \left(\left(1 + \frac{\eta}{(n - 1)(1 - \eta)} \right) + \mathbb{1}^\top x_U^s - \frac{\eta(\eta + 1)}{1 + \eta\mathbb{1}^\top y} \mathbb{1}^\top y \right), \quad (2.17)$$

where $\mathbb{1}$ is a vector filled with ones and $y := (I - P_U)^{-1}\mathbb{1}$.

Hence, Theorem 2.4.4 states that the DProbWStrw probabilities given by \tilde{x}_U^s can easily be computed via (2.17), if we compute x_U^s and $y = (I - P_U)^{-1}\mathbb{1}$. Consequently, the computation of the DProbWStrw probabilities require to solve one additional sparse linear

system in comparison to the number of linear systems needed to compute the DProbWS probabilities, which costs much less effort than solving a dense linear system per seed.

The proof of Theorem 2.4.4 is deferred to Appendix A.2. This proof consists of two parts. First, we demonstrate that the absorption probabilities of TRW with self-loops are equivalent to those of TRW without self-loops. This equivalence must be established because the DProbWS Laplacian matrix remains unchanged when self-loops are added, although the transition probability matrix does change.

The second part involves leveraging the equivalence that permits the addition of self-loops, allowing us to express P_{TRW} as a 1-rank perturbation of P . This is because we do not need to subtract the identity in the second term of equation (2.15) to remove self-loops. Utilizing the Sherman-Morrison formula (see Lemma A.2.2), which provides an efficient way to compute the inverse of a matrix perturbed by the addition of a 1-rank matrix, we establish a relationship between the inverse matrices of P_{TRW} and P .

2.5 Directed Power Watershed

In the ProbWS paper [57], it was proven that the Power Watershed [41] is equivalent to applying the ProbWS restricted to the minimum cost spanning forests. This restriction corresponds to the case of a Gibbs distribution of minimal entropy over the forests. In this section, we will prove the analogous result for the DProbWS: When the entropy of the Gibbs distribution over the directed in-forests (2.3.1) is minimal, then DProbWS is restricted to the minimum cost spanning in-forests (mSF). This permits us to define a natural extension of the Power Watershed to directed graphs.

In Section 2.3.1, we defined the weight of an edge e as $w(e) = \exp(-\mu c(e))$, where $c(e)$ was the edge-cost and μ implicitly determined the entropy of the 2-in-forest distribution. When $\mu \rightarrow \infty$, the distribution will have minimal entropy. Consequently, only for minimum cost spanning in-forests, mSFs (or analogously the maximum weight spanning in-forests, MSFs) the limit does not yield a zero probability, so that only they will be considered in the sampling.

Theorem 2.5.1. Given two seeds s_1 and s_2 . Let further $w_{\max} := \max_{\mathbf{F} \in \mathcal{F}_{s_1}^{\vec{s}_2}} w(\mathbf{F})$. If the entropy of the Gibbs distribution over the in-forests is minimized (2.6), then

$$x_q^{s_1} = \frac{\left| \{ \mathbf{F} \in \mathcal{F}_{s_1, q}^{\vec{s}_2} : w(\mathbf{F}) = w_{\max} \} \right|}{\left| \{ \mathbf{F} \in \mathcal{F}_{s_1}^{\vec{s}_2} : w(\mathbf{F}) = w_{\max} \} \right|}.$$

Proof: The proof is analogous to the one presented in [57]. The entropy of the Gibbs distribution (2.6) is minimized when $\mu \rightarrow \infty$. For a fixed $\mu_0 > 0$, let us define $\mu = \alpha \cdot \mu_0$,

then

$$w(e) = \exp(-\mu c(e)) = \exp(-\alpha \cdot \mu_0 c(e)) = w_0(e)^\alpha,$$

where $w_0(e) = \exp(-\mu_0 c(e))$. Since $\mu \rightarrow \infty \iff \alpha \rightarrow \infty$, we have

$$P_\alpha(q \sim s_1) := \frac{\sum_{F \in \mathcal{F}_{s_1, q}^{\vec{s}_2}} \prod_{e \in F} w_0(e)^\alpha}{\sum_{F \in \mathcal{F}_{s_1}^{\vec{s}_2}} \prod_{e \in F} w_0(e)^\alpha} = \frac{\sum_{F \in \mathcal{F}_{s_1, q}^{\vec{s}_2}} w_0(F)^\alpha}{\sum_{F \in \mathcal{F}_{s_1}^{\vec{s}_2}} w_0(F)^\alpha} = \frac{\sum_{F \in \mathcal{F}_{s_1, q}^{\vec{s}_2}} \left(\frac{w_0(F)}{w_{\max}} \right)^\alpha}{\sum_{F \in \mathcal{F}_{s_1}^{\vec{s}_2}} \left(\frac{w_0(F)}{w_{\max}} \right)^\alpha} \stackrel{\alpha \rightarrow \infty}{(\star)} x_q^{s_1} \quad (2.18)$$

In (\star) we used the fact that $\frac{w(F)}{w_{\max}} < 1 \iff w(F) \neq w_{\max}$. When $\alpha \rightarrow \infty$, only for the MSFs the fraction $\left(\frac{w(F)}{w_{\max}} \right)^\alpha$ does not tend to 0, but to 1. Thus, we are counting MSFs. \square

2.6 Experiments

While our research has stemmed from a theoretical standpoint, we conducted a practical experiment to illustrate the performance of DProbWS in node classification.⁶ Utilizing a graph with labeled nodes, we aimed to infer the labels of the remaining nodes. We used datasets from UCI [50] including *Digits* [183] and *20Newsgroups* [100], although these datasets are not inherently graph-based. To address this, we employed a common practice of inferring a graph from data embedded in a metric space, using k-Nearest Neighbors (kNN) graphs. Given their asymmetric nature, kNN graphs can be treated as directed graphs. We constructed kNN graphs with a parameter value of $k = 5$.

Additionally we consider the *Email-EU* [105, 106, 187], *Cora* [120] and *CiteseerX* [146] network datasets. We compare the DProbWS with the methods exposed in [45, 54, 194] referred as ARW, GTG and LLUD respectively. For further details regarding the datasets, comparison methods, and graph construction, please refer to Appendix A.3.

Due to the high sparsity of the Cora and CiteseerX networks, many of the nodes are zero-knowledge nodes and therefore its label can not be inferred (see Remark 3.1). To remedy this, we make use of the teleporting random walker (TRW) as explained in Section 2.4.1. Since both ARW [45] and DProbWS methods are based on the random walker, we can implement the variants ARWtrw and DprobWstrw, which make use of the TRW. We set the η value equal to 10^{-6} for all datasets except for the *Digits* dataset, where $\eta = 10^{-2}$.

To evaluate the methods we use the accuracy (number of correctly labeled nodes divided by the total number of nodes). In the computation of the accuracy, we also include the zero-knowledge nodes. Inspired by [45], we sample a certain fraction r of all nodes from each

⁶Code publicly available at https://github.com/hci-unihd/Directed_Probabilistic_Watershed.git

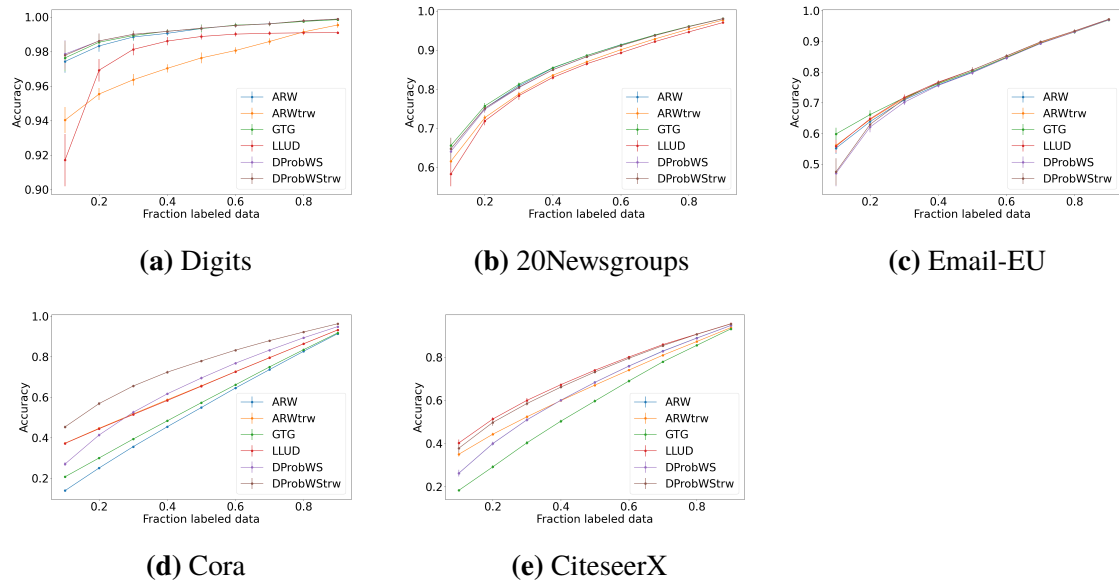


Figure 2.3. DProbWS Performance Comparison. Accuracy comparisons on the *Digits*, *20Newsgroups*, *Email-EU*, *Cora* and *CiteseerX* datasets. In all plots, the horizontal axis denotes the fraction of labeled nodes in the graph varying from 0.1 to 0.9 with 0.1 increment. We depict the average accuracy over 20 runs with error bars of one standard deviation. Our methods DProbWS and DProbWStrw are on par with existing methods and sometimes outperform them.

class uniformly as seeds. In Figure 2.3, we show the average accuracy over 20 runs for each of the r values between 0.1 and 0.9. We observe that our method obtains comparable empirical results sometimes outperforming all competing methods.

2.7 Conclusion

In this chapter, we presented an extension of the Probabilistic Watershed algorithm [57] that can be applied to directed graphs. Following the rationale exposed in the Probabilistic Watershed article [57], we defined a Gibbs distribution over all the directed spanning in-forests rooted at the seeds of a directed graph (Definition 4). We also demonstrated, using the directed version of the Matrix Tree Theorem [102], that the Directed Probabilistic Watershed is computationally and also by result equivalent to the Directed Random Walker as in the case for the undirected version. Finally, we showed that when the entropy of the Gibbs distribution is minimized, we obtain an extension of the Power Watershed potentials [41] to directed graphs.

Chapter 3

Expected Degree and Variance in Random Spanning Trees

Spanning trees serve as fundamental structures in graph theory, finding applications across diverse domains. We delve into the expected degree and variance of nodes within spanning trees of weighted graphs. We establish a probability distribution over all spanning trees of a weighted graph, where each tree's likelihood is proportional to its weight. We explore two degree variants: weighted degree, where we aggregate the edge weights of incident edges, and unweighted degree, where we count the number of neighbors. Our approach begins defining polynomials from which we derive expressions for the expectation and variance of node degree within a spanning tree. Leveraging the Matrix Tree Theorem, we establish a relation between these polynomials and the determinant of the Laplacian matrix. This enables us to derive analytical expressions for the expectation and variance of the (un)weighted degree of a node in a spanning tree, dependent ultimately on the inverse of a Laplacian submatrix. Furthermore, we also establish a relation between edge probabilities and expected node degrees. While our primary focus is on undirected graphs, we demonstrate that the results can be extended to the directed setting by considering in-trees instead.

3.1 Introduction

Graph theory serves as a foundational framework for understanding and analyzing various complex systems, ranging from social networks to transportation networks and beyond. One area of significant focus involves analyzing graph properties within random settings. Here, randomness enters the equation through the selection of edges, vertices, or entire subgraphs based on specific probability distributions. This stochastic approach offers valuable insights into the behavior of graphs under probabilistic scenarios, shedding light on their inherent structural properties and facilitating the development of efficient algorithms for diverse ap-

plications.

In the realm of random graphs, much attention has been devoted to exploring a multitude of properties, including connectivity [141], diameter [1], clustering coefficient [138], and spectral characteristics [76], among others. These investigations have provided deep insights into the emergence of specific patterns and structures in random graphs, offering valuable theoretical foundations for understanding real-world networks and guiding practical decision-making processes.

Another area of interest lies in understanding random spanning trees and their relevance. These trees, which are subgraphs of the original graph, play a crucial role in analyzing network properties such as resilience [88] or information flow [82]. Random spanning trees are essential in various fields, including computer science, physics, and biology, as they represent fundamental structures within networks. Overall, investigating random spanning trees offers a deeper understanding of the intricate relationships and emergent behaviors within complex networks.

One intriguing property of random spanning trees is the expected degree of a node. Little has been done in the literature regarding the distribution of the degree of a node in a spanning tree. It is known that for an unweighted complete graph the degree distribution of a node follows a binomial distribution given by $1 + \text{Binomial}(N - 2, \frac{1}{N})$. Willemain and Bennett examined the node degree distribution in maximum (longest) spanning trees of networks where nodes are randomly distributed in the plane or hyper-plane according to uniform or normal distributions. These distributions were estimated through Monte Carlo simulations. Pozrikidis proposed a method for calculating the node degree distribution on unweighted graphs without explicitly constructing the trees. This approach involves employing derivatives of the determinant of the Laplacian. However, while their method offers a probability distribution over all possible degrees of a node, it does not scale efficiently as the number of a node's neighbors increases, primarily due to reliance on the inclusion-exclusion principle.

In this chapter, we present analytical formulas for computing both the expected node degree and the variance in random spanning trees of weighted graphs. We define a probability distribution over all spanning trees of a weighted graph, where the probability of each tree is proportional to its weight. We consider two types of degrees: the unweighted degree, which counts only the number of neighbors, and the weighted degree, where we sum the total edge weights of the incident edges to a node. We provide analytical expressions for the expected degree and variance, which rely on the inverse of the Laplacian of the weighted graph once a node has been removed. Ultimately, our results are grounded in the Matrix Tree Theorem [94, 168].

Outline We begin with a brief overview of the notation and basic concepts in Section 3.2. In Section 3.3, we introduce two polynomials from which we derive expressions for the expectation and variance. Specifically, Theorem 3.3.2 demonstrates that the (un)weighted degree's expectation and variance can be expressed using the first and second derivatives of these polynomials, evaluated at -1 .

Moving on to Section 3.4, we utilize the Matrix Tree Theorem (Theorem 1.2.1) to establish a connection between the derivatives of these polynomials and the determinant of the Laplacian. This connection facilitates the derivation of analytical expressions for the expectation and variance, expressed in terms of the inverse of a submatrix of the Laplacian. The primary outcome of this section is outlined in Theorem 3.4.1, stating the explicit relations.

In Section 3.5, we present an alternative approach to computing the expectation of node degree. This method is stated in Theorem 3.5.1 and links the probability of an edge incident to v being present in a spanning tree with the expected degree of v . Although our results are framed in the undirected setting, Section 3.6 illustrates how these findings extend to directed graphs, focusing on the consideration of in-trees instead.¹ We then analyze the expected degrees of two toy examples in Section 3.7, and provide concluding remarks for the chapter in Section 3.8.

3.2 Notation

Throughout this chapter, we consider a weighted graph $G = (V, E, w)$, where the edge weights are defined by the function $w : E \rightarrow \mathbb{R}^+$. If $w(e) = 0$, it implies that $e \notin E$. For simplicity, we concentrate on undirected graphs. However, Section 3.6 demonstrates the applicability of our results to directed graphs as well.

For a given node $v \in V$, let $\mathcal{N}_G(v)$ and $\mathcal{E}_G(v)$ represent the set of neighbors and edges incident to v in G , respectively. This is defined as:

$$\mathcal{N}_G(v) := \{u : (u, v) \in E\}, \quad \mathcal{E}_G(v) := \{v\} \times \mathcal{N}_G(v) = \{e \in E : v \in e\}.$$

In this chapter, we distinguish between the weighted and unweighted degree of a node v . The unweighted degree of v refers to the number of neighbors of v and is denoted by $d_v(G) := |\mathcal{N}_G(v)|$. Conversely, the weighted degree represents the sum of the weights of the incident edges to v and is denoted by $d_v^w(G) := \sum_{e \in \mathcal{E}_G(v)} w(e)$. It's important to note that for unweighted graphs, both types of degrees coincide.

The set of all spanning trees of G will be denoted by \mathcal{T}_G . Following a similar approach to that taken in Chapter 2 with in-forests, we establish a probability distribution over the set

¹See Definition 2 for a definition of in-tree.

of spanning trees where the probability of each tree T is proportional to its weight. Mathematically, this can be expressed as:

$$\Pr(T) = \frac{w(T)}{w(\mathcal{T}_G)} \propto w(T). \quad (3.1)$$

For unweighted graphs, we obtain a uniform distribution over all spanning trees of G .

For a given node v , let

$$\mathcal{T}_{G,v}(k) := \{T \in \mathcal{T}_G : d_v(G) = k\}$$

represent the subset of spanning trees for which the unweighted degree of v is equal to k . Similarly, we define the subset of spanning trees for which the weighted degree of v is equal to k as

$$\mathcal{T}_{G,v}^w(k) := \{T \in \mathcal{T}_G : d_v^w(G) = k\}$$

The expected unweighted and weighted degree of v in a random tree will be given by

$$\mathbb{E}[d_T(v)] := \sum_{k=1}^{d_v(G)} k \cdot \Pr(T \in \mathcal{T}_{G,v}(k)) = \sum_{k=1}^{d_v(G)} k \cdot \frac{w(\mathcal{T}_{G,v}(k))}{w(\mathcal{T}_G)},$$

and

$$\mathbb{E}[d_T^w(v)] := \sum_{k \in D_G^w(v)} k \cdot \Pr(t \in \mathcal{T}_{G,v}^w(k)) = \sum_{k \in D_G^w(v)} k \cdot \frac{w(\mathcal{T}_{G,v}^w(k))}{w(\mathcal{T}_G)}$$

respectively. Note that in the weighted case, k iterates over the set of feasible weighted degrees, denoted here by $D_G^w(v) := \{d_T^w(v)\}_{T \in \mathcal{T}}$.

We remind the definition of the Laplacian matrix, as provided in Definition 3. The Laplacian matrix plays an important role in the Matrix Tree Theorem (Theorem 1.2.1), which forms the foundation of our results. Since we consider undirected graphs, here we define the Laplacian matrix for undirected graphs. The Laplacian matrix of a graph G is given by

$$L := D - A \quad (3.2)$$

where $A \in \mathbb{R}^{|V| \times |V|}$ is the vertex-adjacency matrix of G , represented entry-wise as $A_{uv} = w((u, v))$, and D denotes the diagonal matrix defined as $D_{uu} = \sum_{j \in V} A_{uj}$, where D_{uu} is the weighted degree of vertex u .² Recall also that for any $v \in V$, $L^{[v]}$ will stand for the Laplacian after removing the row and column indexed by v .

²Note that the definition of the Laplacian for undirected coincides with the one given in Definition 3, since for undirected graphs the adjacency matrix is symmetric, i.e. $A = A^\top$.

3.3 Polynomial-Based Computation of Expectation and Variance of Node Degree in Spanning Trees

In this section, we will define two polynomials. Their first and second derivatives will prove to be intricately connected to the expectation and variance of both the unweighted and weighted degree of a node in a spanning tree (Theorem 3.3.2).

Definition 5. Let $h(x)$ and $\mathfrak{h}(x)$ be the polynomials defined as follows:

$$h(x) = \sum_{k=1}^{|V|-1} (-1)^k w(\mathcal{T}_{G,v}(k)) x^k, \quad (3.3)$$

$$\mathfrak{h}(x) = \sum_{k \in \mathcal{D}_G^w(v)} (-1)^k w(\mathcal{T}_{G,v}^w(k)) x^k, \quad (3.4)$$

where $\mathcal{T}_{G,v}(k)$ and $\mathcal{T}_{G,v}^w(k)$ represent the sets of spanning trees with unweighted and weighted degrees equal to k , respectively. It is clear that when $x = -1$, we have

$$h(-1) = \mathfrak{h}(-1) = w(\mathcal{T}_G),$$

since both polynomials sum the weights of all spanning trees of G . The following lemma shows that for $\alpha > 0$, $h(-\alpha)$ and $\mathfrak{h}(-\alpha)$ return the weight of all spanning trees of two modified versions of G .

Lemma 3.3.1. Let $G^\alpha = (V, E, w^\alpha)$ and $\mathbf{G}^\alpha = (V, E, \mathfrak{w}^\alpha)$ be altered versions of G defined as follows:³

- $G^\alpha = (V, E, w^\alpha)$ is the graph where all edge weights of the edges incident to v have been scaled by α , i.e., with the edge-weight function given by

$$w^\alpha(e) = \begin{cases} w(e) & \text{if } e \notin \mathcal{E}_G(v), \\ \alpha w(e) & \text{otherwise} \end{cases}. \quad (3.5)$$

- $\mathbf{G}^\alpha = (V, E, \mathfrak{w}^\alpha)$ is the graph where all edge weights of the edges incident to v are scaled by α powered to the value of the edge weight, i.e., with the edge-weight function given by

$$\mathfrak{w}^\alpha(e) = \begin{cases} w(e) & \text{if } e \notin \mathcal{E}_G(v), \\ \alpha^{w(e)} w(e) & \text{otherwise} \end{cases}. \quad (3.6)$$

Then, for $\alpha > 0$, we have that $h(-\alpha) = w^\alpha(\mathcal{T}_{G^\alpha})$ and $\mathfrak{h}(-\alpha) = \mathfrak{w}^\alpha(\mathcal{T}_{\mathbf{G}^\alpha})$.

³Note that both G^α and \mathbf{G}^α depend on the node v . Nonetheless, we omit this dependence in the notation since v is arbitrary but fixed node throughout the section.

Proof: We will only prove the h case since the other one is analogous. First of all, we notice that for each $T \in \mathcal{T}_{G,v}^w(k)$, we have

$$\begin{aligned} \alpha^k w(T) &= \alpha^k \prod_{e \in E_T} w(e) = \alpha^k \prod_{e \in \mathcal{E}_T(v)} w(e) \prod_{\substack{e \in E_T \\ e \notin \mathcal{E}_T(v)}} w(e) \stackrel{*}{=} \prod_{e \in \mathcal{E}_T(v)} w(e) \alpha^{w(e)} \prod_{\substack{e \in E_T \\ e \notin \mathcal{E}_T(v)}} w(e) \\ &= \prod_{e \in \mathcal{E}_T(v)} w^\alpha(e) \prod_{\substack{e \in E_T \\ e \notin \mathcal{E}_T(v)}} w^\alpha(e) = \mathbf{w}^\alpha(T) \end{aligned} \quad (3.7)$$

where in (*) we have used the fact that $\sum_{e \in \mathcal{E}_T(v)} w(e) = k$ since $T \in \mathcal{T}_{G,v}^w(k)$. From here, it follows easily that

$$w(\mathcal{T}_{G,v}(k)) = \sum_{T \in \mathcal{T}_{G,v}(k)} w(T) \alpha^k = \sum_{T \in \mathcal{T}_{G,v}(k)} \mathbf{w}^\alpha(T) = \mathbf{w}^\alpha(\mathcal{T}_{G,v}(k))$$

and therefore

$$\begin{aligned} h(-\alpha) &= \sum_{k=1}^{|V|-1} (-1)^k w(\mathcal{T}_{G,v}(k)) (-\alpha)^k = \sum_{k=1}^{|V|-1} w(\mathcal{T}_{G,v}(k)) \alpha^k \\ &= \sum_{k=1}^{|V|-1} \mathbf{w}^\alpha(\mathcal{T}_{G^\alpha,v}(k)) = \mathbf{w}^\alpha(\mathcal{T}_{G^\alpha}) \end{aligned} \quad (3.8)$$

□

The next result relates the derivatives of the polynomials h and \mathbf{h} with the expectation and variance of the (un)weighted degree of a node in a random spanning tree.

Theorem 3.3.2. Let $G = (V, E, w)$ be a graph and v be an arbitrary but fixed node. Then the expected and variance of the unweighted and weighted degrees of node v in G are given by:

$$\mathbb{E}[d_T(v)] = -\frac{h'(-1)}{h(-1)}, \quad \mathbb{E}[d_T^w(v)] = -\frac{\mathbf{h}'(-1)}{\mathbf{h}(-1)} \quad (3.9)$$

$$\text{Var}[d_T(v)] = \frac{\partial \left(-x \frac{h'(-x)}{h(-x)} \right)}{\partial x} \Bigg|_{x=1} = \frac{h''(-1)}{h(-1)} - \frac{h'(-1)h'(-1)}{h^2(-1)} - \frac{h'(-1)}{h(-1)}. \quad (3.10)$$

$$\text{Var}[d_T^w(v)] = \frac{\partial \left(-x \frac{\mathbf{h}'(-x)}{\mathbf{h}(-x)} \right)}{\partial x} \Bigg|_{x=1} = \frac{\mathbf{h}''(-1)}{\mathbf{h}(-1)} - \frac{\mathbf{h}'(-1)\mathbf{h}'(-1)}{\mathbf{h}^2(-1)} - \frac{\mathbf{h}'(-1)}{\mathbf{h}(-1)}. \quad (3.11)$$

where h and \mathbf{h} are defined as in (3.3) and (3.4).

Proof: The proof will focus on the unweighted case, since the proof for the weighted degree follows mutatis mutandis. The derivative of h is given by:

$$h'(x) = \frac{\partial h(x)}{\partial x} = \sum_{k=1}^{|V|} (-1)^k k w(\mathcal{T}_{G,v}(k)) x^{k-1} \quad (3.12)$$

Therefore, combining (3.8) and (3.12), we easily derive (3.9):

$$\begin{aligned} -\frac{h'(-1)}{h(-1)} &= -\frac{\sum_{k=1}^{|V|} (-1)^k k w(\mathcal{T}_{G,v}(k)) (-1)^{k-1}}{w(\mathcal{T}_G)} = \frac{\sum_{k=1}^{|V|} k w(\mathcal{T}_{G,v}(k))}{w(\mathcal{T}_G)} \\ &= \sum_{k=1}^{|V|} k \Pr(\mathcal{T}_G(k)) = \mathbb{E}[\mathbf{d}_T(v)] \end{aligned} \quad (3.13)$$

Now we will prove (3.10). Based on equation (3.12), we deduce that:

$$\begin{aligned} x \frac{\partial x h'(x)}{\partial x} &= x \frac{\partial \sum_{k=0}^{|V|} (-1)^k k w(\mathcal{T}_{G,v}(k)) x^k}{\partial x} \\ &= x \sum_{k=1}^{|V|} (-1)^k k^2 w(\mathcal{T}_{G,v}(k)) x^{k-1} = \sum_{k=0}^{|V|} (-1)^k k^2 w(\mathcal{T}_{G,v}(k)) x^k \end{aligned} \quad (3.14)$$

On the other hand, using the chain rule, we obtain:

$$x \frac{\partial [x h'(x)]}{\partial x} = x^2 h''(x) + x h'(x) \quad (3.15)$$

Following the same reasoning as in equation (3.13), we obtain

$$\begin{aligned} \frac{h''(-1) - h'(-1)}{h(-1)} &= \frac{\sum_{k=0}^{|V|} k^2 w(\mathcal{T}_{G,v}(k)) (-1)^{2k}}{h(-1)} = \frac{\sum_{k=0}^{|V|} k^2 w^\alpha(\mathcal{T}_{G,v}(k))}{w(\mathcal{T}_G)} \\ &= \sum_{k=0}^{|V|} k^2 \Pr(t \in (\mathcal{T}_G(k))) = \mathbb{E}[(\mathbf{d}_T(v))^2] \end{aligned} \quad (3.16)$$

Consequently, we have

$$\text{Var}[\mathbf{d}_T(v)] = \mathbb{E}[(\mathbf{d}_T(v))^2] - (\mathbb{E}[\mathbf{d}_T(v)])^2 = \frac{h''(-1)}{h(-1)} - \frac{h'(-1)}{h(-1)} - \frac{h'(-1)h'(-1)}{h^2(-1)} \quad (3.17)$$

Finally, notice that

$$\begin{aligned} \frac{\partial \left(-x \frac{h'(-x)}{h(-x)} \right)}{\partial x} \Big|_{x=1} &= \left[\frac{x h''(-x)}{h(-x)} - \frac{h'(-x)}{h(-x)} - \frac{x h'(-x) h'(-x)}{h^2(-x)} \right] \Big|_{x=1} \\ &= \frac{h''(-1)}{h(-1)} - \frac{h'(-1)}{h(-1)} - \frac{h'(-1)h'(-1)}{h^2(-1)} \underbrace{=}_{\text{Eq. (3.17)}} \text{Var}[\mathbf{d}_T(v)] \end{aligned} \quad (3.18)$$

□

3.4 Laplacian-Based Computation of Expectation and Variance of Node Degree in Spanning Trees

In this section, leveraging the Matrix Tree Theorem, we first discover that the polynomials defined in Definition 5 can be represented using the determinants of Laplacians from two

modified versions of the graph G . These modifications involve scaling the weights of edges incident to a node v by a factor determined by a variable α . Notably, this variable α corresponds to the negated input variable x of the polynomials. Ultimately, building on the insights of Theorem 3.3.2, we establish in Theorem 3.4.1 a connection between the derivatives of these polynomials and the determinant of the Laplacian. This connection allows us to derive analytical expressions for the expectation (un)weighted degree and its variance, expressed in terms of the inverse of a submatrix of the Laplacian.

Given a node v in an undirected, edge-weighted, connected graph $G = (V, E, w)$ with n nodes, let $L_{G_v} \in \mathbb{R}^{n \times n}$ denote the Laplacian matrix of the subgraph $G_v = (V, \mathcal{E}_G(v), w)$, which consists of only the edges incident to v . Formally,

$$[L_{G_v}]_{ij} = \begin{cases} -w((i, j)) & \text{if } v \in (i, j) \text{ \& } i \neq j \\ d_G^w(v) & \text{if } i = j = v \\ 0 & \text{otherwise} \end{cases} \quad (3.19)$$

Analogously, we define the Laplacian matrix $\alpha^{L_{G_v}}$ of the graph that consists of the edges incident to v whose weights have been scaled by $\alpha^{w(e)}$, i.e.

$$[\alpha^{L_{G_v}}]_{ij} = \begin{cases} -w((i, j))\alpha^{w((i, j))} & \text{if } v \in (i, j) \text{ \& } i \neq j \\ \sum_{\ell \in \mathcal{N}_G(v)} w((v, \ell))\alpha^{w((v, \ell))} & \text{if } i = j = v \\ 0 & \text{otherwise} \end{cases}.$$

Let L_G be the Laplacian matrix of G , and define $\bar{L} = L_G - L_{G_v}$. Thus, the Laplacians of the graphs G^α and G^α defined in Lemma 3.3.1 are given by:

$$L_{G^\alpha} = \bar{L} + \alpha L_{G_v}, \quad L_{G^\alpha} = \bar{L} + \alpha^{L_{G_v}} \quad (3.20)$$

Given an arbitrary node r of G , we deduce from the Matrix Tree Theorem (Theorem 1.2.1) and Lemma 3.3.1 that

$$\det \left(L_{G^\alpha}^{[r]} \right) = \det \left((\bar{L} + \alpha L_{G_v})^{[r]} \right) = w^\alpha (\mathcal{T}_{G^\alpha}) = h(-\alpha) \quad (3.21)$$

$$\det \left(L_{G^\alpha}^{[r]} \right) = \det \left((\bar{L} + \alpha^{L_{G_v}})^{[r]} \right) = w^\alpha (\mathcal{T}_{G^\alpha}) = h(-\alpha) \quad (3.22)$$

where the exponent $[r]$ represents that the row and column indexed by r have been removed from the matrix. The next theorem leverages these equations to obtain expressions for the expectation and variance of the (un)weighted degree in terms of the Laplacian matrix.

Theorem 3.4.1. Let $G = (V, E, w)$ be an undirected, edge-weighted connected graph and r an arbitrary node of G . For a given node v , we have that:

(A) The expected unweighted and weighted degrees of v in a tree $t \in \mathcal{T}_G$ are given by:

$$\begin{aligned} \text{(a)} \quad \mathbb{E} [d_{\mathcal{T}}(v)] &= \text{Tr} \left[L_{G_v}^{[r]} \left(L_G^{[r]} \right)^{-1} \right] \\ \text{(b)} \quad \mathbb{E} [d_{\mathcal{T}}^w(v)] &= \text{Tr} \left[\left(L_{G_v}^{[r]} \right)^{\odot 2} \left(L_G^{[r]} \right)^{-1} \right] \end{aligned}$$

(B) The variance of the unweighted and weighted degree of v in a tree $t \in \mathcal{T}_G$ are given by:

$$\begin{aligned} \text{(a)} \quad \text{Var} [d_{\mathcal{T}}(v)] &= \text{Tr} \left[L_{G_v}^{[r]} \left(L_G^{[r]} \right)^{-1} \left(I - L_{G_v}^{[r]} \left(L_G^{[r]} \right)^{-1} \right) \right] \\ \text{(b)} \quad \text{Var} [d_{\mathcal{T}}^w(v)] &= \text{Tr} \left[\left(L_{G_v}^{[r]} \right)^{\odot 3} \left(L_G^{[r]} \right)^{-1} - \left(L_{G_v}^{[r]} \right)^{\odot 2} \left(L_G^{[r]} \right)^{-1} \left(L_{G_v}^{[r]} \right)^{\odot 2} \left(L_G^{[r]} \right)^{-1} \right] \end{aligned}$$

where L_G is the Laplacian of G , L_{G_v} is the matrix defined in (3.19), $L_G^{[r]}$ and $L_{G_v}^{[r]}$ represent the corresponding matrices once the row and column indexed by node r have been removed, Tr is the trace operator, and the exponent $L_{G_v}^{\odot p}$ represents the Laplacian of the graph when only the edges incident to v are considered and have been raised to the power of p , i.e.,

$$[L_{G_v}^{\odot p}]_{ij} = \begin{cases} -w((i, j))^p & \text{if } v \in (i, j) \text{ \& } i \neq j \\ \sum_{\ell \in \mathcal{N}_G(v)} w((v, \ell))^p & \text{if } i = j = v \\ 0 & \text{otherwise} \end{cases}.$$

Proof: We will only demonstrate the case for the unweighted degree, as the reasoning for the weighted case is analogous. First, notice the following relation:

$$x \frac{\partial \log h(-x)}{\partial x} = -x \frac{h'(-x)}{h(-x)} \quad (3.23)$$

Additionally, from equation (3.9) stated in Theorem 3.3.2, we know that $\mathbb{E} [d_{\mathcal{T}}(v)] = -\frac{h'(-1)}{h(-1)}$. Therefore, in combination with equation (3.21), we deduce that

$$\mathbb{E} [d_{\mathcal{T}}(v)] = \alpha \frac{\partial \log h(-\alpha)}{\partial \alpha} \Big|_{\alpha=1} = \alpha \frac{\partial \log \det \left(L_{G^\alpha}^{[r]} \right)}{\partial \alpha} \Big|_{\alpha=1} = \alpha \text{Tr} \left[\frac{\partial L_{G^\alpha}^{[r]}}{\partial \alpha} \left(L_{G^\alpha}^{[r]} \right)^{-1} \right] \Big|_{\alpha=1}. \quad (3.24)$$

Since

$$\frac{\partial L_{G^\alpha}^{[r]}}{\partial \alpha} \Big|_{\alpha=1} = \frac{\partial \left(\bar{L} + \alpha L_{G_v} \right)^{[r]}}{\partial \alpha} \Big|_{\alpha=1} = L_{G_v}^{[r]},$$

it follows from (3.24)

$$\mathbb{E} [d_{\mathcal{T}}(v)] = \alpha \text{Tr} \left[L_{G_v}^{[r]} \left(L_{G^\alpha}^{[r]} \right)^{-1} \right] \Big|_{\alpha=1} = \text{Tr} \left[L_{G_v}^{[r]} \left(L_G^{[r]} \right)^{-1} \right]. \quad (3.25)$$

Now we will prove the variance equality stated in the theorem. We know from equation (3.13) of Theorem 3.3.2 that

$$\text{Var} [d_T(v)] = \left. \frac{\partial \left(-x \frac{h'(-x)}{h(-x)} \right)}{\partial x} \right|_{x=1}.$$

Thus, we need to derive (3.25) again with respect to α and evaluate it at 1:

$$\begin{aligned} \text{Var} [d_T(v)] &= \left. \frac{\partial \alpha \text{Tr} \left[L_{G_v}^{[r]} \left(L_{G^\alpha}^{[r]} \right)^{-1} \right]}{\partial \alpha} \right|_{\alpha=1} \\ &= \text{Tr} \left[L_{G_v}^{[r]} \left(L_{G^\alpha}^{[r]} \right)^{-1} - \alpha L_{G_v}^{[r]} \left(L_{G^\alpha}^{[r]} \right)^{-1} L_{G_v}^{[r]} \left(L_{G^\alpha}^{[r]} \right)^{-1} \right] \Big|_{\alpha=1} \\ &= \text{Tr} \left[L_{G_v}^{[r]} \left(L_G^{[r]} \right)^{-1} \left(I - L_{G_v}^{[r]} \left(L_G^{[r]} \right)^{-1} \right) \right]. \end{aligned} \tag{3.26}$$

Note that we applied the following derivative rules for matrices (see [142]):

$$\begin{aligned} \partial (\log (\det(X))) &= \text{Tr} [(\partial X)X^{-1}], & \partial (\text{Tr}(X)) &= \text{Tr}(\partial X), \\ \partial (XY) &= \partial(X)Y + X\partial(Y), & \partial (X^{-1}) &= -X^{-1}(\partial X)X^{-1}. \end{aligned} \tag{3.27}$$

□

Remark 3.4.2 (Computation entire probability distribution of node degree). It should be noted that while our focus has been on computing the expectation and variance of the degree of a node in a spanning tree, equations (3.21) and (3.22) offer methods for calculating the entire probability distribution of the node degree. By evaluating the determinants with respect to α and dividing by the total weight of the trees, the absolute values of the coefficients multiplying the monomials α^i reveal the probability of sampling a tree where the degree of node v is equal to i . This method appears to offer greater efficiency compared to the approach outlined in [144], which also relies on determinant computation but additionally employs the inclusion-exclusion principle on edges incident to v . Consequently, it may not scale optimally with the number of neighbors of v . Furthermore, our method extends to weighted graphs, whereas the method described in the aforementioned reference is limited to unweighted ones.

Remark 3.4.3 (Weighted degree with weights independent of the edge-weights determining the tree probability distribution). The weighted degree of a node v has been defined as the sum of the edge-weights, denoted as $w(e)$, of all edges incident to v . While these edge-weights influence the probability distribution of the tree, we can extend this concept by considering alternative edge-weights that are independent of the tree's probability distribution. Let $\omega : E \rightarrow \mathbb{R}$ be a secondary edge-weight function, allowing us to define the

weighted degree as $d_v^\omega(G) := \sum_{e \in \mathcal{E}_G(v)} \omega(e)$. Note that unlike $w(\cdot)$, $\omega(\cdot)$ is not constrained to be non-negative. In this framework, Theorem 3.3.2 remains valid if we instead consider the graph $G^\alpha = (V, E, w^\alpha)$ when applying Lemma 3.3.1, where w^α is redefined as follows:

$$w^\alpha(e) = \begin{cases} w(e) & \text{if } e \notin \mathcal{E}_G(v), \\ \alpha^{\omega(e)} w(e) & \text{otherwise} \end{cases}. \quad (3.28)$$

This formulation scales the edge-weights determining the tree probability distribution by $\alpha^{\omega(e)}$ for edges incident to v , instead of weighting by $\alpha^{w(e)}$ as defined in equation (3.6). Consequently, we can adapt Theorem 3.4.1 to derive the following expressions for the expectation and variance:

$$\begin{aligned} \mathbb{E}[d_T^\omega(v)] &= \text{Tr} \left[\left(\Lambda_{G_v}^{[r]} \right)^{\odot 1} \left(L_G^{[r]} \right)^{-1} \right] \\ \text{Var}[d_T^\omega(v)] &= \text{Tr} \left[\left(\Lambda_{G_v}^{[r]} \right)^{\odot 2} \left(L_G^{[r]} \right)^{-1} - \left(\Lambda_{G_v}^{[r]} \right)^{\odot 1} \left(L_G^{[r]} \right)^{-1} \left(\Lambda_{G_v}^{[r]} \right)^{\odot 1} \left(L_G^{[r]} \right)^{-1} \right] \end{aligned}$$

where $\Lambda_{G_v}^{\odot p}$ represents the Laplacian of the graph when only the edges incident to v are considered, with weights given by $w(e)\omega(e)^p$, i.e.,

$$[\Lambda_{G_v}^{\odot p}]_{ij} = \begin{cases} -w(i, j)\omega((i, j))^p & \text{if } v \in (i, j) \text{ \& } i \neq j \\ \sum_{\ell \in \mathcal{N}_G(v)} w(v, \ell)\omega((v, \ell))^p & \text{if } i = j = v \\ 0 & \text{otherwise} \end{cases}.$$

The remaining symbols are defined as in Theorem 3.4.1.

Example 3.4.4 (When node v connects to all nodes with equal weights). Consider the scenario where node v is connected to all other nodes with equal edge weights set to κ . Let $\bar{G} = (V \setminus \{v\}, E \setminus \mathcal{E}_G(v))$ represent the graph obtained by removing node v from G . In this case, according to equation (3.19), the Laplacian matrix $L_{G_v}^{[v]}$ is κ times the identity matrix. Thus,

$$L_G^{[v]} = L_{\bar{G}} + L_{G_v}^{[v]} = L_{\bar{G}} + \kappa I^{[v]}.$$

Let $\{\mu_i\}$ and $\{\lambda_i\}$ be the eigenvalues of $L_G^{[v]}$ and $L_{\bar{G}}$, respectively. They are related as follows:

$$\mu_i = \lambda_i + \kappa.$$

This relation can be understood by observing that

$$\mu_i \nu_i = L_G^{[v]} \nu_i = (L_{\bar{G}} + \kappa I) \nu_i = L_{\bar{G}} \nu_i + \kappa \nu_i = (\lambda_i + \kappa) \nu_i,$$

where ν_i is an eigenvector of $L_G^{[v]}$ and $L_{\bar{G}}$.

Thus, from Theorem 3.4.1, it follows that when a node is connected to all other nodes with equal edge weights, the expected unweighted degree of node v is related to the eigenvalues of the Laplacian of the graph without v as follows:

$$\mathbb{E} [d_T(v)] = \text{Tr} \left(\underbrace{L_{G_v}^{[v]}}_{\kappa I^{[v]}} \left(L_G^{[v]} \right)^{-1} \right) = \text{Tr} \left(\kappa \left(L_G^{[v]} \right)^{-1} \right) = \sum_{i=1}^{|V|-1} \frac{\kappa}{\mu_i} = \sum_{i=1}^{|V|-1} \frac{\kappa}{\lambda_i + \kappa}. \quad (3.29)$$

We have utilized the fact that the trace of a matrix equals the sum of its eigenvalues, and that the eigenvalues of the inverse of a matrix are the reciprocals of the eigenvalues of the original matrix. Analogously, for the variance, we obtain:

$$\begin{aligned} \text{Var} [d_T(v)] &= \text{Tr} \left[\kappa \left(L_G^{[v]} \right)^{-1} - \kappa^2 \left(L_G^{[v]} \right)^{-2} \right] \\ &= \sum_{i=1}^{|V|-1} \frac{\kappa}{\lambda_i + \kappa} - \left(\frac{\kappa}{\lambda_i + \kappa} \right)^2 = \sum_{i=1}^{|V|-1} \frac{\kappa \lambda_i}{(\lambda_i + \kappa)^2} \end{aligned} \quad (3.30)$$

Similarly, for the weighted degree, we obtain the following expressions

$$\mathbb{E} [d_T^w(v)] = \kappa \mathbb{E} [d_T(v)] = \sum_{i=1}^{|V|-1} \frac{\kappa^2}{\lambda_i + \kappa}, \quad (3.31)$$

$$\text{Var} [d_T^w(v)] = \kappa^2 \text{Var} [d_T(v)] = \sum_{i=1}^{|V|-1} \frac{\kappa^3 \lambda_i}{(\lambda_i + \kappa)^2}. \quad (3.32)$$

For a complete unweighted graph with N nodes, the eigenvalues of its Laplacian matrix are $\lambda_1 = 0$ and $\lambda_i = N$ for $2 \leq i \leq N$. As a consequence of equations (3.29) and (3.30), we can determine the expectation and variance of a node's degree in a random spanning tree with N nodes:

$$\begin{aligned} \mathbb{E} [d_T(v)] &= \sum_{i=1}^{N-1} \frac{1}{\lambda_i + 1} = 1 + \sum_{i=1}^{N-2} \frac{1}{N-1+1} = 1 + \frac{N-2}{N}, \\ \text{Var} [d_T(v)] &= \sum_{i=1}^{N-1} \frac{\lambda_i}{(\lambda_i + 1)^2} = 1 + \sum_{i=1}^{N-2} \frac{N-1}{(N-1+1)^2} = 1 + \frac{(N-2)(N-1)}{N^2}. \end{aligned}$$

These values match the expectation and variance of $1 + \text{Binomial}(N-2, \frac{1}{N})$, which represents the degree distribution of a fixed vertex in a uniformly random spanning tree with N nodes [144].

3.5 Relation Between Edge Probability and Expected Node Degree in Spanning Trees

In this section, we will establish the relationship between the expected (un)weighted degree and the presence probability of the edges incident to node v in a spanning tree. Specifically,

we demonstrate that the (weighted) sum of edge presence probabilities, when summed over the edges incident to v , equals the (weighted) unweighted expected degree.

Theorem 3.5.1. Given a node v in an undirected edge-weighted connected graph G , the expected unweighted and weighted degrees of node v in a random spanning tree $T \in \mathcal{T}_G$ are given by

- (a) $\mathbb{E}[d_T(v)] = \sum_{k=1}^{|V|} k \cdot \Pr(T \in \mathcal{T}_{G,v}(k)) = \sum_{e \in \mathcal{E}_G(v)} \Pr(e \in T)$,
 (b) $\mathbb{E}[d_T^w(v)] = \sum_{k \in d_T^w(v)} k \cdot \Pr(T \in \mathcal{T}_{G,v}^w(k)) = \sum_{e \in \mathcal{E}_G(v)} w(e) \Pr(e \in T)$.

Here, $\Pr(e \in T)$ represents the probability of edge e being present in a spanning tree of G .

Proof:

- (a) Using Bayes' theorem, we derive the following expressions:

$$\begin{aligned} \sum_{e \in \mathcal{E}_G(v)} \Pr(e \in T) &= \sum_{e \in \mathcal{E}_G(v)} \sum_{k=1}^{|V|} \Pr(T \in \mathcal{T}_{G,v}(k)) \Pr(e \in T | T \in \mathcal{T}_{G,v}(k)) \\ &= \sum_{k=1}^{|V|} \Pr(T \in \mathcal{T}_{G,v}(k)) \sum_{e \in \mathcal{E}_G(v)} \Pr(e \in T | T \in \mathcal{T}_{G,v}(k)). \end{aligned} \quad (3.33)$$

Therefore we just need to show that $\sum_{e \in \mathcal{E}_G(v)} \Pr(e \in T | T \in \mathcal{T}_{G,v}(k)) = k$.

$$\begin{aligned} \sum_{e \in \mathcal{E}_G(v)} \Pr(e \in T | T \in \mathcal{T}_{G,v}(k)) &= \sum_{e \in \mathcal{E}_G(v)} \sum_{\substack{T \in \mathcal{T}_{G,v}(k) \\ \text{s.t. } e \in T}} \frac{w(T)}{w(\mathcal{T}_{G,v}(k))} \\ &= \sum_{T \in \mathcal{T}_{G,v}(k)} \frac{k \cdot w(T)}{w(\mathcal{T}_{G,v}(k))} = \frac{k \cdot w(\mathcal{T}_{G,v}(k))}{w(\mathcal{T}_{G,v}(k))} = k \end{aligned} \quad (3.34)$$

In (*), we utilized the fact that any tree $T \in \mathcal{T}_{G,v}(k)$ will have k edges from $e \in \mathcal{E}_G(v)$, thus $w(T)$ will be counted k times.

- (b) The reasoning for the weighted case is similar to the unweighted one. Analogously to equation (3.33), in the weighted case we have that

$$\begin{aligned} \sum_{e \in \mathcal{E}_G(v)} w(e) \Pr(e \in T) &= \sum_{e \in \mathcal{E}_G(v)} w(e) \sum_{k=1}^{|V|} \Pr(T \in \mathcal{T}_{G,v}^w(k)) \Pr(e \in T | T \in \mathcal{T}_{G,v}^w(k)) \\ &= \sum_{k=1}^{|V|} \Pr(T \in \mathcal{T}_{G,v}^w(k)) \sum_{e \in \mathcal{E}_G(v)} w(e) \Pr(e \in T | T \in \mathcal{T}_{G,v}^w(k)). \end{aligned} \quad (3.35)$$

By demonstrating that the summation $\sum_{e \in \mathcal{E}_G(v)} w(e) \Pr(e \in T | T \in \mathcal{T}_{G,v}^w(k))$ equals k , we establish the desired result.

$$\begin{aligned}
 \sum_{e \in \mathcal{E}_G(v)} w(e) \Pr(e \in \mathbf{T} \mid \mathbf{T} \in \mathcal{T}_{G,v}^w(k)) &= \sum_{e \in \mathcal{E}_G(v)} w(e) \sum_{\substack{t \in \mathcal{T}_{G,v}^w(k) \\ \text{s.t. } e \in t}} \frac{w(t)}{w(\mathcal{T}_{G,v}^w(k))} \\
 &\stackrel{*}{=} \sum_{\mathbf{T} \in \mathcal{T}_{G,v}^w(k)} \frac{k \cdot w(\mathbf{T})}{w(\mathcal{T}_{G,v}^w(k))} = \frac{k \cdot w(\mathcal{T}_{G,v}^w(k))}{w(\mathcal{T}_{G,v}^w(k))} = k
 \end{aligned} \tag{3.36}$$

In (*) we used the fact that any tree $\mathbf{T} \in \mathcal{T}_{G,v}^w(k)$ will consist of a subset of edges from $\mathcal{E}_G(v)$ whose total weight equals k . □

The following theorem outlines how to compute the presence probability of an edge in a random spanning tree in terms of $(L^{[r]})^{-1}$. With this relation established, deriving the formula for the expected degree obtained in the previous section becomes straightforward. However, from such an expression, it is not immediately clear how to derive the expression for the variance.

Theorem 3.5.2. Given a connected edge-weighted undirected graph $G = (V, E, w)$, let \mathcal{T}_G stand for the set of all spanning trees of G . For an edge $e = (u, v) \in E$, the probability that e belongs to spanning tree $\mathbf{T} \in \mathcal{T}_G$ is given by

$$\Pr(e \in \mathbf{T}) = \begin{cases} w(e) \left(\ell_{uu}^{-1,[r]} + \ell_{vv}^{-1,[r]} - 2\ell_{uv}^{-1,[r]} \right) & \text{if } r \neq u, v \\ w(e)\ell_{uu}^{-1,[v]} & \text{if } r = v \\ w(e)\ell_{vv}^{-1,[u]} & \text{if } r = u \end{cases}, \tag{3.37}$$

where $\ell_{ij}^{-1,[r]}$ denotes the entry ij of the inverse of the matrix $L_G^{[r]}$ (the Laplacian L_G after removing the row and the column corresponding to node r).

Proof: We postpone the detailed proof to Appendix B.1. However, here we outline its main concept. The proof relies on the fact that the set of spanning trees \mathcal{T}_G containing edge e is equal to the total number of spanning trees minus the ones not containing edge e . Calculating the latter involves enumerating the total number of spanning trees of the graph $G \setminus e$, obtained by removing edge e from G . Leveraging the Matrix Tree Theorem (Theorem 1.2.1), we can determine the number of trees for each graph by utilizing their respective Laplacian matrices. This, combined with the observation that the difference between the Laplacians of these two graphs yields a one-rank matrix, allows us to apply the Determinant Lemma (Lemma 2.3.4) to compute the probability. □

Remark 3.5.3. It is noteworthy to mention that the probability that an edge is present in a random spanning tree is closely related with the effective resistance, since $\Pr(e \in \mathbf{T}) = w(e)r_e$, where r_e is the effective resistance between the extreme nodes of edge e (See Section 2.5.2 [60] and [116]).

3.6 Extension to Directed Graphs

We can extend the results obtained in the previous sections to the directed case. In this section, we will briefly expose how the theorems generalize when a directed graph is considered.

The cornerstone of our derivations in the previous sections is the Matrix Tree Theorem for undirected graphs. In Theorem 2.2.1, we introduced the directed version of the Matrix Tree Theorem, which asserts that

$$w(\mathcal{T}^{\vec{r}}) = \det(L^{[r]}).$$

where $\mathcal{T}^{\vec{r}}$ represents the set of in-trees rooted at r (see Definition 2). Thus, it follows, with appropriate adjustments, from the previous theorems that the same formulas can be applied to compute the expected (un)weighted degree of any node v and its variance in a random spanning in-tree rooted at r . This holds true when considering the analogous probability distribution defined over the spanning trees, as stated in (3.1), but in this case applied to all in-trees rooted at r .

3.7 Toy Examples

Figures 3.1 and 3.2 depict the expected weighted degree and its standard deviation (square root of variance) of an unweighted grid graph and a weighted kNN graph, respectively.⁴ In addition, we also show relative expected degree and standard deviation, where we divide the respective values by the actual degree of the node in the graph. This allows for an assessment of the average proportion of edges incident to a node present in a spanning tree out of all incident edges to the node.

In the case of the grid graph, as shown in Figure 3.1, the high regularity allows us to distinguish between two types of nodes based on their position within the grid: those located on the border and those not on the border. Notably, nodes with high expected degree exhibit lower relative expected degree, and vice versa. This phenomenon arises from the fact that non-border nodes possess a higher number of neighbors, thus they can be reached by any of these neighbors; however, on average, a lower proportion of these incident edges are utilized. In contrast, border nodes have fewer connections, making it more challenging to connect them to the rest of nodes. Consequently, the few connections they have tend to show up more frequently in the tree, leading to a higher proportion of incident edges being used.

Similarly, in the case of the kNN graph weighted degree exploration (Figure 3.2), we observe a comparable pattern. Many nodes with low expected degree exhibit high relative

⁴In the case of the grid graph, since the graph is unweighted, both the unweighted and weighted degrees coincide.

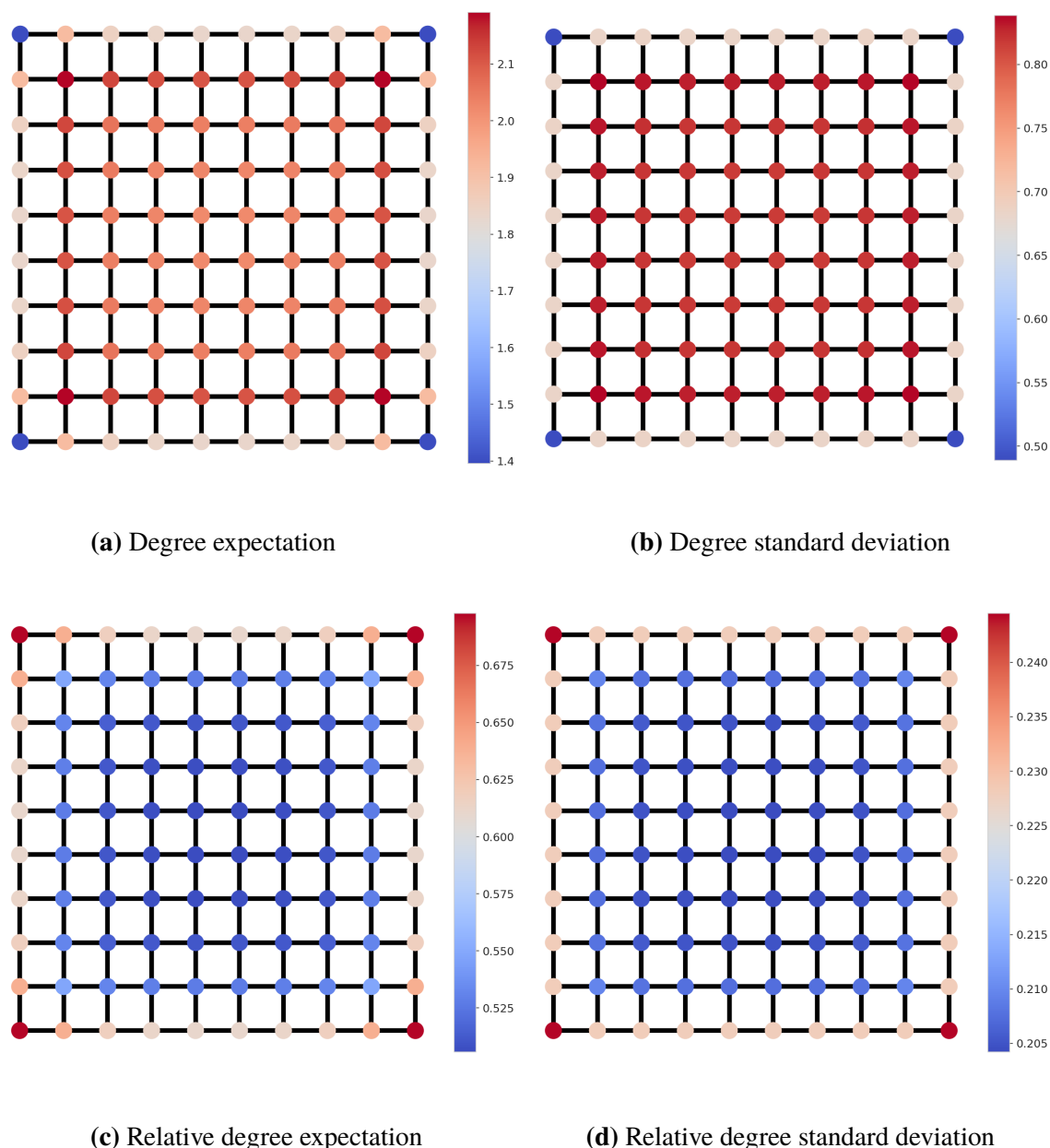


Figure 3.1. Expectation and Standard Deviation of Unweighted Degree in a Spanning Tree of a Grid Graph. 3.1a-3.1b) Expectation and standard deviation (square root of variance) of the unweighted degree. Due to the regularity of the graph, most nodes exhibit similar values, with expectations and standard deviations decreasing only at the borders. 3.1c-3.1d) Expectation and standard deviation of the unweighted degree divided by the actual unweighted degree of the node in the graph. Nodes located at the borders utilize on average a higher proportion of incident edges.

expected degree. However, not all nodes with high expected degree have low relative expected degree. This is exemplified by the node connecting the right and central "cluster" of the graph. This indicates that despite of its high weighted degree, it uses a high proportion

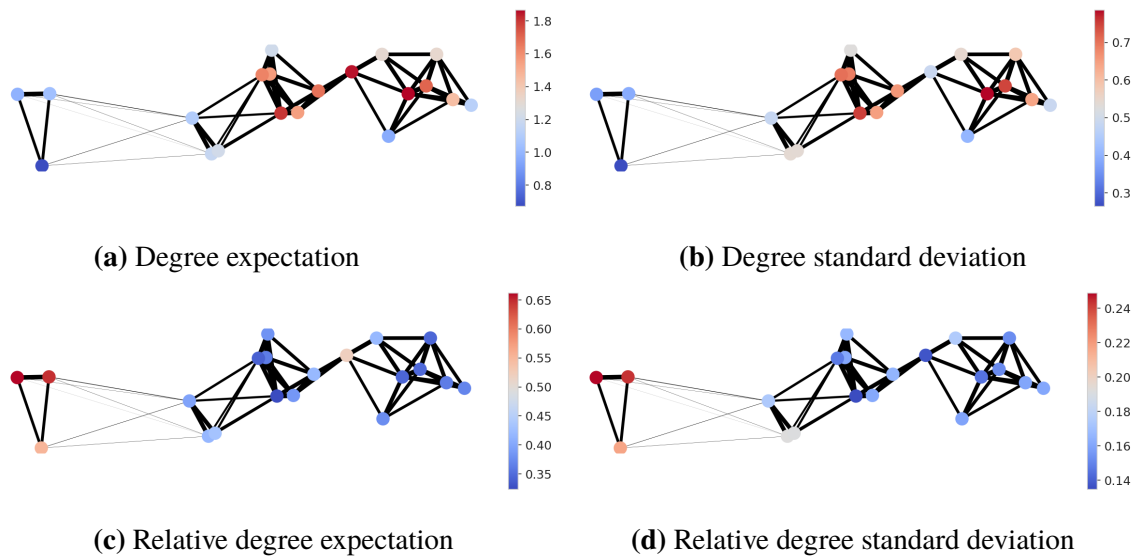


Figure 3.2. Expectation and Standard Deviation of Weighted Degree in a Spanning Tree of a Weighted Graph: kNN weighted-graph of 20 points sampled uniformly from a rectangle. The edge-weights are represented by the widths of the edges. 3.2a-3.2b) Expectation and standard deviation (square root of variance) of the weighted degree. 3.2c-3.2d) Expectation and standard deviation of the weighted degree divided by the actual weighted degree of the node in the graph. We can intuit that points with high weighted expected degree and high relative weighted degree are more crucial for connectivity, as they are connected to more nodes and their edges are utilized more often. This is exemplified by the node connecting the central and right “clusters” of the graph.

of its incident edges. Moreover, we also observe that the standard deviation at this point is low compared to other nodes. High expected degree in conjunction with relative expected degree may indicate that the node is crucial for the connectivity of the graph. Such measures may find utility in various applications, such as telecommunications or transportation. However, a thorough exploration of these applications is not within the scope of this research and is left for future work.

3.8 Conclusion

In this chapter, we have provided analytical expressions to compute the expected degree of a node in a random spanning tree of a weighted graph, as well as its variance. We have considered two types of degree: the unweighted, which counts the number of neighbors; and the weighted one, which returns the sum of the weights of the adjacent edges.

The expectation and variance are expressed in terms of the inverse of the Laplacian, once a row and a column have been removed. In addition, we have also indicated that the same

expressions extend to the directed case, where instead spanning in-trees are considered.

We believe that the provided expressions are of theoretical interest, and could have practical relevance in the future, though no specific application has been identified yet. Understanding the expected degree of a node may serve as a key parameter influencing various network dynamics and processes, such as information diffusion, routing efficiency, and resilience to failures. For example, in telecommunications and computer networks, understanding the expected degree and variance can inform the design of efficient routing protocols and network optimization strategies. Further study is postponed for future work.

Chapter 4

Algebraic Path Problem for Graph Metrics

Finding paths with optimal properties is a foundational problem in computer science. The notions of shortest paths (minimal sum of edge costs), minimax paths (minimal maximum edge weight), reliability of a path and many others all arise as special cases of the “algebraic path problem” (APP). Indeed, the APP formalizes the relation between different semirings such as min-plus, min-max and the distances they induce. We here clarify, for the first time, the relation between the potential distance and the log-semiring. We also define a new unifying family of algebraic structures that include all above-mentioned path problems as well as the commute cost and others as special or limiting cases. The family comprises not only semirings but also strong bimonoids (that is, semirings without distributivity). We call this new and very general distance the “log-norm distance”. Finally, we derive some sufficient conditions which ensure that the APP associated with a semiring defines a metric over an arbitrary graph.

4.1 Introduction

Graphs are a versatile abstraction permitting the modeling and analysis of an extremely broad range of problems from vision, NLP and learning with structured data. Measuring the similarity between the nodes in a graph is, in turn, a task of fundamental importance that often decides on success or failure of an application. Consequently, the study and development of graph node metrics with different properties is a problem of high interest.

The shortest path distance is arguably the most popular graph node metric. Given two nodes in a graph with edge costs, the shortest path problem aims to find the path with minimum cost, i.e., the path for which the sum over the cost of its constituent edges is minimized. This problem is determined by the min and $+$ operations: the $+$ to indicate that edge costs

are summed along any path; and the min to state that the overall cost is given by the smallest cost of any single path. Together, these form the “min-plus” semiring. A semiring is an algebraic structure with two operations that relaxes the concept of a ring by dropping the requirement for inverses under the “addition” operation. The algebraic path problem (APP) generalizes the notion of shortest path by replacing the min-plus semiring by an alternative semiring. Different semirings result in dramatically different preferences of paths, see the toy example in Table 4.1. This generalization encompasses a great variety of problems and applications in diverse research areas like NLP [40, 152] or routing protocols [67]. For more applications see [11, 64].

Other common graph metrics that raise from the APP framework are the commute cost distance (CCD) [32, 52, 97], which calculates the average first passage cost between two nodes in both directions; and the minimax or bottleneck shortest path distance [31, 122], which computes the path with minimal maximum edge cost among its edges.

In some situations, these metrics may fail to take the global structure of the graph into consideration. On the one hand, the minimax and shortest path distances are determined by a single path. Thus, they may ignore the topology of the surrounding graph since the degree of connectivity between the nodes is not reflected by the metrics. On the other hand, though CCD weighs all paths, it is known that for large graphs, it only takes the degree of the source and target nodes into account [115, 137].

We aim to combine the advantages and compensate for the deficiencies of these metrics. To do so, we propose a novel parametrized family of distances, dubbed log-norm distances, that interpolate between the shortest path, CCD and the minimax distances up to a constant factor. We base our interpolation on the algebraic path problem. First, we present a family of semirings whose associated APP yields a metric which interpolates between the shortest and minimax distance. Moreover, we study the potential distance [61, 95], which interpolates between the CCD and the shortest path distance. We redefine this distance via the well-known log-semiring and its associated APP. As far as we know, we are the first to formalize the interpolations of these metrics from the APP point of view. Finally, we introduce a greater parametrized set of algebraic structures. Though not all the members of this family of algebraic structures define a semiring, but strong bimonoids [49], their associated APP define the log-norm family of metrics. These distances are parametrized by a parameter r , which controls the relevance of higher cost edges in the paths; and a parameter μ , which regulates how much individual paths contribute to the distance, favoring paths with lower cost. Table 4.1 summarizes these relations and highlights, on a toy example, how drastically different distances vary. Clearly, these properties greatly impact the machine learning on graphs.

Intrigued by the fact that so many metrics can be retrieved from the APP framework, we



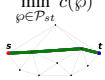





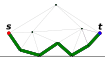
$r \backslash \mu$	0^+	$(0, \infty)$	∞
1	<p><i>Eisner semiring</i></p> <p>Commute cost distance</p> $\mathbb{E}_{\varphi \sim \mathcal{P}_{st}^h} [c(\varphi)] + \mathbb{E}_{\varphi \sim \mathcal{P}_{ts}^h} [c(\varphi)]$ <p>[97]</p> 	<p><i>Log-semiring</i></p> <p>Potential distance</p> $-\frac{1}{\mu} \left(\log \left(\mathbb{E}_{\varphi \sim \mathcal{P}_{st}^h} [e^{-\mu c(\varphi)}] \right) + \log \left(\mathbb{E}_{\varphi \sim \mathcal{P}_{ts}^h} [e^{-\mu c(\varphi)}] \right) \right)$ <p>[61, 95]</p> 	<p><i>Min-plus semiring</i></p> <p>Shortest path distance</p> $\min_{\varphi \in \mathcal{P}_{st}} c(\varphi)$ 
$(1, \infty)$	<p><i>Exp-norm bimonoid</i></p> <p>Exp-norm distance</p> $\mathbb{E}_{\varphi \sim \mathcal{P}_{st}^h} [\ c(\varphi)\ _r] + \mathbb{E}_{\varphi \sim \mathcal{P}_{ts}^h} [\ c(\varphi)\ _r]$ 	<p><i>Log-norm bimonoid</i></p> <p>Log-norm distance</p> $-\frac{1}{\mu} \left(\log \left(\mathbb{E}_{\varphi \sim \mathcal{P}_{st}^h} [e^{-\mu \ c(\varphi)\ _r}] \right) + \log \left(\mathbb{E}_{\varphi \sim \mathcal{P}_{ts}^h} [e^{-\mu \ c(\varphi)\ _r}] \right) \right)$ 	<p><i>Min-norm semiring</i></p> <p>Min-norm distance</p> $\min_{\varphi \in \mathcal{P}_{st}} \ c(\varphi)\ _r$ <p>[127]</p> 
∞	<p><i>Exp-max bimonoid</i></p> <p>Exp-max distance</p> $\mathbb{E}_{\varphi \sim \mathcal{P}_{st}^h} \left[\max_{e \in \varphi} c(e) \right] + \mathbb{E}_{\varphi \sim \mathcal{P}_{ts}^h} \left[\max_{e \in \varphi} c(e) \right]$ 	<p><i>Log-max bimonoid</i></p> <p>Log-max distance</p> $-\frac{1}{\mu} \left(\log \left(\mathbb{E}_{\varphi \sim \mathcal{P}_{st}^h} [e^{-\mu \max_{e \in \varphi} c(e)}] \right) + \log \left(\mathbb{E}_{\varphi \sim \mathcal{P}_{ts}^h} [e^{-\mu \max_{e \in \varphi} c(e)}] \right) \right)$ 	<p><i>Minimax semiring</i></p> <p>Minimax distance</p> $\min_{\varphi \in \mathcal{P}_{st}} \max_{e \in \varphi} c(e)$ <p>[122]</p> 

Table 4.1. Family of Log-Norm Distances. Family of log-norm distances and limit cases (**bold**) with their associated algebraic structure (*red*). The family of log-norm distances interpolates between the commute cost, shortest path and minimax distances. The algebraic structures associated to the cells in cyan ($r > 1$ and $0^+ < \mu < \infty$) do not form a semiring, but a strong bimonoid. The distances derived from these bimonoids are presented here for the first time. The graphs below represent the contribution of individual edges/paths to the s, t -distance in a toy problem. The cost of an edge is given by its length and wider edges have higher relevance. The r parameter regulates the impact of the edge costs in the paths, with higher r favoring paths with shorter edges. The parameter μ regulates the distribution of the contribution of the paths; higher μ favors the concentration of the distribution into a smaller number of lowest cost paths. Different distances differ radically in what part of a graph they emphasize

finally study under which circumstances the APP associated with a semiring defines a metric. In concrete, we focus on the setting where only hitting paths (paths whose last node appears only once) are considered. We find that one of the key factors lies in the function that maps elements of the semiring to the non-negative real numbers. Under natural conditions, we find that it is necessary, but not sufficient, that this function is subadditive with respect to the product operation of the semiring. We also provide sufficient conditions based on the monotonicity behavior of the function.

In summary we: 1) review the APP and how some of the most common graph metrics can be recovered by defining appropriate semirings, demonstrating this relation for the first time for some of them (Section 4.2); 2) introduce a novel unifying family of graph metrics, dubbed

log-norm distances, which interpolate between the CCD, minimax and shortest path metrics up to a constant factor (Section 4.3); 3) study under which conditions the APP associated with a semiring defines a graph metric (Section 4.4).

4.1.1 Related Work

There has been much work on interpolations between the shortest path distance and the CCD. In [186], these distances are interpolated by a family of dissimilarities inspired by the randomized shortest paths framework (RSP) [148]. These dissimilarities do not fulfill the triangle inequality, ergo they are not metrics. Also, based on the RSP, [61] and [95] define the same interpolating family of distances, which they call potential and free energy distances, respectively. We study this distance as an instance of the APP framework. The logarithmic forest distances [35] are a family of distances, based on the matrix forest theorem, which also interpolate between the SP and the CCD. The walk distances [36] are a broader set of distances which include the logarithmic forest, the CCD and the SP distances. The set of p -resistance distances [6] generalize the definition of the effective resistance distance [97], which is proportional to the CCD [32]. For $p = 1$, the SP distance is retrieved.

Closely related to our work, [92] define a family of similarity measures whose behavior resembles the one observed for the log-norm distances. Our metric adapts their similarities and transforms them into a proper distance while building a connection between the CCD, shortest path and minimax distances. As far as we know, the family of distances proposed in [71] is the only one that interpolates between the three aforementioned distances (CCD, minimax and shortest path distances) besides our proposal. Similar to the p -resistance approach, Gurvich generalizes the effective resistance concept by introducing two parameters. One difference between this proposal and ours lies in the fourth limit case that emerges from the approaches taken. While our limit computes the expected maximum edge cost of a path, its limit retrieves the inverse value of maximum flow between two nodes.

4.2 Preliminaries

4.2.1 Semirings

To fix notation and give necessary background, in this section we review the semiring algebraic structure, which constitutes the primary tool to understand the algebraic path problem. For a more extensive analysis of semirings, we refer the interested reader to [64].

Definition 6. A **semiring** is an algebraic structure $(S, \oplus, \otimes, \bar{0}, \bar{1})$ formed by a set S and two binary closed operations, \oplus and \otimes , with the following properties $\forall a, b, c \in S$:

Semiring	S	\oplus	\otimes	$\bar{0}$	$\bar{1}$	class
Min-plus	\mathbb{R}^+	min	+	∞	0	selective
Minimax	\mathbb{R}^+	min	max	∞	0	selective
Power set	$\mathcal{P}(A)$	\cup	\cap	\emptyset	A	idempotent

Table 4.2. Semiring Examples. $\mathcal{P}(A)$ refers to the power set of a set A .

- \oplus **commutativity**: $a \oplus b = b \oplus a$
- \oplus **associativity**: $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
- $\bar{0}$ **neutral element of \oplus** : $a \oplus \bar{0} = \bar{0} \oplus a = a$
- \otimes **associativity**: $(a \otimes b) \otimes c = a \otimes (b \otimes c)$
- $\bar{1}$ **neutral element of \otimes** : $a \otimes \bar{1} = \bar{1} \otimes a = a$
- **distributivity of \otimes relative to \oplus** :

$$(a \oplus b) \otimes c = a \otimes c \oplus b \otimes c, \quad c \otimes (a \oplus b) = c \otimes a \oplus c \otimes b$$

- $\bar{0}$ **absorbing for \otimes** : $a \otimes \bar{0} = \bar{0} \otimes a = \bar{0}$

Individually, \oplus and \otimes define a monoid over S . A semiring is idempotent if for all $a \in S$, $a \oplus a = a$. Furthermore, a semiring is called selective if $a \oplus b \in \{a, b\}$. If we drop the distributivity property from the list of requirements of a semiring, then the algebraic structure is called **strong bimonoid** [49]. Table 4.2 summarizes some common semirings.

A semiring has a canonical preorder relation¹, \preceq , given by

$$a \preceq b \iff \exists c \in S : a \oplus c = b. \quad (4.1)$$

As for the usual addition and product operations, we can extend the \oplus and \otimes to the matrix domain. Let A, B be two matrices in $S^{n \times m}$, then the following operations define a semiring over the matrices on S :

$$[A \oplus B]_{ij} = A_{ij} \oplus B_{ij}, \quad [A \otimes B]_{ij} = \bigoplus_k A_{ik} \otimes B_{kj}.$$

4.2.2 Graph Notation

Let $G = (V, E)$ be a directed graph where V and E represent the sets of vertices and edges respectively. A path $\wp = (v_0, \dots, v_k)$ from s to t is defined as a sequence of adjacent nodes, i.e. $(v_i, v_{i+1}) \in E$ with $v_0 = s$ and $v_k = t$. Note that a node can appear multiple times in a path. A hitting path is a path whose last node, t , appears only once. The set of all paths

¹Reflexive ($a \preceq a$) and transitive ($a \preceq b$ and $b \preceq c \rightarrow a \preceq c$) properties are satisfied, but antisymmetry ($a \preceq b$ and $b \preceq a \rightarrow a = b$) may not hold.

from s to t will be represented by \mathcal{P}_{st} . The subset of paths with exactly k edges is denoted by $\mathcal{P}_{st}[k]$. Analogously, we define the variant \mathcal{P}_{st}^h for the set of hitting paths.

Let $(S, \oplus, \otimes, \bar{0}, \bar{1})$ be a semiring. We say that a graph G is S -valued or S -graph if there is a cost function $c : V \times V \rightarrow S$ that assigns a cost $c(e) \in S$ to each edge. We set $c(e) = \bar{0}$ if and only if $e \notin E$. Additionally, the cost of a path is defined as the product of the cost of its edges, $c(\pi) = \bigotimes_{e \in \pi} c(e)$. For S -valued graphs, the entry A_{ij} of the adjacency matrix is equal to the cost of the edge (i, j) .

4.2.3 Algebraic Path Problem

Given a graph with $c(e) \in \mathbb{R}^+$, the shortest path problem (SPP) computes

$$\min_{\wp \in \mathcal{P}_{st}} \sum_{e \in \wp} c(e). \quad (4.2)$$

As previously mentioned, the min and $+$ operations characterize the min-plus semiring [143, 155]. The algebraic path problem (APP) extends the SPP through the use of general binary operations \oplus and \otimes that jointly form a semiring. On the one hand, the \otimes operation ($+$ in the SPP) acts over the cost of the edges. It can be interpreted as an edge concatenation operator which constructs the path by “multiplying” the cost of the edges. On the other hand, the \oplus operation (min in the SPP) acts over paths and behaves like a path aggregation operator which condenses the cost of different paths. When the semiring is selective (e.g. min-plus semiring), \oplus can also be interpreted as a choice operator where a single path is being selected. Formally, the APP generalizes (4.2) by calculating

$$\text{APP}(s, t) := \bigoplus_{\wp \in \mathcal{P}_{st}} \bigotimes_{e \in \wp} c(e) = \bigoplus_{\wp \in \mathcal{P}_{st}} c(\wp). \quad (4.3)$$

Let A be the adjacency matrix of the graph G . It can be verified that $[A^k]_{st} = \bigoplus_{\wp \in \mathcal{P}_{st}[k]} c(\wp)$. Since $\bigcup_k \mathcal{P}_{st}[k] = \mathcal{P}_{st}$ and $\mathcal{P}_{st}[k] \cap \mathcal{P}_{st}[k'] = \emptyset$ if $k \neq k'$, we obtain

$$\begin{aligned} \lim_{k \rightarrow \infty} [I \oplus A \oplus \dots \oplus A^k]_{st} &= \bigoplus_{k=0}^{\infty} \bigoplus_{\wp \in \mathcal{P}_{st}[k]} c(\wp) \\ &= \bigoplus_{\wp \in \mathcal{P}_{st}} c(\wp), \end{aligned} \quad (4.4)$$

where I is the diagonal matrix with $\bar{1}$'s in the diagonal. The limit in equation (4.4) is called the closure of A and it is denoted by A^* . Note that A^* , and consequently $\text{APP}(s, t)$, may not always exist. An interesting property of A^* is that it is the minimal solution of $X = A \otimes X \oplus I$ (see Proposition 6.2.2, Ch. 3 [64]). In the semirings considered in this paper, these limits will be well defined. Alternatively, the closure of an element a in a semiring may be defined as the solution of equation $x = a \otimes x \oplus \bar{1}$ instead of as a limit [103].

Specialized algorithms for the SPP have been generalized to the APP. The Dijkstra algorithm [47], which solves the Single Source Shortest Path Problem, has been generalized for some specific semirings [80, 133]. Analogously, the Floyd-Warshall algorithm [59], which solves the All Pairs of Shortest Paths Problem, can be generalized to solve the APP for any S -valued graph, for which A^* exists. This algorithm generalizes the Gauss–Jordan Method and solves $X = A \otimes X \oplus I$ [28].

4.2.4 Semiring Distances

If the edge costs of a graph are strictly positive, the shortest path distance defines a metric over the nodes of the graph. In this section, we will review other common graph metrics (minimax and CCD) and present how these can be expressed in terms of the APP.

Minimax Distance

An alternative popular distance in the literature is the minimax distance [31, 92, 122]. As its name indicates, this metric can be retrieved from the APP framework with the underlying minimax semiring (see Table 4.2)

$$\bigoplus_{\wp \in \mathcal{P}_{st}} \bigotimes_{e \in \wp} c(e) = \min_{\wp \in \mathcal{P}_{st}} \max_{e \in \wp} c(e). \quad (4.5)$$

This semiring calculates the so-called minimax path, i.e., the path which minimizes the most expensive edge of a path between two nodes. The minimax semiring can be used to calculate a minimum spanning tree (mST) since every simple path (path where each node appears once) between two nodes in a mST is a minimax path [122].

Commute Cost Distance

A prominent metric used for graphs is the commute cost distance (CCD) [60]. To compute the CCD between two nodes, it is necessary that each edge (i, j) has two values, p_{ij} and c_{ij} , associated to it. The first represents the probability that a random walker located at node i will transition from node i to node j through edge (i, j) . Usually, p_{ij} is set proportional to some weight w_{ij} that measures the affinity between the nodes i and j . The value c_{ij} is a positive value which indicates some kind of cost associated to the traversal of the edge (i, j) . The first hitting cost dissimilarity between two nodes s and t , $\mathcal{H}(s, t)$, is the expected cost that it takes a random walker starting at s to reach t for the first time. The CCD symmetrizes this dissimilarity. Formally,

$$\text{CCD}(s, t) = \underbrace{\mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} [c(\wp)]}_{\mathcal{H}(s,t)} + \underbrace{\mathbb{E}_{\wp \sim \mathcal{P}_{ts}^h} [c(\wp)]}_{\mathcal{H}(t,s)}. \quad (4.6)$$

The term $\mathcal{H}(s, t)$ can be expressed in the framework of the APP if one uses the so-called expectation semirings [11]. In concrete, the Eisner semiring [52], defined over the ground set $\mathbb{R}^+ \times \mathbb{R}^+$ with operations:

$$(a, b) \oplus (c, d) = (a + c, b + d), \quad (a, b) \otimes (c, d) = (ac, cb + ad),$$

recovers $\mathcal{H}(s, t)$. If we set the edge costs of the graph as $(p_e, p_e c_e)$, it can be verified that (see Appendix C.2)

$$\mathcal{H}(s, t) = \left[\bigoplus_{\varphi \in \mathcal{P}_{st}^h} \bigotimes_{e \in \varphi} (p_e, p_e c_e) \right]_2, \quad (4.7)$$

where the subindex 2 indicates the second entry. Therefore, CCD can be expressed in terms of APP. Note that (4.7) considers only hitting paths, in contrast to the general APP. The set \mathcal{P}_{st}^h over an arbitrary graph G is equal to the set \mathcal{P}_{st} over the graph $G^h[t]$, where the out-going edges of node t have been removed. Hence, t becomes an absorbing node and any path to t is therefore a hitting path. Consequently, CCD requires two APP on the graphs $G^h[t]$ and $G^h[s]$ to compute $\mathcal{H}(s, t)$ and $\mathcal{H}(t, s)$ respectively. The shortest path problem can also be constrained to hitting paths, since, as far as the costs are positive, each node appears only once in the optimal path.

Remark 4.2.1. It is known [95, appendix] that given some fixed random walker probabilities, p_e , CCD is proportional to the commute time distance (CTD), i.e. the expected length of a path (expected number of edges, that is $c_e = 1 \forall e \in E$). The c_e values only determine the proportionality constant between CCD and CTD.

4.3 Log-Norm Distances

In this section, we propose the novel family of log-norm distances, which interpolate between the above mentioned distances up to a constant factor. To do so, we introduce an interpolating family of distances between the minimax and shortest path. Moreover, we study the potential distance [61], which interpolates between the CCD and the shortest path distance. We prove that both metrics are instances of the APP framework once the appropriate semiring has been defined. Finally, we combine these semirings to define a greater family of strong bimonoids that will define the log-norm family of distances (Table 4.1).

4.3.1 Shortest Path and Minimax Distance Interpolation

The min-plus and min-max semiring can be continuously interpolated by the semiring $S = (R^+, \min, \otimes_r, \infty, 0)$ where $a \otimes_r b = \sqrt[r]{a^r + b^r}$. We are not aware that this algebraic structure has been acknowledged in the literature as a semiring, though it has been used in earlier

works [92]. Therefore, we dub it min-norm semiring. In Appendix C.1, we demonstrate that it is indeed a semiring. The APP derived from the min-norm semiring is defined as

$$\bigoplus_{\wp \in \mathcal{P}_{st}} \bigotimes_{e \in \wp} c(e) = \min_{\wp \in \mathcal{P}_{st}} \sqrt[r]{\sum_{e \in \wp} (c(e))^r}. \quad (4.8)$$

Clearly, when $r = 1$, \otimes_1 is equal to the regular sum, and consequently we recover the min-plus semiring. On the other extreme, when $r \rightarrow \infty$, \otimes_∞ is reduced to the max operation, hence the minimax semiring is retrieved. In appendix C.7.1, we show that (4.8) defines a metric over the nodes of a graph, which in turn also interpolates between the shortest path and minimax distances. This distance is also known in the literature as the power weighted shortest path metric [127].

The r parameter, which characterizes \otimes_r , regulates the impact of the edge costs in the paths. On the one hand, for high r , the min-norm path tends to have edges with low cost, though it may contain a higher number of edges (more similar to the minimax distance). On the other hand, for lower r , the min-norm path is more dominated by the total additive cost of the edges, which must be low overall (closer to the shortest path distance) but may contain edges whose cost are relatively high. Last column of Figure 4.1 illustrates this pattern, which was already pointed out in [92]. The shortest path ($r = 1$) contains only three, but one long edge in contrast to the minimax path ($r = \infty$) which contains many short edges. The min-norm path ($1 < r < \infty$) interpolates between both path patterns.

4.3.2 Commute Cost and Shortest Path Distance Interpolation

As mentioned in the introduction, CCD has some inconveniences if the graph is large [115]. Many node distances have been proposed that interpolate between the shortest path and the CCD distances in order to exploit the benefits of both metrics. Among them, we call attention to the potential distance (PD) [61, 95]. This distance is based on the randomized shortest paths (RSP) framework [148]. The PD can be interpreted as the logarithm of the expected reward $\exp(-\mu c(\wp))$ of the paths. Formally,

$$\text{PD}(s, t) = -\frac{1}{\mu} \log \left(\mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} [\exp(-\mu c(\wp))] \right) - \frac{1}{\mu} \log \left(\mathbb{E}_{\wp \sim \mathcal{P}_{ts}^h} [\exp(-\mu c(\wp))] \right), \quad (4.9)$$

where the parameter μ regulates implicitly the entropy of the distribution defined by the RSP framework. Kivimäki et al. showed that when $\mu \rightarrow 0$, PD tends to CCD, and when $\mu \rightarrow \infty$, it tends to the shortest path distance up to a constant factor (see Appendix C.5).

This distance also fits in the APP framework if one uses the log-semiring [109, 114] defined by

$$a \oplus_\mu b = -\frac{1}{\mu} \log (\exp(-\mu a) + \exp(-\mu b)), \quad a \otimes b = a + b.$$

Though the distance and the semiring were already known, we show the relation between them for the first time. Setting the edge costs of the graph equal to $-\log(p_e \exp(-\mu c_e))/\mu$, it can be verified that (see Appendix C.3)

$$\text{PD}(s, t) = \bigoplus_{\varnothing \in \mathcal{P}_{st}^h} \bigotimes_{e \in \varnothing} \mu - \frac{1}{\mu} \log(p_e \exp(-\mu c_e)) + \bigoplus_{\varnothing \in \mathcal{P}_{ts}^h} \bigotimes_{e \in \varnothing} \mu - \frac{1}{\mu} \log(p_e \exp(-\mu c_e)) \quad (4.10)$$

Remark 4.3.1. In [61] it was noted that the PD could also be computed by applying a generalization of the Bellman-Ford formula [16]. This is a direct consequence of the APP framework, since the PD can be retrieved by the log-semiring.

Remark 4.3.2. It is worth to mention that although the PD interpolates between the shortest path and CCD distances, the log-semiring does not interpolate between the min-plus and the Eisner semiring. When $\mu \rightarrow \infty$, \oplus_∞ becomes the min operator and

$$\lim_{\mu \rightarrow \infty} -\log(p_e \exp(-\mu c_e))/\mu = c_e.$$

Thus, the min-plus semiring arises. Nonetheless, when $\mu \rightarrow 0^+$, \oplus_{0^+} and the costs themselves are not well defined.

In contrast to the min-norm semirings, the log-semiring is not selective, i.e. it does not make a choice over the paths, but aggregates their costs. The operation \oplus_μ weighs all the paths by their probability and cost. When μ approaches 0^+ , the metric considers all paths primarily based on their probability. In such cases, a greater number of high-probability paths between two nodes result in a shorter distance, with the costs having a negligible effect as per Remark 4.2.1. This intuitively implies that if there are more connections between s and t , the random walker is more likely to be absorbed earlier. Conversely, as μ increases, lower-cost paths are favored due to the RSP framework. Consequently, paths with lower costs become more relevant. In the extreme case of $\mu \rightarrow \infty$, only the shortest paths are considered. See first row of Figure 4.1.

4.3.3 The Family of Log-Norm Distances

In the previous sections we have shown how popular node metrics can be posed as particular instances of the APP. In concrete, we presented a semiring that interpolates between the shortest path and minimax distances via the parameter r , which conditions the \otimes_r operation. Additionally, we have discussed the log-semiring, whose APP interpolates between the CCD and the shortest path distances via a parameter μ that determines the \oplus_μ operation. A natural question arises: *is there any semiring that defines a distance interpolating between the CCD and the minimax distance?* In this section we aim to answer this question by proposing a

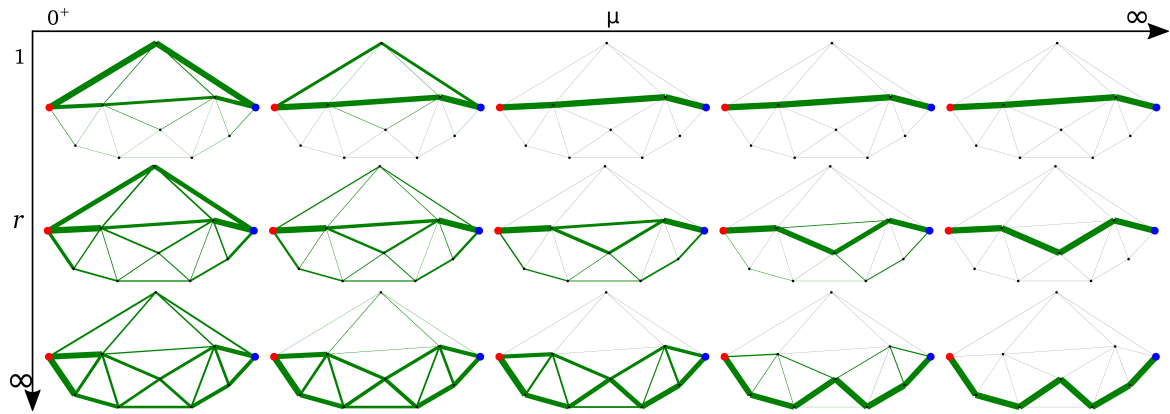


Figure 4.1. Schematic Path Relevance in the Log-Norm Distance for Different Values of μ and r in a Graph. We computed the 1000 shortest paths according to the costs $\|c(\wp)\|_r$ for different r and μ values. The edge width is proportional to $\sum_{\wp: e \in \wp} \Pr(\wp) \exp(-\mu \|c(\wp)\|_r)$. It depicts how much each edge contributes to the value of the APP: the wider the edge, the more significant. The random walker probabilities are uniform at each node. The parameter μ regulates how the importance of the paths is distributed conditioned by their cost, while r regulates the cost of the paths.

family of strong bimonoids, whose associated APP interpolates between those distances. The key operators that allow to interpolate between the distances are \oplus_μ and \otimes_r . To relate all above mentioned distances, we propose to define an algebraic structure that combines the Eisner, log- and min-norm semirings. We define the following operations over $\mathbb{R}^+ \times \mathbb{R}^+$

$$\begin{aligned} (a, b) \oplus_\mu (c, d) &= \left(1, -\frac{1}{\mu} \log \left(a e^{-\mu b} + c e^{-\mu d} \right) \right), \\ (a, b) \otimes_r (c, d) &= \left(ac, \sqrt[r]{b^r + d^r} \right). \end{aligned} \quad (4.11)$$

Unfortunately, the distributive property of \oplus_μ with respect to \otimes_r does not hold for arbitrary r and μ values. Consequently, these operations do not define a semiring, but a strong bimonoid. Nonetheless, its APP, with edge costs equal to (p_e, c_e) as defined in Section 4.2.4, still defines a distance, which we name the log-norm distance (LN):

$$\begin{aligned} \text{LN}(s, t) &= \left[\bigoplus_{\wp \in \mathcal{P}_{st}^h} \bigotimes_{e \in \wp} (p_e, c_e) + \bigoplus_{\wp \in \mathcal{P}_{ts}^h} \bigotimes_{e \in \wp} (p_e, c_e) \right]_2 \\ &= -\frac{1}{\mu} \log \left(\mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} [\exp(-\mu \|c(\wp)\|_r)] \right) - \frac{1}{\mu} \log \left(\mathbb{E}_{\wp \sim \mathcal{P}_{ts}^h} [\exp(-\mu \|c(\wp)\|_r)] \right), \end{aligned}$$

where $\|c(\wp)\|_r = \sqrt[r]{\sum_{e \in \wp} (c_e)^r}$. Table 4.1 summarizes all the graph metrics that the log-norm family includes along with the algebraic structure, whose APP defines the metric. Note that the algebraic structures do not interpolate between them for the same reason that was explained in remark 4.3.2.

The log-norm distances are closely related with the similarities presented in [92], defined as $\sum_{\varphi \in \mathcal{P}_{ts}^h} \exp(-\mu \|c(\varphi)\|_r)$. These similarities are not defined when $\mu \rightarrow 0^+$. In our setting, we transform these similarities into a metric by computing the expected similarity between two nodes followed by the $-\log(\cdot)/\mu$ function. This way, we are able to study the limit $\mu \rightarrow 0^+$, which retrieves the CCD when $r = 1$.

The log-norm family can also be related with the Helmholtz free energy, following the same reasoning that related the PD with the free energy in [95]. We prove in Appendix C.9, that the log-norm distance between two nodes s and t is equal to $\Phi_r(\text{Pr}_{st}) + \Phi_r(\text{Pr}_{ts})$, where $\Phi_r(\text{Pr}_{st}) = \sum_{\varphi \in \mathcal{P}_{st}^h} \text{Pr}_{st}(\varphi) \|c(\varphi)\|_r + \frac{1}{\mu} \text{KL}(\text{Pr}_{st}, \text{Pr}^{\text{ref}})$ is the free energy of a thermodynamical system and Pr_{st} is a probability distribution over the hitting paths between s and t that minimizes the free energy at a certain temperature $1/\mu$. KL denotes the KL-divergence, and Pr^{ref} denotes a reference probability distribution over the hitting paths given by the random walker.

In Figure 4.1, we schematically plot the path relevance determined by the log-norm metric for different r and μ values. We computed the 1000 shortest paths according to the costs $\|c(\varphi)\|_r$ for varying r values. The width of each edge, visually representing the contribution of each edge to the value of the APP, is proportional to $\sum_{\varphi: e \in \varphi} \text{Pr}(\varphi) \exp(-\mu \|c(\varphi)\|_r)$. Wider edges indicate greater significance. The random walker probabilities are set uniform at each node. We observe that for lower μ , the influence among the edges of different paths is more evenly distributed. As a consequence of Remark 4.2.1, for the CCD ($r = 1, \mu \rightarrow 0^+$), the cost of a path does not determine its influence. Consequently, the more influential paths are the ones with higher probability. In our example, the high probability paths coincide with the shorter length paths (low number of edges) because we assume a uniform transition probability at each node. For r and μ values close to 1 and 0 respectively, we expect similar behaviour. Indeed, this pattern is evident in the top-left corner graph. However, as the value of r increases, the influence shifts to the paths located in the lower part of the graph, which have a lower cost with respect to $\|\cdot\|_r$. This shift suggests that the cost factor becomes more dominant. Nonetheless, the distribution of mass remains spread across all edges of all paths, and the probabilities of the paths continue to hold significance.

Contrarily, for higher μ , the contribution of the paths shifts from paths with higher probability to paths with lower cost $\|c(\varphi)\|_r$. In the limit as μ tends to infinity, only the path with the minimum cost is considered. This convergence is more pronounced for lower values of r , as $\text{Pr}(\varphi) \exp(-\mu \|c(\varphi)\|_r) \leq \text{Pr}(\varphi) \exp(-\mu \|c(\varphi)\|_{r'})$ for $r' \leq r$. Thus, for higher r values the relevance of the factor $\text{Pr}(\varphi)$ in $\text{Pr}(\varphi) \exp(-\mu \|c(\varphi)\|_r)$ decreases at a slower rate, with respect to the increase of μ , than it does for lower r values. For higher r , paths which contain low cost edges are favored. In summary, both r and μ exhibit similar effects as observed for

the min-norm and potential distances but can be combined within our framework. It's worth noting that Figure 4.1 provides an approximation, considering only a finite number of paths. This approximation is more reliable for higher μ values, where the influence of high-cost paths is negligible.

In Appendix C.8, we show that LN defines a distance. Though PD and LN are similar in form, the same strategy that was followed in [61] to prove that PD is a distance does not apply for LN due to the absence of distributivity. Instead, we expose an step by step derivation to show the triangle inequality. The proof builds on the factorization of the set of paths from s to t into those that cross a third node q and those which do not. Furthermore, the absence of distributivity makes any attempt to compute the LN distance in finite time impractical, since one can not factor out common terms for different paths. General algorithms proposed for the APP can not be applied here either. Though there have been papers that have tackled the non-distributivity question, they are not applicable here. For instance, in [44] only selective semirings are considered. Alternatively, the proposed approach in [104] does not simplify the computation of our algebraic structure. Nonetheless, in Appendix C.10 we sketch an algorithm to compute the Exp-max and Log-max distances (last row Table 4.1). The analysis and implementation of this algorithm is out of the scope of the current thesis and, therefore, is left for future work.

4.4 When Does a Semiring Define a Distance?

In the previous sections, we have expressed some of the most common graph metrics in terms of the APP. Now, we wonder which properties must a semiring satisfy such that its associated APP defines a metric. Since all the metrics we have analyzed could be expressed in terms of the APP framework which considers only hitting paths, we will focus on the hitting case, denoted here by APP^h . We hope that our results will allow researchers to more easily define semirings that yield new, potentially useful graph distances and reveal some of the underlying structure of semiring based graph distances.

A metric maps a pair of points to a non-negative real number. Since a semiring can be defined over an arbitrary set, we need a function $\mathfrak{g}: S \mapsto \mathbb{R}^+ \cup \{\infty\}$ that maps an element of the semiring to the non-negative real numbers. We assume $\mathfrak{g}(\bar{1}) = 0$ and $\mathfrak{g}(\bar{0}) = \infty$, since $APP^h(s, s) = \bar{1}$ and $\bar{0}$ represents the cost of the non-existing edges/paths. Let $G = (V, E)$ be a S -graph. We assume that the graph is strongly connected, such that there exists a path connecting any two arbitrary nodes and that $APP^h(\cdot, \cdot)$ is defined for each pair of nodes. Given $s, t \in V$ we define the following dissimilarity function

$$d(s, t) = \mathfrak{g}(APP^h(s, t)) + \mathfrak{g}(APP^h(t, s)). \quad (4.12)$$

We aim to find out which properties \mathfrak{g} and the semiring S must satisfy such that (4.12) determines a metric function. Our proofs will focus only on the left summand of (4.12), since for the right term the same properties will follow:

$$d_L(s, t) := \mathfrak{g}(\text{APP}^h(s, t)) = \mathfrak{g} \left(\bigoplus_{\wp \in \mathcal{P}_{st}^h} \bigotimes_{e \in \wp} c(e) \right). \quad (4.13)$$

Through the whole section we assume that distributivity commutativity and associativity of the semiring operations also hold for infinite sums and products.

4.4.1 Identity of Indiscernibles

The non-negativity of d follows trivially from the definition. We have $d(s, s) = 0$, since $\text{APP}^h(s, s) = \bar{1}$ and $\mathfrak{g}(\bar{1}) = 0$. The opposite direction of the identity of indiscernibles property is more delicate. The next lemma states some conditions that ensure $s = t$ if $d(s, t) = 0$. Concretely, it requires $\bar{1}$ to be the unique element mapped to 0 and that all paths have cost greater than $\bar{1}$.

Lemma 4.4.1. Let d be defined as in (4.12). If

1. $a \preceq \bar{1} \iff a = \bar{1}$ or $a = \bar{0}$, where \preceq is the canonical preorder relation defined in (4.1),
2. $\mathfrak{g}(a) = 0$ if and only if $a = \bar{1}$,
3. none of the edge costs is invertible with respect to \otimes ,

then $d(s, t) = 0$ if and only if $s = t$.

Proof: According to assumption 2, we just need to prove that $\text{APP}^h(s, t) \neq \bar{1}$ for arbitrary distinct vertices s and t . First we recall the definition of the canonical order of a semiring which was stated in equation (4.1):

$$a \preceq b \iff \exists c \in S \text{ such that } a \oplus c = b.$$

As a consequence of the definition of \preceq and the first assumption, there do not exist any a and b distinct of $\bar{1}$ such that $a \oplus b = \bar{1}$. Therefore,

$$\text{APP}^h(s, t) = \bigoplus_{\wp \in \mathcal{P}_{st}^h} \bigotimes_{e \in \wp} c(e) = \bar{1} \implies \exists \wp \in \mathcal{P}_{st}^h \text{ such that } \bigotimes_{e \in \wp} c(e) = \bar{1}.$$

Thanks to assumption 3, $\bigotimes_{e \in \wp} c(e) \neq \bar{1}$, otherwise the costs $c(e)$ would have inverse elements. □

4.4.2 Triangle Inequality

To prove when the triangle inequality holds, we will show first that a necessary condition, but not sufficient, is the subadditivity of \mathfrak{g} with respect to \otimes . Subsequently, we will show sufficient conditions for it to hold. We will assume that the edge costs, and also $\text{APP}^h(\cdot, \cdot)$, can take arbitrary values in the semiring S .

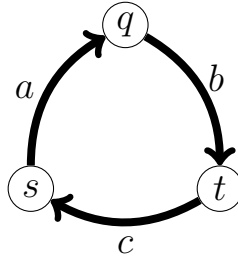


Figure 4.2. Necessity of the Subadditivity of \mathfrak{g} for the Triangle Inequality to Hold.

Graph where all paths connecting s and t pass through q . The terms $a, b \in S$ indicate the cost of the corresponding edges. The triangle inequality can only be satisfied if \mathfrak{g} is subadditive such that $d_L(s, t) = \mathfrak{g}(\text{APP}^h(s, q) \otimes \text{APP}^h(q, t)) = \mathfrak{g}(a \otimes b) \leq \mathfrak{g}(a) + \mathfrak{g}(b) = \mathfrak{g}(\text{APP}^h(s, q)) + \mathfrak{g}(\text{APP}^h(q, t)) = d(s, q) + d(q, t)$.

Necessity of the subadditivity of \mathfrak{g}

Our arguments are grounded in the observation that the paths between s and t can be categorized into those that traverse a node q and those that do not. This partitioning leads to the following result, the proof of which is provided in Appendix C.6.1.

Lemma 4.4.2. Given a graph G and arbitrary nodes s, t and q then

$$\text{APP}^h(s, t) = \alpha^h \oplus \beta^h \otimes \text{APP}^h(q, t), \quad \text{APP}^h(s, q) = \beta^h \oplus \alpha^h \otimes \text{APP}^h(t, q), \quad (4.14)$$

where

$$\alpha^h := \bigoplus_{\substack{\wp \in \mathcal{P}_{st}^h \\ q \notin \wp}} \bigotimes_{e \in \wp} c(e), \quad \beta^h := \bigoplus_{\substack{\wp_1 \in \mathcal{P}_{sq}^h \\ t \notin \wp_1}} \bigotimes_{e \in \wp_1} c(e).$$

The first equality of (4.14) decomposes the cost of the paths from s to t in two terms: one that depends on the paths that pass through a third node q and a second term where the paths do not pass through q . Indeed, the term α^h aggregates all hitting paths which do not cross node q , while the term $\beta^h \otimes \text{APP}^h(q, t)$ considers all hitting paths that pass through q . The second equality, performs the same decomposition as the first equality but considering the paths from s to q .

Let us prove now that \mathfrak{g} must be subadditive, if the triangle inequality holds for the function d_L , (4.13). It follows from equation (4.14) that

$$\begin{aligned} \mathfrak{g}(\alpha^h \oplus \beta^h \otimes \text{APP}^h(q, t)) &= \mathfrak{g}(\text{APP}^h(s, t)) = d_L^h(s, t) \\ &\leq d_L^h(s, q) + d_L^h(q, t) = \mathfrak{g}(\text{APP}^h(s, q)) + \mathfrak{g}(\text{APP}^h(q, t)). \end{aligned} \quad (4.15)$$

For a graph where all paths from s to t cross node q (e.g. Figure 4.2) we have $\alpha^h = 0$ and

therefore $\beta^h = \text{APP}^h(s, q)$. Hence

$$\begin{aligned} \mathfrak{g}(\text{APP}^h(s, t)) &= \mathfrak{g}(\text{APP}^h(s, q) \otimes \text{APP}^h(q, t)) = d_L(s, t) \\ &\leq d_L(s, q) + d_L(q, t) = \mathfrak{g}(\text{APP}^h(s, q)) + \mathfrak{g}(\text{APP}^h(q, t)). \end{aligned} \quad (4.16)$$

Note that in the graph of Figure 4.2 the cost of the edge from s to q , a , is equal to $\text{APP}^h(s, q)$. Analogously, $\text{APP}^h(q, t) = b$. Thus, for arbitrary values $a, b \in S$ the subadditivity of \mathfrak{g} in S follows:

$$\mathfrak{g}(a \otimes b) \leq \mathfrak{g}(a) + \mathfrak{g}(b).$$

We have proven that exists a graph where \mathfrak{g} must be subadditive to ensure that d_L satisfies the triangle inequality on the graph in Figure 4.2. Thus, the next theorem holds.

Theorem 4.4.3. If the function d_L , satisfies the triangle inequality over an arbitrary graph and $\text{APP}^h(\cdot, \cdot)$ can take any value in the semiring S , then \mathfrak{g} is \otimes -subadditive, i.e.,

$$\mathfrak{g}(a \otimes b) \leq \mathfrak{g}(a) + \mathfrak{g}(b), \quad \forall a, b \in S. \quad (4.17)$$

Remark 4.4.4. Note that $\text{APP}^h(s, q)$ and $\text{APP}^h(q, t)$ may not take any possible value in the semiring. For instance, in the Eisner semiring, which characterizes the commute cost distance, the possible values of these variables lie in the set $\{1\} \times \mathbb{R}^+$. In the Eisner semiring, the first entry of $\text{APP}(s, t)$ is always equal to $\sum_{\varphi \in \mathcal{P}_{st}^h} \Pr(\varphi) = 1$, since it is the sum of the probabilities of the hitting paths from s to t .² In this concrete case, the subadditivity should be constrained to this set.

Sufficient conditions

Now we will present some conditions that are sufficient to ensure that (4.12) satisfies the triangle inequality over an arbitrary graph. In particular, we analyze the triangle inequality when \mathfrak{g} is monotone with respect to \preceq .

Theorem 4.4.5. Let $G = (V, E)$ be an S -graph. If

1. \mathfrak{g} is \otimes -subadditive, i.e., $\mathfrak{g}(a \otimes b) \leq \mathfrak{g}(a) + \mathfrak{g}(b) \quad \forall a, b \in S$,
2. \mathfrak{g} is decreasing, i.e., $a \preceq b \rightarrow \mathfrak{g}(b) \leq \mathfrak{g}(a) \quad \forall a, b \in S$,
3. $a \otimes \text{APP}^h(t, q) \otimes \text{APP}^h(q, t) \preceq a, \quad \forall a \in S, q, t \in V$.

then d satisfies the triangle inequality over the nodes of G .

The third assumption states that, if $c(\varphi) = a \in S$ is the cost of a path, φ , and this path is concatenated (\otimes operation) with all cycles starting at a node t and traversing an arbitrary node q , then the aggregation of all these new costs is not greater (according to (4.1)) than the original cost of the path $c(\varphi)$. Consequently, since \mathfrak{g} is decreasing, the concatenation

²Appendix A of [61] proves that the sum of the path likelihoods is equal to 1 for hitting paths.

(and aggregation) of the cycles does not increase the \mathfrak{g} -value of the path. The proof of Theorem 4.4.5 focuses on demonstrating that

$$\text{APP}^h(s, q) \otimes \text{APP}^h(q, t) \preceq \alpha^h \oplus \beta^h \otimes \text{APP}^h(q, t).$$

If β^h was equal to $\text{APP}^h(s, q)$, requiring \mathfrak{g} to be decreasing and subadditive would suffice to prove the triangle inequality. Since $\beta^h \preceq \text{APP}^h(s, q)$, assumption 3 is needed to ensure the triangle inequality.

Proof: We need to prove that the triangle inequality (4.15) holds. Let s, q and t be arbitrary nodes of G . Due to the subadditivity of \mathfrak{g} , we have

$$\mathfrak{g}(\text{APP}^h(s, q) \otimes \text{APP}^h(q, t)) \leq \mathfrak{g}(\text{APP}^h(s, q)) + \mathfrak{g}(\text{APP}^h(q, t)).$$

Therefore, as a consequence of the first equality of equation (4.14), it will be enough to show

$$\begin{aligned} d_L(s, t) &= \mathfrak{g}(\text{APP}^h(s, t)) = \mathfrak{g}(\alpha^h \oplus \beta^h \otimes \text{APP}^h(q, t)) \\ &\leq \mathfrak{g}(\text{APP}^h(s, q) \otimes \text{APP}^h(q, t)) = d_L(s, q) + d_L(q, t), \end{aligned}$$

which will follow from

$$\text{APP}^h(s, q) \otimes \text{APP}^h(q, t) \preceq \alpha^h \oplus \beta^h \otimes \text{APP}^h(q, t), \quad (4.18)$$

since \mathfrak{g} is decreasing. From the second equality of equation (4.14), it suffices to prove the following inequality

$$\begin{aligned} \text{APP}^h(s, q) \otimes \text{APP}^h(q, t) &= (\alpha^h \otimes \text{APP}^h(t, q) \oplus \beta^h) \otimes \text{APP}^h(q, t) \\ &= \alpha^h \otimes \text{APP}^h(t, q) \otimes \text{APP}^h(q, t) \oplus \beta^h \otimes \text{APP}^h(q, t) \quad (4.19) \\ &\preceq \alpha^h \oplus \beta^h \otimes \text{APP}^h(q, t), \end{aligned}$$

which holds if

$$\alpha^h \otimes \text{APP}^h(t, q) \otimes \text{APP}^h(q, t) \preceq \alpha^h. \quad (4.20)$$

Indeed, (4.20) holds thanks to our third assumption. \square

One can derive as special cases of Theorem 4.4.5 that the min-norm distances (including the shortest path and minimax distances) and the PD define metrics over arbitrary graphs (see Corollary C.7.1 and Corollary C.7.2). However, this theorem does not apply to CCD. The next theorem mirrors Theorem 4.4.5, but considers \mathfrak{g} to be increasing instead of decreasing. We defer the proof to the Appendix C.6.1 due to its similarity with Theorem 4.4.5. As a corollary, we can prove the fact that the CCD defines a metric (see Corollary C.7.3).

Theorem 4.4.6. Let $G = (V, E)$ be an S -graph. If

1. \mathfrak{g} is \otimes -subadditive, i.e., $\mathfrak{g}(a \otimes b) \leq \mathfrak{g}(a) + \mathfrak{g}(b) \forall a, b \in S$,
2. \mathfrak{g} is increasing in $S \setminus \{\bar{0}\}$, i.e., $a \preceq b \rightarrow \mathfrak{g}(a) \leq \mathfrak{g}(b) \forall a, b \in S \setminus \bar{0}$,

$$3. a \preceq a \otimes \text{APP}^h(t, q) \otimes \text{APP}^h(q, t), \forall a \in S, q, t \in V.$$

then d satisfies the triangle inequality over the nodes of G .

In Appendix C.7 we provide, in addition to simplified proofs of the validity of existing metrics, a case example of a new metric, which can be easily verified to be a metric thanks to our theorems. Note that, none of the results stated in this section apply to the log-norm distance, since it is not defined by a semiring but a strong bimonoid. Distributivity is an essential part of the proofs.

4.5 Conclusion

In this chapter, we have revisited some of the most common graph metrics (shortest path, CCD and minimax distances) and have presented them in terms of the algebraic path problem. We reviewed semirings whose associated algebraic path problems retrieve these metrics. We also discussed the potential distance (which interpolates between the CCD and the shortest path distance) and the min-norm distance (which interpolates between the shortest path and minimax distances) from a new perspective. We showed that these metrics can be expressed as instances of the APP framework.

Moreover, we have proposed a novel unifying family of metrics which includes and relates all the aforementioned distances. This family of metrics is parameterized by a parameter r that regulates the impact of edges with high cost, and another parameter μ which regulates the influence of the paths based on their $\|\cdot\|_r$ -cost. Moreover, inspired by [95], we have proven that the log-norm distance between two nodes coincides with the symmetrized minimum Helmholtz free energy between the nodes. Unfortunately, this distance cannot be obtained as the APP of a semiring but of a strong bimonoid, and its exact computation remains infeasible. An efficient approximate computation is left for future work.

Finally, we have provided sufficient conditions which ensure that the APP constrained to hitting paths associated with a semiring, S , defines a metric over the nodes of a graph. In addition, we showed that the function g that maps elements from S to \mathbb{R}^+ must be \otimes -subadditive if $g(\text{APP}^h(\cdot, \cdot))$ is to satisfy the triangle inequality. We hope that these results can help in the design of new metrics, and as such help enrich the toolbox available to graph-centric machine learning.

Chapter 5

Central Spanning Tree

Spanning trees are an important primitive in many data analysis tasks, when a data set needs to be summarized in terms of its “skeleton”, or when a tree-shaped graph over all observations is required for downstream processing. Popular definitions of spanning trees include the minimum spanning tree and the optimum distance spanning tree, a.k.a. the minimum routing cost tree. When searching for the shortest spanning tree but admitting additional branching points, even shorter spanning trees can be realized: Steiner trees. Unfortunately, both minimum spanning and Steiner trees are not robust with respect to noise in the observations; that is, small perturbations of the original data set often lead to drastic changes in the associated spanning trees. In response, we make two contributions when the data lies in a Euclidean space: on the theoretical side, we introduce a new optimization problem, the “(branched) central spanning tree”, which subsumes all previously mentioned definitions as special cases. On the practical side, we show empirically that the (branched) central spanning tree is more robust to noise in the data, and as such is better suited to summarize a data set in terms of its skeleton. We also propose a heuristic to address the NP-hard optimization problem, and illustrate its use on single cell RNA expression data from biology.

5.1 Introduction

Many data analysis tasks call for the summary of a data set in terms of a spanning tree, or use tree representations for downstream processing. Examples include the inference of trajectories in developmental biology [38, 147], generative modeling in chemistry [2], network design [177] or skeletonization in image analysis [10, 170]. The problem is akin to, but more complicated than, the estimation of principal curves because good recent methods such as [117] cannot account for branched topologies. For a spanning tree representation to be meaningful, it is of paramount importance that the tree structure be robust to minor perturbations of the data, e.g. by measurement noise. In this work, we address the geometric

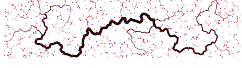
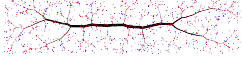
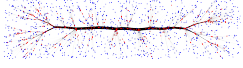
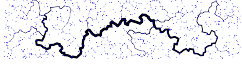
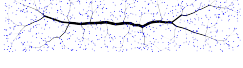
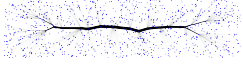
		Edge Centrality Exponent α		
		0	(0, 1)	1
SP	α			
Add Steiner points	YES	Steiner Tree ^[63] $\min_{X_B, T} \sum_{(i,j) \in E_T} \ x_i - x_j\ $ 	Branched Central Spanning Tree $\min_{X_B, T} \sum_{(i,j) \in E_T} (m_{ij}(1 - m_{ij}))^\alpha \ x_i - x_j\ $ 	Branched Minimum Routing Cost Tree $\min_{X_B, T} \sum_{i,j \in V \times V} d_T(i, j)$ 
	NO	Minimum Spanning Tree ^[99] $\min_T \sum_{(i,j) \in E_T} \ x_i - x_j\ $ 	Central Spanning Tree $\min_T \sum_{i,j \in E_T} (m_{ij}(1 - m_{ij}))^\alpha \ x_i - x_j\ $ 	Minimum Routing Cost Tree ^[125] $\min_T \sum_{i,j \in V \times V} d_T(i, j)$ 

Figure 5.1. Euclidean Central Spanning Tree Family of Problems with and without Steiner Points. The central spanning tree weighs the costs of the edges, given by node distances, with the centrality of the edges, $m_{ij}(1 - m_{ij})$. The influence of the centrality is regulated by the parameter $\alpha \in [0, 1]$. For lower α values the centrality becomes insignificant, and the tree tends to contain short edges. For higher α values, the tree encourages central edges of low cost, at the expense of peripheral edges of higher cost. We study central spanning trees with and without additional Steiner points (shown in red). The widths of all edges are proportional to their centrality. The central spanning tree problems includes well-known and novel spanning tree problems, the latter highlighted in yellow.

stability of spanning trees over points lying in an Euclidean space.

The minimum spanning tree (mST) is surely the most popular spanning tree, owing to its conceptual simplicity and ease of computation. For a graph $G = (V, E)$ with edge costs, the mST is a tree that spans G while minimizing the total sum of edge costs. It prioritizes shorter edges that connect closely located nodes enhancing data faithfulness. Unfortunately, its greedy nature makes the mST susceptible to small data perturbations that may lead to drastic changes in its structure, see Figure 5.4. An alternative, the minimum routing cost tree (MRCT), minimizes the sum of pairwise shortest path distances [125]. Unlike the mST, solving the MRCT is NP-hard [179]. Despite this, the MRCT exhibits a more stable geometric structure compared to the mST, as it tends to be more "star-shaped" (see Figure 5.4). Nevertheless, this star-shaped tendency inclines towards connecting nodes that are spatially distant, introducing a risk of information loss and compromised data fidelity. This effect becomes particularly pronounced in high-dimensional spaces, potentially rendering

the MRCT approach unusable (see Figures 5.5i-5.5l)). Achieving a balance between data fidelity and geometric robustness is crucial for an effective spanning tree.

Central spanning trees (CST) In this paper, we propose a novel parameterized family of spanning trees that interpolate and generalize all the aforementioned ones. Unless otherwise stated, we will assume a complete Euclidean graph, where the nodes are embedded in the Euclidean space with coordinates $X_V = \{x_1, \dots, x_{|V|}\} \subset \mathbb{R}^n$ and the edge costs are given by the distances between the vertices, $c_{ij} = \|x_i - x_j\|$. We define the CST as the spanning tree of G that minimizes the objective

$$\text{CST} := \arg \min_{\text{T}} \sum_{e \in E_{\text{T}}} (m_e(1 - m_e))^{\alpha} c_e = \arg \min_{\text{T}} \sum_{(i,j) \in E_{\text{T}}} (m_{ij}(1 - m_{ij}))^{\alpha} \|x_i - x_j\|, \quad (5.1)$$

where m_e and $(1 - m_e)$ are the normalized cardinalities of the components resulting from the removal of the edge e from T . Thus, $m_e(1 - m_e)$ is the product of the number of nodes on both sides of the edge divided by $|V|^2$. Because this value is proportional to the ‘‘edge betweenness centrality’’¹ of e in T [23], we call the problem the Central Spanning Tree (CST) problem. The exponent α is the interpolating parameter that modulates the effect of the edge centrality. For α close to 0, the centrality becomes insignificant, so the tree tends to contain lower cost edges overall. For $\alpha = 0$ we retrieve the mST. On the other hand, as α increases, the centrality becomes more relevant, leading the tree to favor topologies with low centrality edges, thus promoting a higher branching effect. For $\alpha = 1$, the resulting expression is proportional to the MRCT. Here, each edge cost is multiplied by the number of shortest paths it belongs to, leading the total sum to represent the sum of shortest path distances (see Appendix D.3).

As will be seen in Section 5.3, the α parameter has an effect on the geometric stability of the spanning tree, with higher α resulting in greater robustness. The robustness of spanning trees has been explored in various contexts. For instance, researchers have investigated the robustness of the mST cost robustness under edge weight uncertainty [90, 154] or studying robustness against node or edge failure in networks [110]. Additionally, studies have delved into the stability regions of mST, under which any change in vertex location does not alter the mST [134, 139]. The central tree problem [20], related by name to ours, focuses on computing a tree that minimizes the maximal distance to a set of given trees. To our knowledge, we are the first to propose a spanning tree whose geometric structure is stable and resilient to data perturbations such as noise.

Finally, we remark the connection between the the CST problem and the Minimum Concave Cost Network Flow (MCCNF) problem [69, 188]. The MCCNF problem aims to minimize the cost of distributing a certain commodity from source to sink nodes. Such a problem

¹The edge betweenness centrality measures an edge’s frequency in shortest paths between nodes, with more traversed edges being deemed more central. In trees, it’s the product of nodes on opposite sides of the edge.

models the cost of an edge as a concave function of the transportation flow. The CST can be reinterpreted as MCCNF where a commodity with mass equal to $|V| - 1/|V|$, concentrated into a single source node, must be transported to the rest of nodes. In our case, the term m_e in (5.1) can be interpreted as the flow of such problem. Since the function $(m_e(1 - m_e))^\alpha c_e$ is concave with respect to m_e for $\alpha \in]0, 1]$, we deduce that the CST is an instance of the MCCNF. A more detailed discussion of the interpretation of the CST as a MCCNF problem is offered in Appendix D.2.

Considering the CST from the perspective of an MCCNF problem, it becomes clear that it falls into the NP-hard category. Indeed, the authors of [70] showed that single-source MCCNF problems with strictly concave functions are NP-hard. Consequently, we conclude that the CST problem is NP-hard for $\alpha \in]0, 1]$ due to the strictly concave nature of the edge cost function $(m_e(1 - m_e))^\alpha c_e$.²

Branched central spanning trees (BCST) Inspired by [108], we also study the variant of the CST problem which allows for the introduction of additional nodes, known in the literature as branching points or Steiner points (SPs). Formally, we distinguish between two types of nodes. On the one hand, we have the terminal nodes with fixed coordinates given by X_V . On the other hand, we allow for an extra set of points, B , whose coordinates X_B must be jointly optimized with the topology T . In this case, T is a spanning tree defined over the nodes $V \cup B$. Accordingly, the objective of the CST problem becomes

$$\min_{T, X_B} \sum_{(i,j) \in E_T} (m_{ij}(1 - m_{ij}))^\alpha \|x_i - x_j\|. \quad (5.2)$$

In this generalization, which we refer to as the branched CST (BCST), the well-known Steiner tree problem [81, 171] arises when $\alpha = 0$. Figure 5.1 summarizes (B)CST and its limiting cases, only some of which have been studied in the literature so far.

Contributions. 1) We present the novel (B)CST problem and provide empirical evidence for its greater stability on toy and real-world data; 2) We propose an iterative heuristic to estimate the optimum of the BCST problem. By exploiting the connection between the branched and unbranched versions of the CST problem, we are able to use the heuristic defined for the BCST to also approximate the Euclidean CST without Steiner points. We benchmark this heuristic and show its competitiveness. 3) On the theoretical side, we prove that for large α or large $|V|$ and $\alpha > 1$, (B)CST converges to a star-tree (hinting modeling limitations when $\alpha > 1$), and for $\alpha \rightarrow -\infty$, it tends towards a path graph. Additionally, we show analytically that if the terminal points lie on a plane, then for $\alpha \in [0, 0.5] \cup \{1\}$ the Steiner points of the optimal solution can have up to degree 3, and we provide empirical evidence that this holds also for $\alpha \in]0.5, 1[$.

²The same argument applies to the NP-hardness of the branched version of the CST problem, which is explained next.

Outline In Section 5.2, we explore the limiting cases of the (B)CST optimum as α approaches $\pm\infty$, along with the scenario where the number of terminals tends to infinity for $\alpha > 1$. Section 5.3 demonstrates empirically the stability of the (B)CST as α increases. In Section 5.4, we establish a relationship between feasible CST and BCST topologies. Section 5.5 analyzes the geometry of optimal BCST topologies, providing analytical expressions for the branching angles at the Steiner points. Moreover, it discusses the feasibility of 4-degree SPs when the BCST is restricted to the Euclidean plane. Section 5.6 presents a heuristic to approximate the (B)CST optimal solution, while Section 5.7 benchmarks this heuristic on small toy datasets. The conclusions of the chapter are given in section 5.8.

5.2 Limit Cases of the CST/BCST Problems Beyond the Range $\alpha \in [0, 1]$

The CST problem, as defined in (5.1), as well as its branched version are parameterized by α . Throughout the manuscript our attention will be on the α -range of $[0, 1]$, nonetheless it is worth studying the problem beyond this range. We will show that when $\alpha \rightarrow \infty$ or $\alpha > 1$ and $|V| \rightarrow \infty$, the (B)CST tends to a star graph centered on the medoid of the graph, i.e. the node that minimizes the distance to the rest of nodes. Consequently, the case with $\alpha > 1$ becomes inadequate for modeling data structure, as the tree becomes increasingly trivial with a growing number of terminals. Conversely, as $\alpha \rightarrow -\infty$, the CST tends towards the path graph that minimizes the CST objective. These scenarios prove inadequate for modeling data structure, thus we restrict our focus exclusively to the α -range of $[0, 1]$.

5.2.1 Limit Cases Where the Optimum (B)CST Transforms into a Star-Tree

In this section, we delve into scenarios at the limits where the optimal solutions for both the CST and BCST problems converge to a star-tree configuration. Specifically, we demonstrate that this outcome occurs as α approaches infinity and as the number of terminals, denoted by N , tends to infinity when $\alpha > 1$. The later limit case is of special relevance, as it indicates that when the parameter α exceeds 1, both CST and BCST exhibit inadequacy in extracting meaningful structural information from the data. In this situation, an increasing number of data points lead to the formation of a star-tree, which, lacks the capacity to convey pertinent information about the underlying dataset.

Before studying the limit cases, we will analyze the optimum star-tree that minimizes the CST and BCST costs. Note that in a star-tree, all edges are adjacent to a leaf node and, therefore all have the same normalized centrality value, which is equal to $(N - 1)/N^2$, where

N is the number of terminals. In this scenario, the centrality of the edges in the cost function from Equation (5.1) can be factored out, simplifying the problem to the identification of the star graph with the minimum cost. Indeed, if u denotes the center node of a star graph, then its CST objective is equal to

$$\sum_{v \neq u} \frac{N-1}{N^2} c_{uv} = \frac{N-1}{N^2} \sum_{v \neq u} c_{uv}.$$

Consequently, the optimal solution for the CST problem manifests as a star graph centered at the node that minimizes the total distance to all other nodes, effectively the medoid. In the context of the BCST problem, where Steiner Points can be introduced, the star graph is centered at the geometric median. This is because the Steiner Points strategically position themselves to minimize the distance to all nodes. The next result formalizes this statement.

Lemma 5.2.1. The tree-star that minimizes the CST cost is the star-shaped tree centered at the medoid of the terminals, that is, centered at the terminal which minimizes the sum of distances to all nodes. For the BCST case, the tree is centered at the geometric median of all terminals.

As a consequence of this result, we infer that the limit cases wherein both CST and BCST converge to a star-tree will yield stable trees. This stability arises from the consistent output of star-trees, with their centers being the medoid and geometric median—both robust points resilient to noise.

In order to study limit cases where the star-tree emerges as the optimal solution, we establish first a sufficient condition for the CST optimal solution to take the form of a star tree. This condition was first identified by Hu [78] in the context of the Minimum Routing Cost Tree (MRCT), corresponding to the CST with $\alpha = 1$. Hu showed that if a “stronger” variant of the regular triangle inequality holds, then the optimum solution of the MRCT is a star tree. The following theorem extends and generalizes this result for arbitrary values of α .

Theorem 5.2.2. Let N be the number of terminals and c_{ij} be the edge-costs of any pair of points (Steiner or terminals) i, j . If there exists

$$t \leq \min_{\ell \in [2, N/2]} \frac{\left(\frac{\ell(N-\ell)}{N-1}\right)^\alpha - 1}{\ell - 1}$$

such that

$$c_{kv} + t c_{uv} \geq c_{ku} \tag{5.3}$$

for all triplets of nodes u, k, v , then there exists an optimum (B)CST evaluated at α which is a star tree.

We defer the complete proof to Appendix D.4.1, though we sketch briefly the key idea. The proof demonstrates that we can always iteratively increase the degree of certain node, by connecting all neighbors of one of its neighbors to it. If the “stronger” variant of the triangle inequality holds, then this process does not increase the cost. Eventually, a node will reach maximum degree, indicating the formation of a star-tree structure.

Remark 5.2.3. Note that Theorem 5.2.2 states only a sufficient condition, which means that the optimum can be a star tree even if the strong triangle inequality does not hold. Additionally, it is worth to highlight that Theorem 5.2.2 also holds true for the CST problem even when the nodes lack embedding in any specific space, allowing for edge costs with arbitrary values.

Let us define

$$h_1(\ell, N, \alpha) := \frac{\left(\left(\frac{\ell(N-\ell)}{N-1}\right)^\alpha - 1\right)}{\ell - 1} = \frac{\left(1 + \frac{(\ell-1)(N-\ell-1)}{N-1}\right)^\alpha - 1}{\ell - 1}. \quad (5.4)$$

which characterizes the upper limit of the threshold t in Theorem 5.2.2. Equation (5.3) represents a weighted version of the triangle inequality. Specifically, when $t = 1$, equation (5.3) recovers the standard triangle inequality. Moreover, if $h_1(\ell, N, \alpha) \geq 1$, there exists a value t such that $h_1(\ell, N, \alpha) \geq t \geq 1$. This implies that the relation is weaker than the triangle inequality. Therefore, if the triangle inequality holds, equation (5.3) will also hold. As a first consequence, is evident that

$$\lim_{\alpha \rightarrow \infty} h_1(\ell, N, \alpha) = \infty, \quad \forall N \geq 3, \ell \in [2, N/2].$$

This indicates that as α tends to infinity, the optimal tree tends to become a star-tree. The intuition behind this is clear: as α increases, the CST/BCST aims to minimize the centrality of the edges, since the edge costs become relatively insignificant in comparison. Among all edges in a tree, those adjacent to a leaf have the least centrality. Thus, any star graph is a tree that minimizes simultaneously the edge centrality of all its edges

One can also show that h_1 is greater than 1 when $\alpha > 1$ and N approaches infinity:

$$\lim_{N \rightarrow \infty} h_1(\ell, N, \alpha) \geq 1, \quad \forall \alpha > 1, \ell \in [2, N/2],$$

though we leave the technical proof for the Appendix D.4.2. Consequently, we obtain the following corollary

Corollary 5.2.4. As the parameter α approaches infinity, or N approaches infinity and $\alpha > 1$ the CST/BCST optimal solution is a star-shaped tree.

Corollary 5.2.4 states that the optimum tree is a star-tree when $\alpha > 1$ and $N \rightarrow \infty$. What’s intriguing is that this limit is reached at relatively low values of $\alpha \approx 1$ for moderate

values of N . In Appendix D.4.3, we show that for N nodes, the threshold $\alpha^*(N)$ at which $h_1(\ell, N, \alpha) \geq 1$ for all $\ell \in [2, N/2]$ and $\alpha \geq \alpha^*(N)$ is given by

$$\alpha^*(N) = \max \left(\frac{\log(2)}{\log \left(1 + \frac{N-3}{N-1} \right)}, \frac{\log(N/2)}{\log \left(1 + \frac{(N/2-1)^2}{N-1} \right)} \right) \quad (5.5)$$

The function $\alpha^*(N)$ serves as a threshold, ensuring that the optimal solution adopts a star-tree configuration. To illustrate this threshold, Figure 5.2 depicts the function $\alpha^*(N)$. Indeed, we see that when $N = 1000$, $\alpha = 1.15$ is enough to guarantee that the optimum is a star tree. A toy example is presented in Figures 5.3, with $N = 1000$, showcasing an instance where the optimum is indeed a star tree.

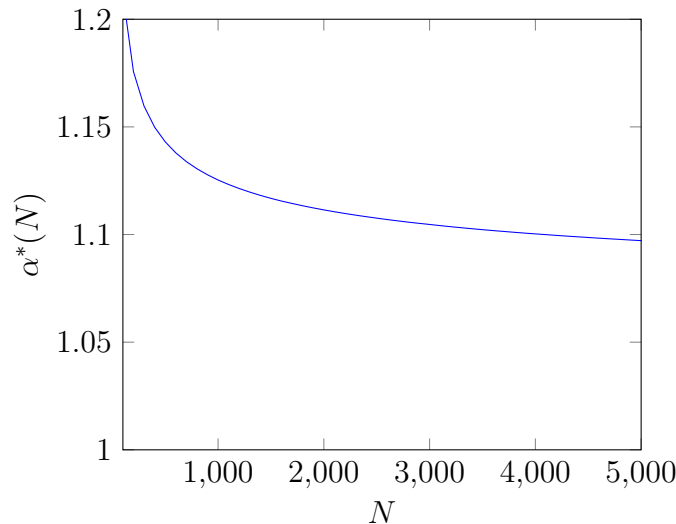


Figure 5.2. Threshold Function $\alpha^*(N)$ Guaranteeing Optimal (B)CST Star-Tree. The threshold function $\alpha^*(N)$, tied to the number of terminals N , defines the minimum α value ensuring the optimal solution for CST/BCST is a star-tree. For all $\alpha > \alpha^*(N)$, the optimum solution is a star-tree. The plot depicts the transition at which the optimum is ensured to be a star-tree is around $\alpha \approx 1$. As N increases, $\alpha^*(N)$ approaches 1, implying that in the limit as N tends to infinity, CST/BCST with $\alpha > 1$ converges to a star-tree.

5.2.2 Limit Cases Where the Optimum (B)CST Transforms into a Path-Tree

Negative values of α favour high central edges, as $(m_e(1 - m_e))^\alpha$ will be lower. Consequently, for sufficiently negative values of α , the CST/BCST problem will prioritize minimizing the number of leaves since the centrality of its adjacent edges attain the minimum centrality. The tree that minimizes the number of leaves is a path. Therefore, when $\alpha \rightarrow -\infty$

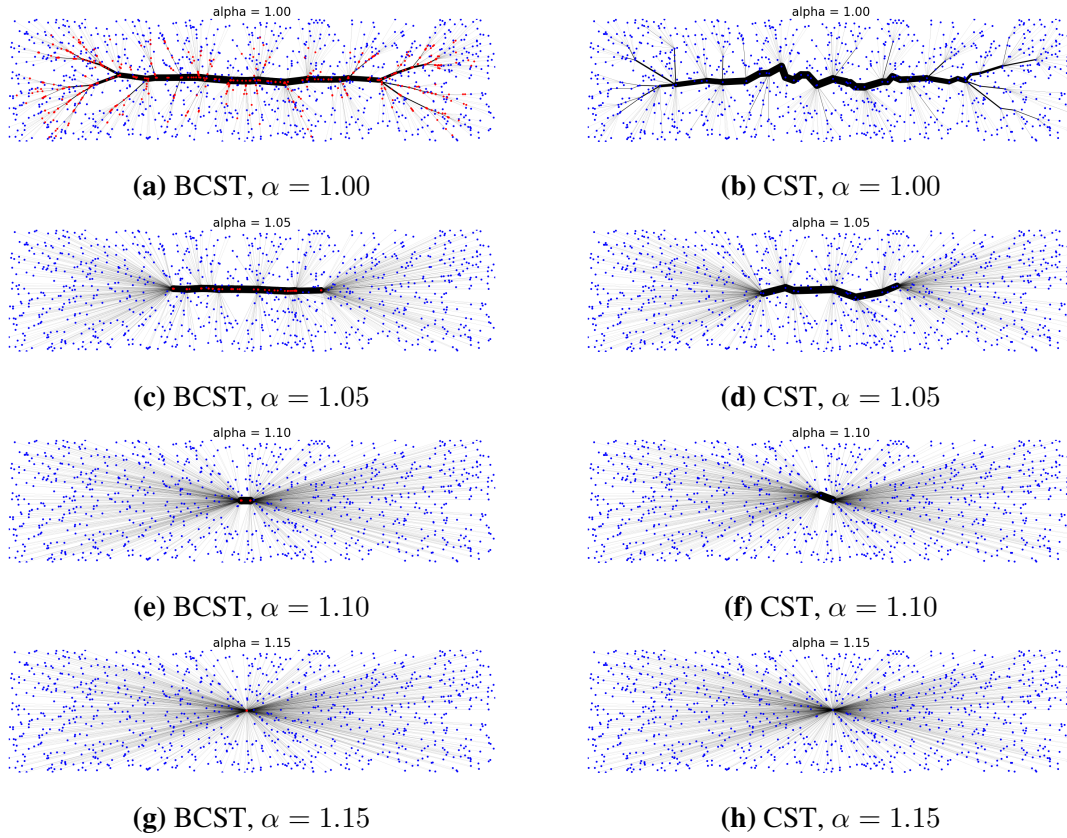


Figure 5.3. (B)CST Star-Tree Optimality with Respect to $\alpha \gtrsim 1$ in Samples Uniformly Drawn from a Rectangle. As α increases, both CST and BCST exhibit a transition towards a star graph. This effect may manifest relatively early. As anticipated in Figure 5.2, for a sample set with 1000, points, the value $\alpha = 1.15$ transforms both CST and BCST into a star graph.

the optimum tree will be the Hamiltonian path that minimizes the CST/BCST objective function. A Hamiltonian path is a path that visits each node exactly once.

Echoing Theorem 5.2.2 we show that if a variant of the triangle inequality holds, then the optimum (B)CST will be a tree.

Theorem 5.2.5. Let N be the number of nodes and c_{ij} be the edge-costs of any pair of points (Steiner or terminals) i, j . If there exists

$$t \leq \min_{\substack{1 \leq s \leq N-3 \\ 1 \leq \ell \leq \min(s, (N-s)/2-1)}} \frac{(\ell(N-\ell))^\alpha - ((\ell+s)(N-\ell-s))^\alpha}{(s(N-s))^\alpha}$$

such that

$$c_{kv} + tc_{uv} \geq c_{ku}$$

for all triangles in the graph, then there exists an optimum (B)CST evaluated at α which is a Hamiltonian path.

In contrast to the proof presented in Theorem 5.2.2, we demonstrate that we can systematically decrease the degree of nodes by iteratively connecting the neighbors of a specific node to one of its neighbors. This iterative reduction does not inflate the cost, provided the triangle inequality variant holds. Ultimately, all nodes will have at most degree 2, meaning that a path has been formed. We defer the complete proof to Appendix D.4.4.

Let

$$h_2(\ell, s, N, \alpha) = \frac{(\ell(N - \ell))^\alpha - ((\ell + s)(N - \ell - s))^\alpha}{(s(N - s))^\alpha}.$$

As a consequence of Theorem 5.2.5, if $h_2(\ell, s) \geq 1$ for $1 \leq s \leq N - 3$ and $1 \leq \ell \leq \min(s, (N - s)/2 - 1)$, then the satisfaction of the triangle inequality is a sufficient condition to ensure that the optimal (B)CST topology is a path. We can easily check that as α approaches minus infinity the following limit holds

$$\begin{aligned} \lim_{\alpha \rightarrow -\infty} h_2(\ell, s, N, \alpha) &= \lim_{\alpha \rightarrow -\infty} \left(\underbrace{\left(\frac{\ell(N - \ell)}{s(N - s)} \right)^\alpha}_{\leq 1} - \left(\underbrace{\frac{((\ell + s)(N - \ell - s))}{(s(N - s))}}_{> 1} \right)^\alpha \right) \\ &= \begin{cases} 1, & \text{if } \ell = s \\ \infty, & \text{if } \ell > s \end{cases}, \end{aligned} \quad (5.6)$$

where we have used the inequalities $1 \leq s \leq N - 3$ and $1 \leq \ell \leq \min(s, (N - s)/2 - 1)$. Consequently, as α approaches $-\infty$, the optimum tree will tend to a path tree. Note however, that if $\ell = s$, then $h_2(\ell, \ell, N, \alpha) < 1$. In this case, according to Theorem 5.2.5, the optimum tree can only be a path for α negative enough, if the triangle inequality holds strictly for all triplets of nodes. Nonetheless, in Corollary D.4.6, we show that when the points lie on a geodesic space, then the requirement of the strict triangle inequality is not necessary for the optimum to become a path as α approaches $-\infty$.

In the previous section, we demonstrated that for a given $\alpha > 1$ we can find N large enough, such that for any configuration of terminals the optimal solution of the CST/BCST problem would be a star-tree. However, in this case, we cannot ensure that if $\alpha < 0$, there exists N large enough, where the optimum is a path. Indeed, given a fixed $\alpha \in \mathbb{R}$ and setting $\ell = N/4 - 1$ and $s = N/2$, we have that the limit of $h_2(N/4 - 1, N/2, N, \alpha)$ as N increases is given by

$$\lim_{N \rightarrow \infty} h_2\left(\frac{N}{4} - 1, \frac{N}{2}, N, \alpha\right) = \lim_{N \rightarrow \infty} \frac{\left(\left(\frac{N}{4} - 1\right)\left(\frac{3N}{4} + 1\right)\right)^\alpha - \left(\left(\frac{N}{4} + 1\right)\left(\frac{3N}{4} - 1\right)\right)^\alpha}{\left(\frac{N}{2}\right)^{2\alpha}} = 0 < 1.$$

Consequently, Theorem 5.2.5 will not guarantee that the optimum is a path, unless all edge-costs are equal.

5.3 Stability of the CST Problem

5.3.1 Toy Data

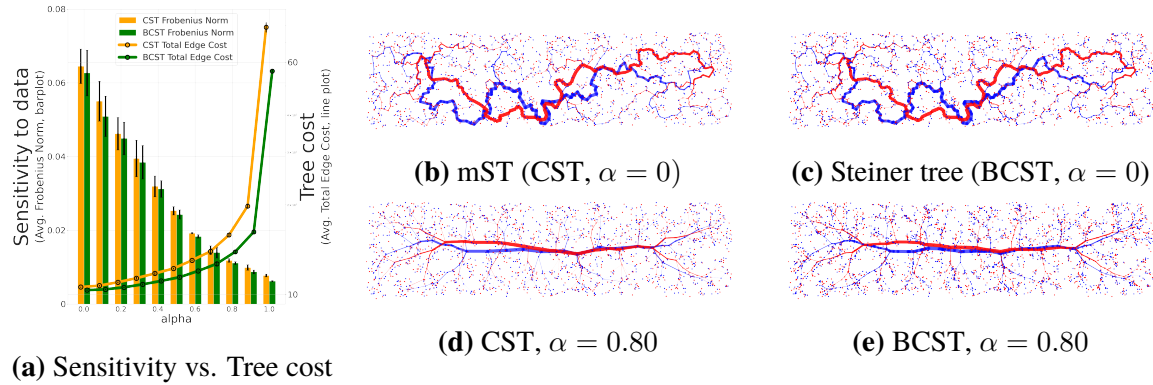


Figure 5.4. (B)CST Robustness Analysis. (B)CST for $\alpha > 0$ are more robust to noise and adhere to large scale structure in the data better than the mST and Steiner tree. **Left)** When increasing α , the sensitivity to random density fluctuations in the data decreases (good). At the same time, the total length of the tree increases (bad). This tradeoff can be adjusted with a single hyperparameter α . More details in Section 5.3.1. **Right)** CST and BCST of two samples, red and blue, drawn from the same distribution. The tree backbone reflects the global structure more accurately for $\alpha > 0$ than for $\alpha = 0$. Edge widths are proportional to their centrality. All trees except for the mST were computed using the heuristic proposed in Section 5.6.2. See Appendix D.1 for more examples.

We explore the robustness of the (B)CST problem against data perturbations by comparing the (B)CST topologies as small noise is introduced. In a toy example, three sets of 1000 points, uniformly sampled from a rectangle, are perturbed with Gaussian noise, yielding five perturbed sets per original set. We then compute the CST and BCST for $\alpha \in \{0, 0.1, \dots, 0.9, 1\}$, deeming a tree robust if minor data perturbations lead to minor structural changes in the tree. Formally, we consider a method δ -robust, if, for any sets of points P_1 and P_2 and their respective trees T_1 and T_2 ,

$$d_T(T_1, T_2) \leq \delta d_P(P_1, P_2),$$

where d_T and d_P measure tree and set distances, respectively. The set distance d_P quantifies the perturbations we aim to withstand, where lower d_P values correspond to sets that are similar based on specific criteria. In our example, we define d_P as the average distance between points and their perturbed counterparts. Since we apply the same noise to each point, the average distance between points approximates the Gaussian noise's standard deviation, making it nearly constant. To quantify structural tree changes, we set d_T equal to the Frobe-

nius norm of the shortest path distance matrices between the original and perturbed (B)CST trees.

Figure 5.4a shows the average Frobenius norm between the original and corresponding perturbed samples across various α values. It is evident that as α increases, there is a noticeable decrease in the Frobenius norm. Since our d_P is fairly constant, showcasing that the Frobenius norm decreases implies a reduction in δ as α rises, i.e. the trees become more robust. However, we also plot the average cost of the trees (sum of the individual edge costs), which increases with α . Thus, the improvement in robustness comes at the expense of adding longer edges. This pattern is expected because, as α increases, the (B)CST tends to a medoid-centered star graph (see Section 5.2.1). This graph will have long edges but will also exhibit robustness to noise due to the medoid’s inherent stability. According to our definition of δ -robustness, the $\alpha \rightarrow \infty$ (B)CST limiting case, which always outputs a star-graph, will be deemed robust despite its undesirability for describing the data structure.

We associate the data structure with the graph node interconnectivity, wherein shorter edges preserve it better. Thus, α serves as a parameter trading off stability vs. data fidelity. Indeed, the mST and Steiner tree ($\alpha = 0$) on the right side of Figure 5.4 are highly sensitive to minor data changes due to their greedy nature, prioritizing shorter edges. Conversely, the (B)CST solutions at $\alpha = 0.8$ are more stable, faithfully representing the data’s overall layout, albeit with longer edges.

5.3.2 Real-World Data

Being able to effectively summarize data structure without being overly affected by random variations is crucial in various contexts. Here, we offer a brief demonstration of how the (B)CST can be applied in single-cell trajectory inference. In the upcoming chapter, Chapter 6, we will delve into its application in 3D-Plant skeletonization.

Single-cell transcriptomics analyzes the gene expression levels of individual cells in a particular population by counting the RNA transcripts of genes at a given time. The high dimensional single cell RNA-sequencing data can be used to model the gene expression dynamics of a cell population as well as the cell differentiation process. The reconstruction of these trajectories can help discover which genes are critical to understand the underlying biological process. It is often assumed that these trajectories can be represented as trees [147, 159], and therefore the (B)CST can be applied to model such trajectories. We will apply the

We show results on the data from [140], here denoted as Paul dataset. The Paul dataset consists of gene expressions measurements of cells of mouse bone marrow [140]. The original dataset is formed by 2730 cells each with 3451 gene measurements. The data is pre-

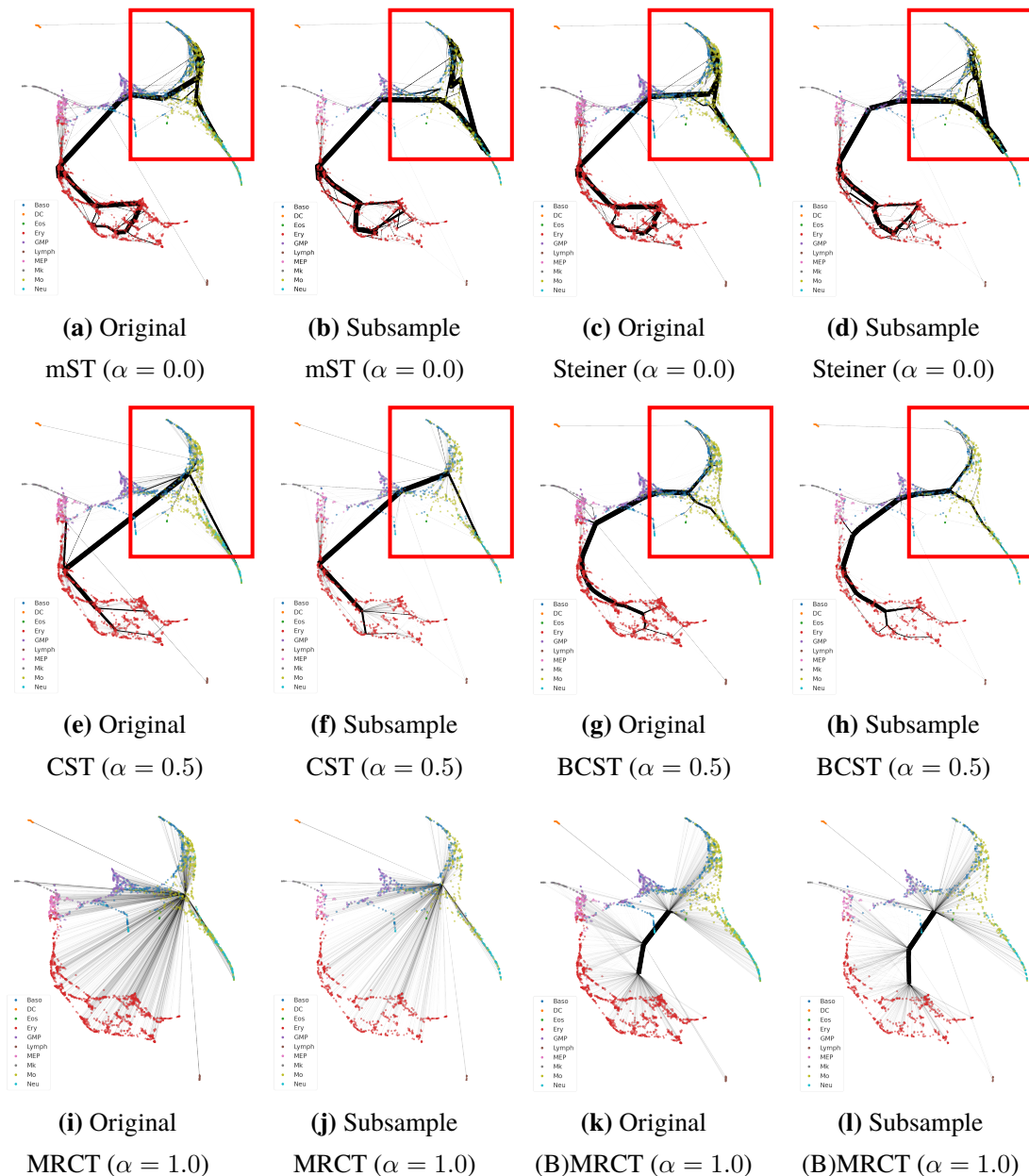


Figure 5.5. mST, (B)CST and BMRCT of the Paul Dataset. We applied the algorithms to both the original data (top) and a perturbed version with half of the points randomly removed (bottom). PAGA was used for 2D visualization, while the trees were computed in a 50-dimensional PCA projection. Colors represent different cell populations. The width of the edges is proportional to their centralities. 5.5a-5.5d) In the original data, the mST and Steiner tree do not faithfully model the trajectory bifurcation highlighted by the rectangle. Moreover, the trajectory changes drastically at this point once a subset of the samples is removed. 5.5e-5.5h) The CST and BCST at $\alpha = 0.5$ are able to detect the bifurcation, and preserve the main backbone of the tree after the data has been perturbed. 5.5i-5.5l) The MRCT and its branched version are robust to perturbations due to its star shape structure, but this shape is also responsible for its incapability to model the data properly

processed using the recipe described in [191], which reduces the dimensionality to 1000 by selecting the most relevant genes. We further reduce the dimensionality of the data, by applying PCA with 50 principal components. Finally, we apply the corresponding spanning tree algorithm.

For visualization purposes, we used the PAGA algorithm [175], one of the best algorithms for single cell trajectory inference [147]. PAGA was designed to faithfully represent the trajectories. Thus, if a spanning tree aligns well with the embedding, this is an indication that the tree approximates the trajectory well.

To study robustness we perturb the data by removing half of the samples and then compare how the backbone of different spanning trees is affected by this perturbation. Figure 5.5 shows the mST ($\alpha = 0$) and CST, BCST (at $\alpha = 0.5$) and (B)MRCT ($\alpha = 1$) of the original sample and a perturbed sample with 50% of the cells randomly sampled. The mST misses the highlighted bifurcation and it is more sensitive to the noise. The CST and BCST are robust to the perturbation and align well with the PAGA embedding, though the CST may not reconstruct well the finer details. The addition of SPs enables the BCST to follow the trajectory more closely.

The MRCT results in a star tree, and its branched version nearly resembles a star tree as well. In high-dimensional data, such as the Paul dataset with 50 dimensions, the star-shaped tendency becomes more prominent. For $\alpha = 1$, most of the intricate structure of the data is lost. Therefore, for higher dimensions lower α values become more relevant.

5.4 Correspondence Between the BCST and CST Topologies

Both the CST and the BCST problems have to be optimized over the set of feasible spanning tree topologies. This optimization is combinatorial in nature and turns out to be NP-hard in both cases. For the CST case, Cayley's formula tells us that the number of feasible topologies is equal to N^{N-2} [29], which grows super-exponentially with the number of nodes N . For the BCST case, w.l.o.g., we can represent any feasible topology as a full tree topology, i.e. as a tree with $N - 2$ Steiner points, each of degree 3, and with leaves corresponding to the N terminals. This representation is justified by the fact that any other feasible topology with SPs of degree higher than 3 can be represented by a full tree topology by collapsing two or more SPs, that is, when two or more SPs have the same coordinates. Figure 5.6 illustrates how a single full tree topology can realize different feasible topologies of the BCST problem. For N terminals, the number of possible full topologies is equal to $(2N - 5)!! = (2N - 5) \cdot (2N - 7) \cdot \dots \cdot 3 \cdot 1$ [151], which also scales super-exponentially, but at a lower rate than

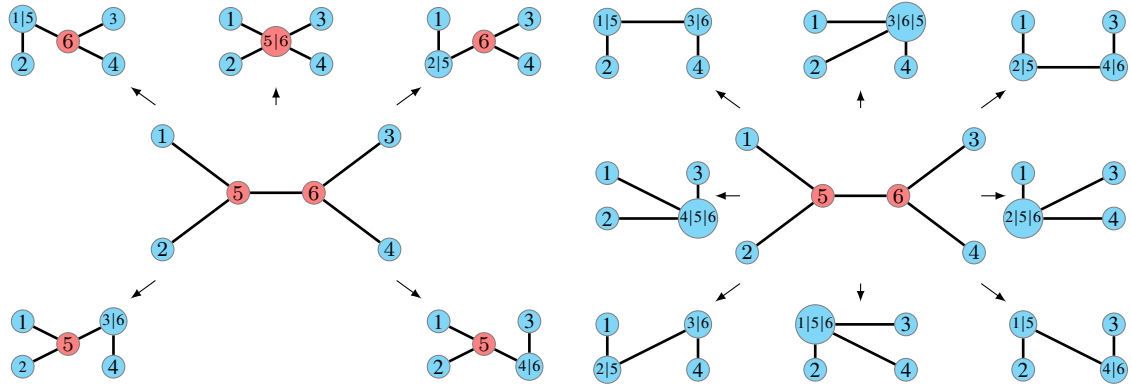
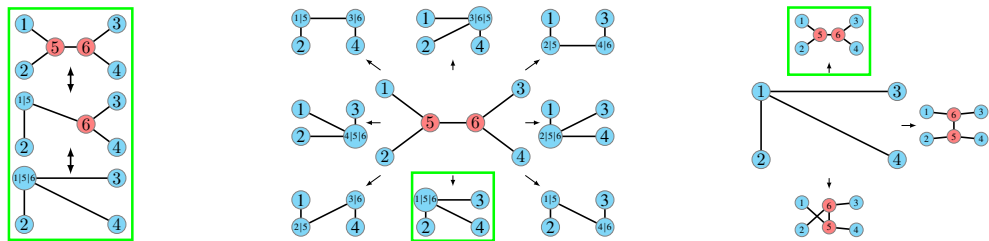


Figure 5.6. Realizable BCST Solutions Topologies from a Full Tree Topology with 4 Terminals. Terminal nodes and SPs are represented in blue and red, respectively. Different topologies emerge from a single full tree topology depending on how the SPs collapse with other nodes.



(a) Steps $T_{CST} \leftrightarrow T_{BCST}$ (b) Topologies derived from T_{BCST} (c) Topologies derived from T_{CST}

Figure 5.7. CST and BCST Topology Correspondence. 5.7a) Mapping from a BCST topology, T_{BCST} , to a CST one, T_{CST} and vice versa. Terminal nodes and SPs are represented in blue and red, respectively. From top to bottom it is shown how the SPs couple to different terminals. From bottom to top, the SPs are spawned by different pairs of adjacent neighbors. First by the pair 3 and 4 and later by 2 and 6. 5.7b) T_{CST} feasible topologies derived from a single T_{BCST} . 5.7c) T_{BCST} feasible topologies derived from a single T_{CST} .

the number of topologies of the CST. Consequently, an exhaustive search through all trees is not feasible.

The heuristic presented in Section 5.6 exploits the correspondence between the feasible topologies of the BCST and the CST problems. Given a full tree topology T_{BCST} , we say that a topology T_{CST} of the CST problem can be derived from T_{BCST} if: 1) we can collapse the SPs of T_{BCST} with the terminals such that the resulting topology is T_{CST} , and 2) for any SP s that is collapsed with terminal t , then all SPs along the path connecting s to t must also collapse with t . In other words, a SP cannot overtake any other SP in the collapse process. Figure 5.7a (from top to bottom) shows the steps to transform a topology T_{BCST} into a topology T_{CST} by iteratively collapsing the SPs. Analogously, we can derive a topology T_{BCST} from T_{CST} by spawning SPs from the terminals in T_{CST} , i.e., introducing SPs connected to the terminals.

Since in a full tree topology SPs have degree 3 and terminals have degree 1, we add one SP per each pair of nodes adjacent to a common terminal node, so that the SP is connected to the triple of nodes.

The correspondence mapping between T_{CST} and T_{BCST} , as shown in Figures 5.7b and 5.7c, is not unique. Multiple T_{CST} can be derived from a single T_{BCST} and vice versa. At most, any T_{BCST} can generate $O(3^{N-2})$ T_{CST} topologies because each SP locally has 3 nodes to merge with. In concrete, the number of T_{CST} topologies derivable from a singular full tree topology is given by the determinant of a submatrix of the Laplacian matrix, denoted as L_{SP_s, SP_s} . This submatrix is obtained by selecting the rows and columns indexed by the SPs. Hence, the number of derivable topologies is precisely $\det L_{SP_s, SP_s}$. This assertion is proven in Theorem D.5.2 (refer to Appendix D.5.2 for details). The proof leverages the bijective relationship between the derivable CST topologies and the terminal separating spanning forests of a full tree topology. By combining this bijectivity with the fact that the minors of the Laplacian matrix provide the count of forests separating the non-indexed rows and columns, the desired result is established.

Similarly, the number of T_{BCST} topologies derived from T_{CST} is given by

$$\prod_{v : d_v \geq 2} (2d_v - 3)!!, \quad (5.7)$$

where d_v is the degree of terminal v in the T_{CST} topology. Higher-degree terminals can generate more topologies as they can spawn more pairs of nodes. For more details on the cardinalities of derivable topologies, see Appendix D.5.

Despite the mapping ambiguity between the topologies, we can reduce the number of trees to explore in the CST/BCST given a BCST/CST topology. Although the optimum of one problem is not guaranteed to be derived from the optimum of the other (see Figure 5.8), we show empirically that the heuristic proposed in Section 5.6 can exploit the positions of the SPs together with the correspondence between the sets of topologies of both problems to produce competitive results.

5.5 Geometry of Optimal BCST Topologies

In this section, we will analyze the geometry of the optimal topologies of a BCST problem. Concretely, we will determine an analytical formula of the angles that form the edges at a SP. Using this relation, we will prove that when the terminal points lie in a plane, then SPs with degree higher than 3 are not realized in optimal solutions for $\alpha \in [0, 0.5] \cup \{1\}$ unless they collapse with a terminal. For $\alpha \in]0.5, 1[$ we provide empirical evidence that the statement also holds in that case.

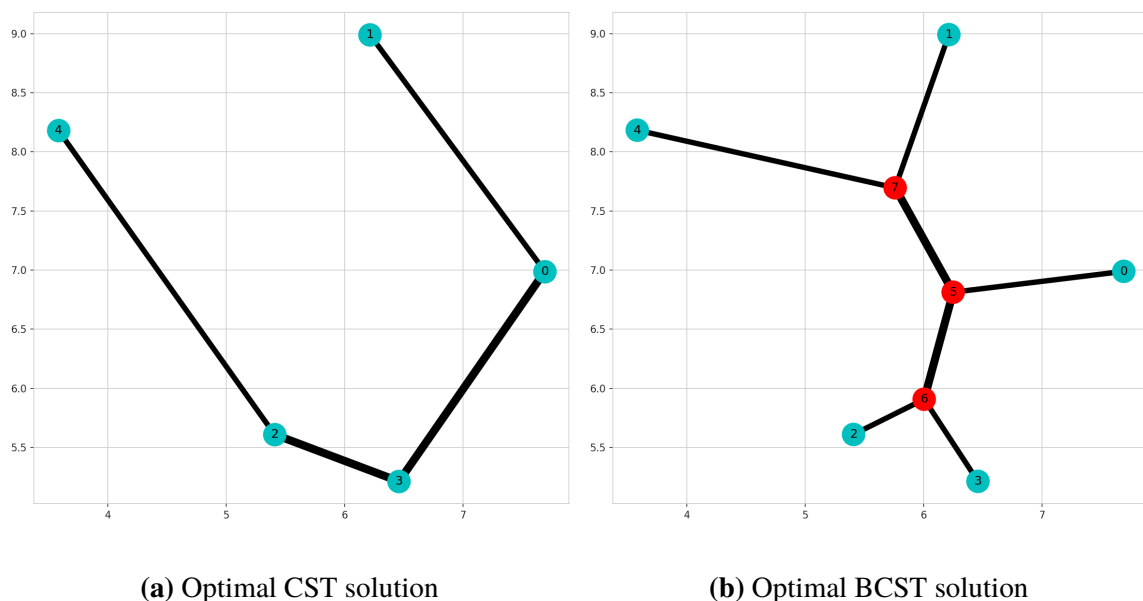


Figure 5.8. Optimal CST and BCST Topologies May Not Be Derived from Each Other Given the Same Terminal Configuration. Left: Optimal CST solution. Right: Optimal BCST solution. The CST topology cannot include nodes 4 and 1 as direct neighbors if derived from BCST, as it would result in nodes adjacent to a common neighbor. Similarly, the optimal BCST topology cannot be derived from the CST topology, as nodes 1 and 4 would not be connected to a common SP.

5.5.1 Branching Angles at the Steiner Points

In this section, we formulate the branching angles in terms of the centralities of the edges for a given topology of the BCST problem. The derivation of the angles is based on previous works [18, 108], which apply analogous arguments for the Branched Optimal Transport (BOT) problem. The main difference lies in the weighting factors that multiply the distances in the objective function (5.2), which in our case are the edge betweenness centralities, and in BOT are flows matching supply to demand. Appendix D.2.1 elaborates on the similarities and differences between the BOT and BCST problems.

First and foremost, we emphasize the locality characteristic of the geometric optimization of SPs of the BCST problem. Because of the convexity of the BCST objective (5.2), it can be shown that the geometric optimization of the SPs coordinates can be solved locally, meaning that the optimal position of a SP is determined by its neighbors and weighting factors. Lemma 5.5.1 formalizes this statement. For a proof, we refer to Lemma 2.1 of [108], where the same statement was shown for the BOT problem. Since the proof is independent of the weighting factors of the distances, the result applies to the BCST problem as well.

Lemma 5.5.1. Given a topology, its SPs are in optimal position w.r.t. the BCST problem

if and only if any individual SP interconnects its neighbors at minimal cost. Moreover, the optimal topology of the BCST is optimal if and only if for any subset of connected nodes the corresponding subtopology solves the respective subproblem.

Proof: See Lemma 2.1 of [108]. □

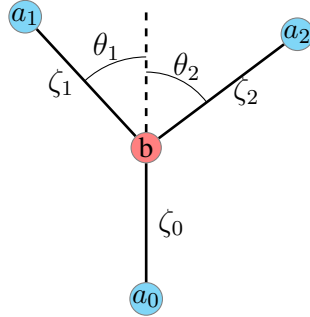


Figure 5.9. Branching Angles at Steiner Point. The symbols ζ_i represent the normalized centralities of the edges, that is $\zeta_i := m_{ba_i}(1 - m_{ba_i})$.

Recall that any feasible topology of the BCST problem can be represented as a full tree topology where each SP has degree 3. Thus, as a consequence of Lemma 5.5.1, it is enough to study the geometric optimization of 3 nodes connected by a single SP. Consider the problem configuration depicted in Figure 5.9, where node b represents the branching point whose coordinates need to be optimized, nodes $\{a_i\}_i$ are the terminals with fixed positions and $\{\zeta_i := m_{ba_i}(1 - m_{ba_i})\}_i$ are the normalized centralities of the edges $\{(b, a_i)\}_i$. The objective to be minimized is

$$C(b) = \zeta_0 \|b - a_0\| + \zeta_1 \|b - a_1\| + \zeta_2 \|b - a_2\| \quad (5.8)$$

Bernot et al. showed that when the b does not collapse with any terminal, then the angles θ_1 and θ_2 are given by

$$\begin{aligned} \cos(\theta_1) &= \frac{\zeta_0^{2\alpha} + \zeta_1^{2\alpha} - \zeta_2^{2\alpha}}{2\zeta_0^\alpha \cdot \zeta_1^\alpha} \\ \cos(\theta_2) &= \frac{\zeta_0^{2\alpha} + \zeta_2^{2\alpha} - \zeta_1^{2\alpha}}{2\zeta_0^\alpha \cdot \zeta_2^\alpha} \\ \cos(\theta_1 + \theta_2) &= \frac{\zeta_0^{2\alpha} - \zeta_1^{2\alpha} - \zeta_2^{2\alpha}}{2\zeta_1^\alpha \cdot \zeta_2^\alpha} \end{aligned} \quad (5.9)$$

Alternatively, we can analyze when node b collapses with one of the terminals. Assuming without loss of generality that b collapses with a_0 , let $\gamma := \angle a_1 a_0 a_2$. According to Lippmann et al., b collapses with a_0 if

$$\gamma \geq \arccos \left(\frac{\zeta_0^{2\alpha} - \zeta_1^{2\alpha} - \zeta_2^{2\alpha}}{2\zeta_1^\alpha \cdot \zeta_2^\alpha} \right) = \theta_1 + \theta_2. \quad (5.10)$$

Hence, b collapses to a_0 if $\angle a_1 a_0 a_2$ exceeds the optimal angle specified by (5.9). This scenario results in the so-called V -branching.

For a comprehensive derivation of the angles, please refer to Appendix D.6, where we present the arguments from the works of Bernot et al. [18] and Lippmann et al. [108].

5.5.2 Infeasibility of Degree-4 Steiner Points in the Plane

In this section, we will prove the infeasibility of degree 4 SPs in the optimal solution of the BCST. Specifically, we will focus on the scenario where the terminal nodes lie in the plane and the value of α falls within the range $\alpha \in [0, 0.5] \cup \{1\}$. Moreover, we will provide compelling evidence to support the validity of the statement for the case where $\alpha \in]0.5, 1[$. We will divide the proof into two parts: one for $\alpha \in [0, 0.5]$, presented here, and the other for $\alpha = 1$, which is detailed in Appendix D.7.

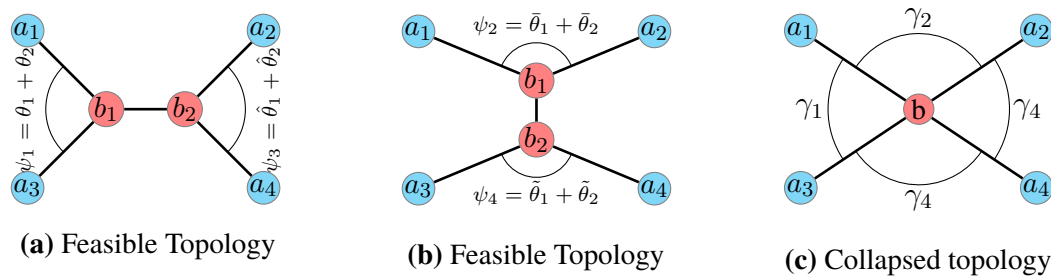


Figure 5.10. Optimal Angles in Degree-4 SP Require $\psi \leq \gamma_i$. Figures 5.10a) and 5.10b) depict two topologies, where the optimal angles given by equation (5.10) are represented by ψ_i . Figure 5.10c illustrates the collapsed solution with the corresponding angles γ_i . An essential requirement for the optimality of Figure 5.10c is that $\psi_i \leq \gamma_i$.

Theorem 5.5.2. Let $\alpha \in [0, 0.5]$. Given a set of terminals which lie in the plane, then the SPs of the optimal solution of the BCST problem will not contain SPs of degree 4 unless these collapse with a terminal.

Proof: The optimality of a solution in the BCST problem relies on the locality characteristic, as stated in Lemma 5.5.1. Specifically, each subtopology within a connected subset must solve its respective problem for the overall solution to be optimal. Consequently, the realization of a degree 4 SP, as depicted in Figure 5.10, requires the collapse of b_2 with b_1 . Moreover, this collapse must occur in any topology. As we have discussed in Section 5.5.1, both nodes b_1 and b_2 will collapse if a V-branching occurs, that is if the angle realized between the collapsed node and the two other nodes connected to it exceeds the optimal angle given by (5.10). Therefore, from Figure 5.10 it follows that $\gamma_i \geq \psi$. We will demonstrate that the sum of $\sum_{i=1}^4 \gamma_i$ is greater than 2π , rendering a SP of degree 4 infeasible. To do this we will prove that $\psi_i > \pi/2$ for all i .

W.l.o.g. let us consider $i = 1$ and denote ψ_1 as ψ . If $\cos(\psi) < 0$, the angle $\psi \in [0, \pi]$ will be greater than $\pi/2$. Based on (5.10), we can derive the following:

$$\cos(\psi) = \frac{F(m_{a_3b_1} + m_{a_1b_1})^{2\alpha} - F(m_{a_3b_1})^{2\alpha} - F(m_{a_1b_1})^{2\alpha}}{2F(m_{a_3b_1})^\alpha F(m_{a_1b_1})^\alpha} \quad (5.11)$$

The function $F(x)^{2\alpha} = (x(1-x))^{2\alpha}$ is strictly subadditive in \mathbb{R}^+ for $\alpha \in [0, 0.5]$,³ that is $F(x+y)^{2\alpha} < F(x)^{2\alpha} + F(y)^{2\alpha}$ if $x, y > 0$. Thus, the numerator of (5.11) is negative and therefore $\cos(\psi) < 0$. Consequently, $\psi > \pi/2$. This argument applies to all ψ_i , hence their sum will be greater than 2π . Consequently, a SP with degree 4 cannot be part of an optimal solution. \square

In [108], Lippmann et al. applied the same argument to establish the infeasibility of 4-degree SPs in the context of the BOT problem for $\alpha \in [0, 0.5]$. Our proof shows explicitly that this argument generalizes to other minimization problems of the same nature, where the edge-lengths are multiplied by weighting factors $F(m_{ij})$, with $F(\cdot)$ being a positive function dependent on m_{ij} and exhibiting subadditivity when squared. For higher $\alpha > 0.5$ the argument used in the previous theorem does not apply. Indeed, we can find values for the edge centralities for which the lower bounds of γ_i , given by (5.10), are all lower than $\pi/2$.⁴ Now, we will take a more general approach that can rule out degree-4 branching for higher α values. In concrete, we will show the infeasibility for $\alpha = 1$.

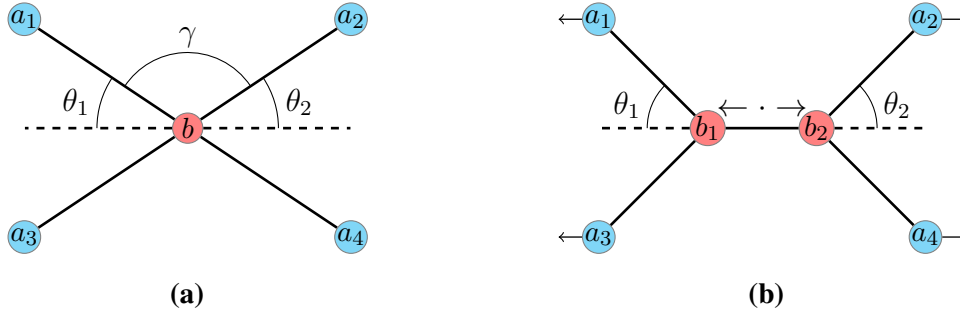


Figure 5.11. Splitting Collapsed SP while Preserving Optimal Angles. 5.11a) illustrates the collapsed solution of a 4-terminal configuration. (5.11b) demonstrates that it is possible to move jointly the terminal points $\{a_1, a_3\}$ in a specific but opposite direction to the one of the terminals $\{a_2, a_4\}$, resulting in the splitting of the collapsed SP b into two distinct SPs, b_1 and b_2 . Remarkably, this split can be executed while preserving the angles θ_1 and θ_2 . Importantly, these angles must correspond to the optimal angles given by (5.9).

The optimal position of the SPs is continuously dependent on the terminal positions and solely relies on the branching angles, as shown in Section 5.5.1. Consequently, assuming that

³In fact, all concave functions, f , with $f(0) \geq 0$ are subadditive on the positive domain. It is easy to see that $(x(1-x))^{2\alpha}$ is concave for $\alpha \in [0, 0.5]$.

⁴For instance for $\alpha = 1$, $m_{a_3b_1} = m_{a_1b_1} = 0.2$ and $m_{a_2b_2} = m_{a_4b_2} = 0.3$ all ψ_i angles are acute. Hence their sum is also lower than 2π .

there exists a configuration such that the SPs collapse, it is possible to find terminal positions that lead to an unstable collapse of the SPs. Here, instability refers to a configuration where an infinitesimal translation of the terminals results in the splitting of the SPs. This scenario is depicted in Figure 5.11. In such cases, the angles realized by the terminals and the SPs will reach the upper bounds specified by (5.10). Therefore, the angles depicted in Figure 5.11a fulfill the condition

$$\gamma = \pi - \theta_1 - \theta_2, \quad (5.12)$$

where the angles satisfy

$$\cos(\gamma) = \frac{F(m_{a_1b} + m_{a_2b})^{2\alpha} - F(m_{a_1b})^{2\alpha} - F(m_{a_2b})^{2\alpha}}{2F(m_{a_1b})^\alpha F(m_{a_2b})^\alpha}, \quad (5.13)$$

$$\cos(\theta_1) = \frac{F(m_{a_3b} + m_{a_1b})^{2\alpha} + F(m_{a_1b})^{2\alpha} - F(m_{a_3b})^{2\alpha}}{2F(m_{a_3b} + m_{a_1b})^\alpha F(m_{a_1b})^\alpha}, \quad (5.14)$$

$$\cos(\theta_2) = \frac{F(m_{a_2b} + m_{a_4b})^{2\alpha} + F(m_{a_2b})^{2\alpha} - F(m_{a_4b})^{2\alpha}}{2F(m_{a_2b} + m_{a_4b})^\alpha F(m_{a_2b})^\alpha}, \quad (5.15)$$

with $F(x) = x(1 - x)$. By processing further equation 5.12, we arrive at the following expression (see AppendixD.7 for further details)

$$(\cos(\gamma) + \cos(\theta_1) \cos(\theta_2))^2 - (1 - \cos(\theta_1))^2 (1 - \cos(\theta_2))^2 = 0. \quad (5.16)$$

Solving (5.16) analytically for all α is difficult. Nonetheless, in AppendixD.7 we show analytically that for $\alpha = 1$ (5.12) can not hold given the constraints on the terms $m_{a_i,b}$, namely $\sum_{i=1}^4 m_{a_i,b} = 1$ and $0 < m_{a_i,b} < 1$ for all i .

Theorem 5.5.3. Let $\alpha = 1$. Given a set of terminals which lie in the plane, then the SPs of the optimal solution of the BCST problem will not contain SPs of degree 4 unless these collapse with a terminal.

Proof: See Appendix D.7 □

Though we have not been able to prove analytically the infeasibility of degree-4 SPs for $\alpha \in]0.5, 1[$, we strongly believe that the statement still holds. Figure 5.12 shows the surface plots of the numerator of the equality 5.16 (once expanded) w.r.t. m_1 and m_2 for different fixed values of α and m_3 . Upon analysis, it appears that the numerator exhibits an increasing trend with respect to α within the interval $[0.5, 1]$. This observation leads us to hypothesize that if the equality holds for $\alpha = 0.5$ and $\alpha = 1$, it is likely to hold for intermediate values as well. However, due to the complexity of the formula, it is challenging to verify this hypothesis analytically.

In section 5.2.1, we have demonstrated that as $\alpha > 1$ and $N \rightarrow \infty$, the BCST tends to converge to a star graph centered at the geometric median. Consequently, for $\alpha > 1$, a degree-4 SP becomes feasible.



Figure 5.12. SPs of Degree 4 in the Plane are Likely not Feasible for $\alpha \in]0.5, 1[$. Surface plots are depicted, illustrating the left side of equation (5.16), as a function of m_1 and m_2 , with different fixed values of α and m_3 . From left to right: m_3 is fixed and α ranges over $\{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. From top to bottom: α is fixed, α and m_3 ranges over $\{0.1, 0.2, \dots, 0.8, 0.9\}$. For m_3 fixed, m_1 and m_2 range over the domain defined by $\{(x, y) : 0 < x + y + m_3 < 1\}$. In all plots, the function values are negative and tend towards 0 as $m_1, m_2,$ or m_3 approaches 0. We can observe that for fixed m_1, m_2 and m_3 , the function seems to be increasing with respect to α (from right to left). Since we have previously demonstrated that the left side of equation (5.16) does not equal zero in the desired domain for $\alpha = 0.5$ and $\alpha = 1$, the plots suggest that this is also the case for $\alpha \in]0.5, 1[$.

Remark 5.5.4. In [108], it was shown for the BOT problem that if degree-4 SPs are not feasible then higher degree SPs are not possible either. The same reasoning applies for the BCST, since the proof does not depend on the weighting factors. Thus for $\alpha \in [0, 0.5] \cup \{1\}$, only degree-3 SPs are feasible unless they do collapse with a terminal node. Due to the compelling evidence shown, we also believe this is the also the case for $\alpha \in]0.5, 1[$. Lippmann et al. [108] also showed that some of the results of the BOT problem obtained on the plane can be extended to other 2-dimensional manifolds. Again, this is also the case for the BCST problem. Among these properties, we emphasize the optimal angles formulae exposed in section 5.5.1 and the infeasibility of degree-4 SPs for appropriate α values. We refer to Appendix F of [108] for more details.

5.6 CST and BCST Optimization Algorithm

This section details the proposed heuristic for optimizing the BCST and CST problems. We will first focus on the BCST. The heuristic iterates over two steps: First, given a fixed topology, the algorithm finds the geometric positions of the Steiner points (SPs) that exactly minimize the cost conditioned on the topology. Given the optimal coordinates of the SPs, we then update the topology of the tree by computing an mST over the terminals and SPs. This procedure is iterated until convergence or until some stopping criterion is met.

5.6.1 Geometry Optimization

The BCST problem can be divided into two subproblems: combinatorial optimization of the tree topology and geometric optimization of the coordinates of the SPs, X_B . When conditioning on a topology T , the BCST objective (5.2) is a convex problem w.r.t. X_B . Despite its convexity, the objective is not everywhere differentiable. We build on the iteratively reweighted least squares (IRLS) approach from Smith [157] and Lippmann et al. [108] to efficiently find the positions of the SPs.

Starting from arbitrary SPs coordinates, denoted as $X^{(0)} = \{x_i^{(0)}\}_{i=1}^{2N-2}$, the algorithm iteratively solves the following linear system of equations.

$$x_i^{(k+1)} = \frac{\sum_{j:(i,j) \in E} \zeta_{ij}^\alpha \frac{x_j^{(k+1)}}{\|x_i^{(k)} - x_j^{(k)}\|}}{\sum_{j:(i,j) \in E} \frac{\zeta_{ij}^\alpha}{\|x_i^{(k)} - x_j^{(k)}\|}}, \quad \forall N+1 \leq i \leq 2N-2. \quad (5.17)$$

where $\zeta_{ij} = m_{ij}(1 - m_{ij})$. We assume, without loss of generality, that the coordinates corresponding to the SPs are indexed from $N+1$ to $2N-2$, where N is the number of terminals. The coordinates for the terminals, which remain fixed throughout all iterations,

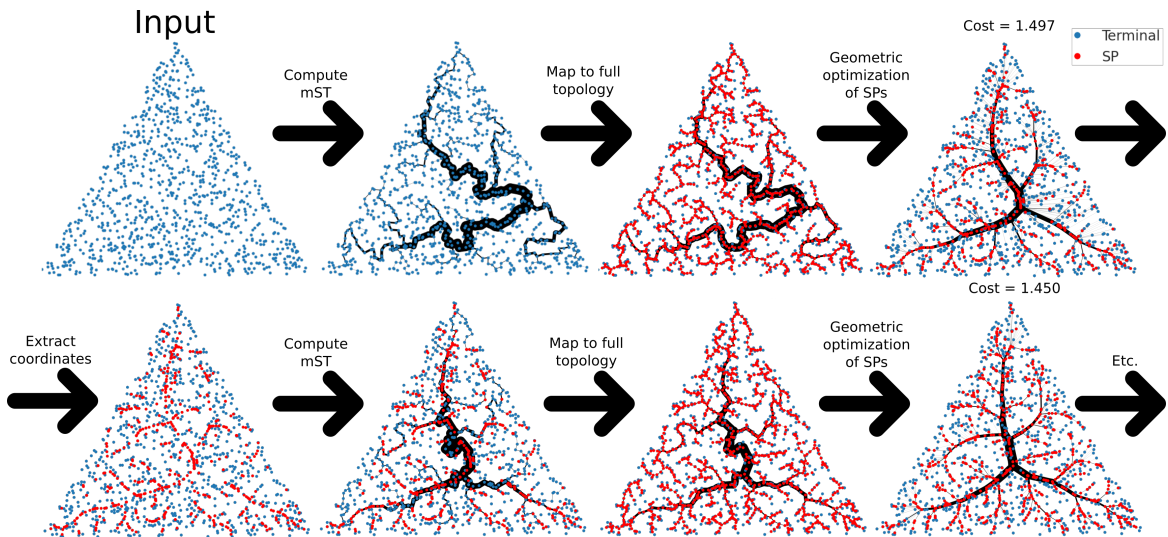


Figure 5.13. mSTreg Heuristic. The mSTreg heuristic iteratively transforms an mST to an approximate BCST. Given a set of points, the heuristic first computes the mST over all points and transforms it into a full tree topology by adding Steiner points (SPs). Next, the optimal positions of the SPs are computed using iteratively reweighted least squares. Given the updated SPs coordinates, the heuristic recomputes the mST over the union of the terminals nodes and the previous SPs. This mST produces a new topology, which is again transformed into a full tree topology by adding SPs whose coordinates are optimized. This process is repeated until some stopping criterion is satisfied.

are represented by the other indices. Thanks to the tree structure of the graph, the linear systems can be efficiently solved in linear time.

In Appendix D.8, we show that the algorithm is agnostic to the weighting factors that multiply the distances, and can therefore be applied to compute any weighted geometric mean.

5.6.2 Heuristic Optimizer for the (B)CST Problem

We now present a heuristic which alternates between the SP geometric coordinate optimization (convex) and a topology update (combinatorial). The heuristic's main characteristic is how it exploits the location of the branching points given an initial topology guess. The heuristic's underlying assumption is that the optimum position of the branching points may suggest a more desirable topology.

Unless otherwise stated, the heuristic we propose starts from the mST over all terminal nodes. At this point, the mST does not contain any SPs and is therefore not a full tree topology. Thus, we need to transform the mST into a full tree topology. As mentioned in Section 5.4, and highlighted in Figure 5.7, this process is not unambiguous. In particular,

for each terminal node v with degree $d_v \geq 2$, we have to add $d_v - 1$ SPs. Consequently, there are $(2d_v - 3)!!$ ways to connect these SPs to the neighbors of v . Among all possible subtopologies connecting the SPs with v and its neighbors, we choose the one given by the dendrogram defined by the hierarchical single linkage clustering algorithm applied to v and its neighbors. In practice, this choice tends to work relatively well since nearby terminals are also closer in the subtopology.

Once we have a full tree topology, we can apply the geometry optimization step to obtain the optimal coordinates of the SPs. We assume that the optimal positions of the SPs indicate which connections between nodes might be more desirable, since they may be biased to move closer to other nodes than the ones to which they are connected. Therefore, we propose to recompute an mST over the terminals together with the SPs. This new mST defines a new topology that needs to be transformed into a full tree topology for the geometry optimization. Once we have a valid full tree topology, we recompute the optimal positions of the SPs. This process is repeated iteratively until convergence or until some stopping criterion is met. We refer to this algorithm as the mST regularization (mSTreg) heuristic. The algorithm's steps are illustrated in Figure 5.13, and its pseudocode is provided in Algorithm 2. The algorithm's complexity is $\mathcal{O}(dn \log(n)^2)$. A detailed complexity analysis is available in Appendix D.9. We remark that the mSTreg heuristic is independent of the weighting factors that multiply the distances, thus it can also be used to approximate other problems as well, for instance a generalized version of the optimum communication tree with SPs.

Optionally, before the mST step is computed over the terminals and previous SPs, we can add intermediate points along the edges of the output generated by the geometry optimization step. These additional points will allow the mST to more reliably follow the edges of the geometry-optimized tree from previous step. Moreover, in case the initial topology was poor, these extra points may help to detect and correct edge crossings, which are known to be suboptimal. An illustration of the effect of these extra points can be found in Appendix D.10.

The heuristic designed for the BCST problem can also be applied to the CST problem by transforming BCST topologies at each iteration into CST topologies. While this transformation isn't unique, we found that iteratively collapsing one SP at a time with the neighbor that leads the smallest increase in cost produces compelling results. Additionally, when collapsing SPs together, centering the new node at the weighted geometric median of its new neighbors improves results slightly. Further details can be found in Appendix D.11.

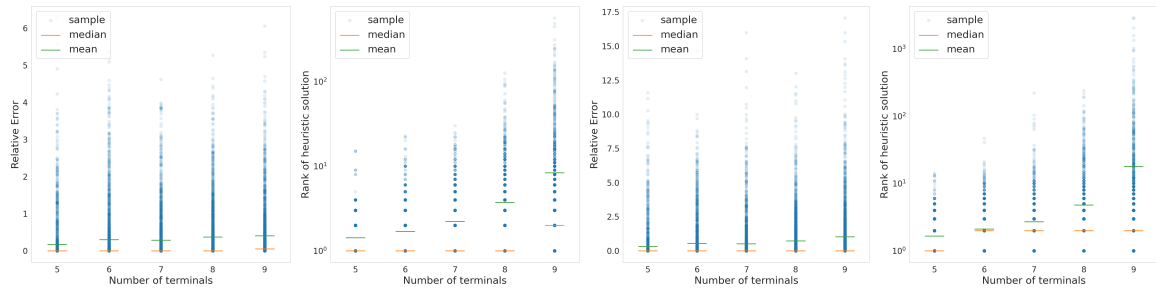
Algorithm 2: mSTreg Heuristic**Input:** X , num_iterations, sampling_frequency, optimize_CST**Output:** Tree

```

1  $mST_{init}$  =minimum_spanning_tree( $X$ ) // Define initial topology as
   mST
2  $T_{BCST}$  =transform2fulltopo( $mST_{init}$ ) // transform topology to full
   tree topology
3  $SP$  =compute_SP( $T_{init}$ ) // compute optimal  $SP$  coordinates
4 bestcost_BCST = $\infty$  if optimize_CST then
5   | bestcost_CST = $\infty$ 

6 while  $it <$  num_iterations do
7   if sampling_frequency  $>$  2 then
8     /* sample extra points from edges */
9      $Y$  =sample_from_edge( $T_{BCST}, X \cup SP$ , sampling_frequency)
10     $SP = SP \cup Y$ 
11   $mST_{X \cup SP}$  =minimum_spanning_tree( $X \cup SP$ )
12   $T_{BCST}$  =transform2fulltopo( $mST_{X \cup SP}$ ) // transform  $mST_{X \cup SP}$  to
   full tree topology
13   $SP$  =compute_SP( $T_{reg}$ ) // compute optimal  $SP$  coordinates
14  if cost( $T_{BCST}$ )  $<$  bestcost_BCST then
15    | bestcost_BCST =cost( $T_{BCST}$ )
16    |  $T_{BCST_{best}} = T_{BCST}$ 
17  if optimize_CST then
18    |  $T_{CST}$  =remove_SP( $T_{BCST}$ ) // Derive CST topology from BCST
   topology
19    | if cost( $T_{CST}$ )  $<$  bestcost_CST then
20      | bestcost_CST =cost( $T_{CST}$ )
21      |  $T_{CST_{best}} = T_{CST}$ 

```



(a) BCST relative error (b) BCST rank of heuristic (c) CST relative error (d) CST rank of heuristic

Figure 5.14. Bruteforce mSTreg Benchmark. Relative cost errors between the mSTreg heuristic and optimal solutions; and sorted position of the heuristic tree for different number of terminals, N . For each N , we uniformly sampled 200 different terminal configuration and solved them for different α values. Most runs ended up close to the global optimum, though the heuristic is slightly better for the BCST problem.

5.7 Benchmark

5.7.1 Brute Force Benchmark

To assess the quality of the mSTreg heuristic, we compare the cost of the trees computed by the mSTreg algorithm with the globally optimal solutions of configurations with up to nine terminals, obtained by brute force. We generate 200 instances with $N \in \{5, 6, 7, 8, 9\}$ terminals sampled from a unit square. Both CST and BCST problems are solved for each $\alpha \in \{0, 0.1, \dots, 0.9, 1\}$. In Figure 5.14, the relative error, calculated as $100(c_h - c_o)/c_o$ where c_h and c_o are heuristic and optimal costs, is shown for different N . We also show how the heuristic solution ranks, once the costs of all topologies are sorted. The heuristic attains the optimum in the majority of cases, with slightly better performance in the BCST problem than in the CST one. Appendix D.12 provides α -based results.

5.7.2 Steiner and MRCT Benchmark

In addition, we evaluate mSTreg on bigger datasets from the OR library⁵ [12] for the Steiner tree ($\alpha=0$) and the MRCT ($\alpha=1$) problems. This dataset includes exact solutions of Steiner problem instances of up to 100 nodes randomly distributed in a unit square. The used instances are labeled as $en.k$, where n denotes the number of terminals, and k represents the instance number. Figure 5.15a compares the cost of our heuristic with the optimal cost. We also provide for reference the costs of the mST and the topology obtained by transforming the mST into a full tree topology with its SP coordinates optimized (referred to

⁵<http://people.brunel.ac.uk/~mastjjb/jeb/orlib/esteinfo.html>

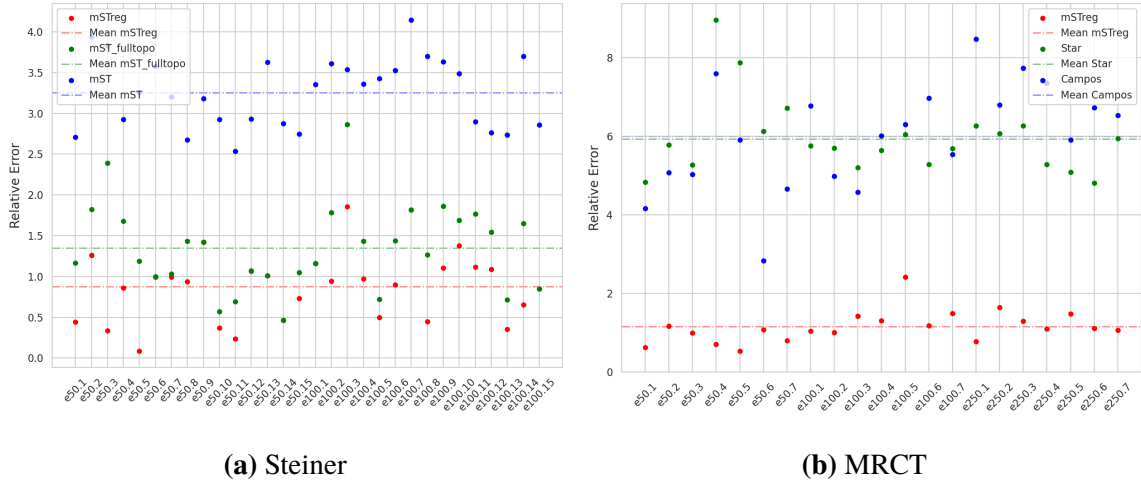


Figure 5.15. Steiner and MRCT OR Library Dataset Benchmark. Relative cost error with respect to a reference cost for the Steiner and MRCT problems for different instances and different methods (lower is better). Left: For the Steiner problem, the reference cost is the optimal cost. mSTreg finds good solutions and improves over mST_fulltopo. Right: For MRCT the reference cost is given by the GRASP_PR algorithm. The heuristic beats all other methods, but is still slightly worse than GRASP_PR algorithm.

as mST_fulltopo). Though our heuristic does not reach the optimal cost, it produces good topologies. The average relative error is lower than 1%.

For the MRCT, we compare our heuristic with the Campos [25] and GRASP_PR [150] algorithms. Campos modifies Prim’s algorithm with heuristic rules, while GRASP_PR conducts local search by exchanging one edge at a time. We test the algorithms on the OR library datasets for problem instances with 50, 100 and 250 terminals. Figure 5.15b shows the relative errors. In this case, we do not have access to the optimal cost, therefore we use GRASP_PR costs cited from Sattari and Didehvar [150] as reference. Campos cost are obtained from our own implementation. For reference, we also show the 2-approximation [177] given by the star graph centered at the data centroid. While mSTreg proves competitive (surpassing Campos but falling short of GRASP_PR by a modest average relative error of 1.16%), it is worth noting that GRASP_PR relies on a time-consuming local search. Leveraging the competitive solution provided by mSTreg as an initial step can enhance the performance and convergence of local search based algorithms, such as GRASP_PR.

5.7.3 Comparing GRASP_PR and mSTreg for the CST Problem

In the previous section, the GRASP_PR algorithm by [150] outperformed the mSTreg heuristic when solving the MRCT problem. However, it’s important to note that each iteration of the GRASP_PR scales quadratically complexity with the number of nodes for complete

graphs due to its local search that swaps edges. In contrast, the complexity per iteration of the mSTreg algorithm is $\mathcal{O}(dn \log(n)^2)$ (refer to AppendixD.9), making it more efficient.

Additionally, GRASP_PR requires an initial random guess to initiate the local search. The quality of this guess impacts the number of iterations needed for the local search to converge. We will show that initializing the local search with a solution generated by the mSTreg allows for initializations that enhance the performance of GRASP_PR. To ensure variability in the mSTreg output, we initialize it with random topologies instead of the mST. Despite this modification, as the mST initialization yielded favorable results, we opt to sample trees similar to it. To achieve this, we construct a tree by randomly sampling edges, prioritizing shorter ones. In concrete, we sample edge (i, j) with probability proportional to $\exp(-\mu \|x_i - x_j\|)$ for some inverse temperature μ . This sampling approach is akin to the one used in Karger’s algorithm for approximating the minimum cut [86, 89] and differs from the GRASP_PR construction phase in that it doesn’t require that one end of the edges belongs to the current tree.

While the previous section relied on the MRCT costs reported in [150] for the GRASP_PR algorithm, we now validate our claims using our implementation of GRASP_PR. Given that the GRASP_PR is a versatile algorithm, we used it to compute the CST with alternative α values besides 1. Acknowledging that for lower α values, the optimum trees will be more similar to the mST, we adapted the construction phase of GRASP_PR. Specifically, for $\alpha < 0.7$, it generates the random tree based on the Karger’s sampling method mentioned above. For $\alpha \geq 0.7$, it uses the construction proposed by [150]. Due to the relatively slow performance of our Python implementation of GRASP_PR, we imposed a 5-minute time threshold. If exceeded, the algorithm returns the best solution at the end of the path relinking phase.

To assess performance, we conducted tests using GRASP_PR, GRASP_PR initialized with mSTreg (GRASP_PR_mSTreg), and the mSTreg algorithms on OR library datasets for problems with 50 and 100 terminals. Analogously to the plots shown in the main paper, Figure5.16 displays the relative errors using GRASP_PR_mSTreg cost as a reference for different α values in the set $\{0.2, 0.4, 0.6, 0.8, 1\}$. Notably, the combination of the mSTreg heuristic with GRASP_PR consistently achieved the lowest cost, as all relative costs are above 0, proving that the mSTreg can enhance GRASP_PR performance.

Across 30 runs (combining 5 α values, 2 problem sizes, and 3 instances), GRASP_PR outperformed mSTreg in only 12 instances. Moreover, achieving a lower cost took minutes for GRASP_PR, while mSTreg run in the order of seconds. Consequently, mSTreg emerges as a favorable alternative, delivering a descent solution quickly.

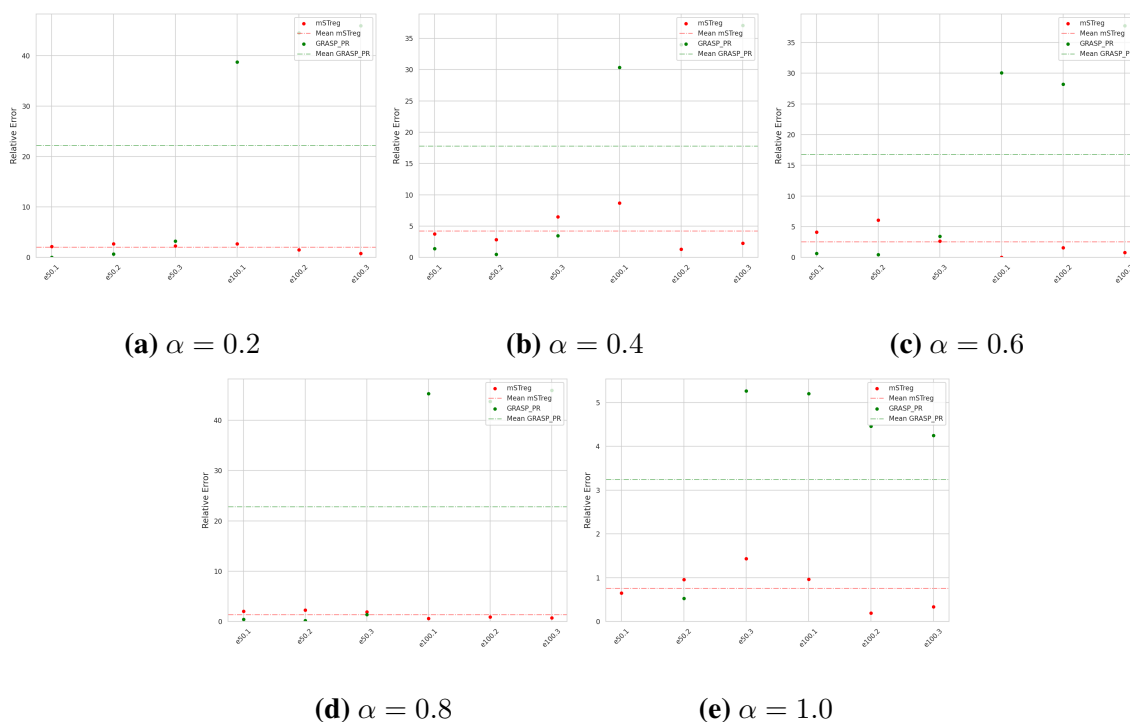


Figure 5.16. Comparing mSTreg and GRASP_PR. Relative cost error concerning the CST problem at different α values in the set 0.2, 0.4, 0.6, 0.8, 1 for various instances of the OR library dataset (lower is better). The comparison involves mSTreg, GRASP_PR, and the combination of GRASP_PR initialized with the mSTreg solution (referred to as GRASP_PR_mSTreg). The relative error is computed using the cost from GRASP_PR_mSTreg as a reference. The combination of the mSTreg heuristic with GRASP_PR consistently achieved the lowest cost, with all relative costs above 0, demonstrating the enhancement of GRASP_PR performance by mSTreg. To manage time constraints, a threshold of 5 minutes was imposed for methods utilizing GRASP_PR.

5.8 Conclusion

This chapter introduced the novel problem of the (branched) central spanning tree, which encompasses the minimum spanning, the Steiner and minimum routing cost trees as limiting cases. The CST weighs the edge-costs with the edge-centralities, whose influence are regulated by the parameter α . We have focused on the Euclidean version of the problem, where the nodes are embedded in an Euclidean space. Moreover, we presented a variant of the problem allowing the addition of extra nodes, referred to as Steiner points (SPs). In addition, we provided empirical evidence for the robustness of the (B)CST tree structure to perturbations in the data, which increases with α . In this regard, α serves as a parameter that trades-off between data-fidelity and stability. We also provided examples of potential applications such as single cell trajectory inference.

On the theoretical side, we showed that as $\alpha \rightarrow \infty$ or $|V| \rightarrow \infty$ with $\alpha > 1$, (B)CST converges to a star-tree, indicating inadequacy in extracting structural information when $\alpha > 1$. Conversely, as $\alpha \rightarrow -\infty$, the (B)CST tends towards a path graph. Additionally, we proved the infeasibility of 4-degree SP when the terminals lie on a plane and $\alpha \in [0, 0.5] \cup \{1\}$ thanks to the closed formulae of the branching angles. We also provided evidence that suggests a similar case for $\alpha \in]0.5, 1[$.

Based on an efficient algorithm to compute the optimal locations of the SPs, we have proposed the mSTreg heuristic, which exploits the optimal position of the SPs and the correspondence between the CST and BCST topologies to find approximate solutions for either. We benchmarked this algorithm and showed its competitiveness on small toy data sets. Since the proposed heuristic is agnostic to the weighting factors that multiply the distances, we leave as future work to test whether it is equally competitive for other problems, like the general optimum communication tree. Another open question is whether the algorithm can be adapted to perform well on non-complete or non-Euclidean graphs.

Chapter 6

BCST-Based Skeletonization with BCST

Plant skeletonization is a fundamental technique for understanding growth patterns, branching hierarchies, and responses to environmental factors in plants. It simplifies intricate plant structures into skeletal representations, often depicted by spanning trees. In this chapter, we illustrate how the Branched Central Spanning Tree (BCST) can accurately model a tree's skeleton using a 3D point cloud of its surface. The BCST spans over all nodes in the point cloud, generating numerous expendable branches that do not significantly contribute to the skeleton structure. To address this, we propose a pruning algorithm based on the Prize Collecting Steiner tree problem, efficiently removing such branches in linear time. Through comparative analysis against existing methods, we demonstrate the competitiveness of the BCST approach.

6.1 Introduction

One-dimensional representations are commonly used to simplify 3D objects, capturing their fundamental shape and geometry. These one-dimensional structures, known as skeletons, have diverse applications in fields such as computer animation [135], shape analysis [113], and object matching [162], among others [149, 163]. Due to their versatility, various approaches have been developed to extract skeletons from 3D objects, including meshes or point clouds.

Within the field of botany, the significance of 3D plant skeletonization has been growing with the advancement in technologies like light detection and ranging (LiDAR). Such technologies enable the efficient capture and generation of large point clouds representing plants. The skeletons extracted can later be used in plant phenotyping, plant morphology or plant virtual modeling among other applications. These application aid researchers to understand and study the traits and growth patterns of plants.

The skeletonization of plants presents persistent challenges, primarily attributed to fac-

tors like noise and occlusions during the scanning process. The intricate structure of trees, characterized by multiple thin branches, further complicates the extraction of an accurate skeleton that faithfully models all branches. Common approaches to skeleton extraction are geometric-based. The typical pipeline of such approaches involves constructing a neighbor graph that spans over the points, such as the k -nearest neighbor graph. From this, a geodesic graph is built, connecting near points starting at a root node. Based on the geodesic graph, node clustering occurs, defining the final skeleton tree by connecting representatives of these clusters [46, 182]. In the presence of occlusions, the neighbor graph may be suboptimal, leading to potential errors in the final skeleton.

Alternative methods include thinning processes that erode or contract points on the surface until a thin structure is obtained [132]. Some methods leverage eigenvectors to define the dominant directions of the trunk and branches [3], while others utilize voxelization. Yet, these methods face limitations, leading researchers to develop refinement procedures to enhance the quality of tree structures [33, 197]. Other approaches enhance the skeleton by incorporating additional attributes, such as semantic segmentation of the point cloud [118, 132]. For a more comprehensive overview of the field, we recommend consulting the survey [27] and references therein.

While some of the current approaches rely on intricate pipelines with multiple parameters that can be challenging to use, we propose a skeletonization method based on the Branched Central Spanning Tree (BCST). As discussed in Chapter 5, the BCST proves to be an appropriate tree to model the skeleton of objects due to its geometric robustness against perturbations such as noise and its ability to condense the main structure of the point cloud. As the parameter α of the BCST increases, the tree tends to a star-tree, resulting in a more stable and robust structure. By adjusting the parameter α we can regulate the level of stability and the connection of spatially close nodes, effectively capturing the original structure of the tree. Consequently, the BCST strikes a good balance between stability and data fidelity that is ideal for describing the skeleton of a 3D object. Furthermore, the inclusion of Steiner points provides greater flexibility, allowing them to be positioned in the volume's center and ensuring that the BCST skeleton traverses this central region—a feature often desirable in object skeletonization.

While the BCST spans over all nodes in the point cloud, allowing for the description of all branches, including the thinner and more challenging ones, it also generates many spurious and expendable branches that do not contribute to the portrayal of the skeleton structure (see Figures 6.1b and 6.3). Consequently, it becomes necessary to prune such branches. We propose a pruning algorithm capable of removing these branches. To do so, we assign two values to each edge. Firstly, we define a quantity referred to as explainability profit, which measures the number of potential nodes that can be explained by the edge.

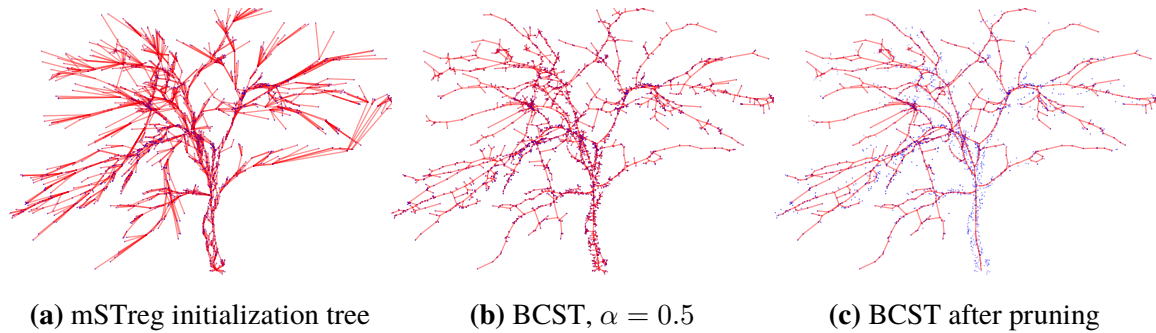


Figure 6.1. BCST-Based Skeletonization Steps. 6.1a) The tree, which includes all shortest paths from the root node to the rest of the terminals, is used as the initialization of the mSTreg heuristic (see Section 5.6.2). 6.1b) The BCST spans over all nodes in the point cloud, allowing for the description of all branches, but also generating many spurious branches that need to be pruned. 6.1c) The BCST is pruned using the prize-collecting Steiner tree-based approach described in Section 6.2.

Intuitively, the longer is the branch to which the edge belongs, the higher its explainability profit. Since long branches tend to be actual branches of the skeleton, the pruned tree should have branches with high explainability. Secondly, we define an additional quantity known as bending energy, which is larger for edges with significant curvature. Spurious branches tend to be more curved as they branch at sharper angles.

The pruning process involves optimizing the overall explainability profit while minimizing the bending energy. This optimization problem is formulated as a Prize Collecting Steiner tree problem, aiming to encourage the presence of longer branches while penalizing short, highly curved branches. Notably, this problem can be efficiently solved in linear time.

We compare the performance of the BCST skeletonization method against different skeletonization methods from the literature. Our results demonstrate its superior performance under diverse experimental settings, simulating the presence of noise, occlusions, and varying density levels. Overall the BCST is able to model most of the branches, including the thinner ones, particularly in low-density scenarios where other methods struggle.

In summary, our contributions consist of: 1) the application of the BCST for 3D plant skeletonization; 2) the introduction of an efficient pruning algorithm, leveraging the Prize Collecting Steiner approach, to eliminate expendable branches generated by the BCST (Section 6.2); and 3) a comprehensive comparison showcasing the competitiveness of our method, especially in low-density scenarios (Section 6.3).

6.2 Skeletonization Method

This section describes the application of the BCST method for extracting the skeleton of a 3D object, particularly focusing on a 3D point cloud representing a plant. Let $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^3$ denote the set of sampled points on the object's surface, from which the skeleton is to be extracted. Given that the BCST generates a spanning tree encompassing all points in X , it requires pruning. Section 6.2.1 defines the optimization problem that characterizes the pruned tree. In Section 6.2.2, we will reformulate this problem as a Prize Collecting Steiner Tree problem, which aims to find the most cost-effective subgraph that connects a specified set of terminal nodes in a graph while maximizing the sum of collected prizes associated with the nodes. Section 6.2.3 will delve into implementation specifics, including parameter selection.

6.2.1 Pruning Optimization Problem: Formulating the Objective

The BCST defines a spanning tree by connecting all terminals through the addition of Steiner points. In consequence, the BCST encompasses the original point cloud along with the added Steiner points, resulting in numerous extraneous branches, each extending to individual data points within the point cloud. We propose a method to prune such branches by assigning a signed weight to each edge based on its curvature (bending energy) and the proportion of data that can be explained by the edge (explainability profit). The final pruned tree is defined as the subtree which maximizes the sum of the edge weights.

In a more intuitive sense, we identify branches as spurious if they are short and connect to terminal nodes. These branches often split at sharp angles, resulting in a high bending energy. The bending energy of a smooth curve indicates the curve's flexibility or stiffness, formally defined as the integral over the curve of its squared curvature. Specifically, the bending energy is expressed in terms of binormal curvature. In our context, we calculate the discretized bending energy [17, 160]. Let $\wp = \{v_1, v_2, \dots, v_\ell\}$ be a path of nodes embedded in \mathbb{R}^3 that collectively model a curve. The discrete binormal curvature at node $v_i \in \wp$ is defined as:

$$\kappa_{v_i} := 2 \frac{(x_{v_i} - x_{v_{i-1}}) \times (x_{v_{i+1}} - x_{v_i})}{\|x_{v_i} - x_{v_{i-1}}\| \cdot \|x_{v_{i+1}} - x_{v_i}\| + \langle x_{v_i} - x_{v_{i-1}}, x_{v_{i+1}} - x_{v_i} \rangle} \quad (6.1)$$

Here, $x_i \in \mathbb{R}^3$ indicates the position of node v_i . Operations \times and $\langle \cdot, \cdot \rangle$ represent the exterior and interior product, respectively. The work by Stuhmer et al. [160] uses the discretized binormal curvature in order to compute the contribution of each node to the total bending energy of the curve. In contrast, our goal is to assign the contribution to each edge in the BCST. First, we note that binormal curvature at a given point is influenced not only by the node's position but also by the positions of the nodes preceding and succeeding it along

the path. To establish this directionality, we designate a predecessor for each node in the BCST by transforming the tree into a directed one through the selection of a root node. The edge's direction is determined by the direction of the paths connecting the root node to the remaining nodes. We denote the predecessor of a node u as $\text{pred}(u)$. For an edge e , we refer to the head of e , $h(e)$, as the outgoing node and the tail of e , $t(e)$, as the ingoing node. The bending energy of an edge is defined as:

$$E_b(e; \mathbf{T}) := 2 \frac{\|\kappa_{h(e)}\|^2}{\|x_{\text{pred}(h(e))} - x_{h(e)}\| + \|x_{t(e)} - x_{h(e)}\|} \quad (6.2)$$

In words, (6.2) calculates the bending energy at edge $e = (h(e), t(e))$ by dividing the squared norm of the binormal curvature at the head of e by the average length of e and its predecessor edge given by $\text{pred}(e) = (\text{pred}(h(e)), h(e))$. Notably, the bending energy increases with the curvature. When the edge and its predecessor are well-aligned, indicating they point in a similar direction, the curvature is lower, resulting in lower bending energy. Consequently, edges belonging to a straight branch of a tree are likely to have low bending energy.

While spurious branches bifurcate at tight angles, the actual branches we aim to preserve may also bifurcate sharply; however, the latter branches tend to be longer. To retain edges that might have high bending energy but indicate the start of a long branch, we introduce a second value for each edge e , referred to as explainability profit and denoted by $E_p(e)$. This quantity measures indirectly the potential length of the branch associated to an edge and, consequently, the amount of data that can be explained by the edge. Formally, for an edge e , the explainability profit $E_p(e)$ is equal to the normalized shortest path distance of the furthest reachable leaf node, where the reachability is determined by the directions of the edges. We normalize the shortest path distance by dividing by the diameter of the tree. Formally, the explainability profit at e is defined as

$$E_p(e; \mathbf{T}) = \max_{\ell \in R(h(e))} \frac{d_{\mathbf{T}}(h(e), \ell)}{\max_{(u,v)} d_{\mathbf{T}}(u, v)}, \quad (6.3)$$

where $R(u)$ denotes the set of reachable leaves from u and $d_{\mathbf{T}}(\cdot, \cdot)$ is the shortest path distance within the tree \mathbf{T} . Intuitively, the farther the leaf with respect to the head of e , the more costly is the exclusion of e from the pruned tree, as that leaf will only be part of the pruned tree if and only if e is included.

We want to penalize edges which have high bending energy, associated with a high curvature, while we want to incentivize the presence of edges with high explainability profit. Given $\lambda > 0$, we assign a weight $w(e) = E_p(e; \mathbf{T}) - \lambda E_b(e; \mathbf{T})$ to each edge $e \in \mathbf{T}$ that balances the penalty and reward given by these values. In consequence, given \mathbf{T} and $\lambda > 0$, the pruned tree $\mathbf{T}_p \subset \mathbf{T}$ is defined as the subtree of \mathbf{T} , containing the root node, that maximizes the following profit function:

$$\text{Profit}(\mathbf{T}_p) = \sum_{e \in \mathbf{T}_p} w(e) = \sum_{e \in \mathbf{T}_p} E_p(e; \mathbf{T}) - \lambda E_b(e; \mathbf{T}). \quad (6.4)$$

The objective of this function is to favor the inclusion of edges with high explainability profit and low bending energy in the pruned tree.

6.2.2 Pruning Algorithm Based on the Prize Collecting Steiner Tree Problem

We can reformulate the optimization problem given by (6.4) as a Prize Collecting Steiner tree (PCST) problem [87, 111]. The classic Steiner Tree problem over a graph $G = (V_G, E_G)$ with positive edge-costs $c(e)$ aims to find the tree with minimum cost that connects a subset of nodes $U \subset V_G$. Contrary to the Euclidean Steiner Tree, the former does not allow the inclusion of Steiner points, but can only connect nodes through the edges of G . Formally, it optimizes the following problem

$$\min_{\substack{T \subset G \\ U \subset V_T}} \sum_{e \in E_T} c(e).$$

Note that when $U = V_G$, the Steiner Tree over G is equivalent to the minimum spanning tree, while when $|U| = 2$ it reduces to finding the shortest path between the two terminals.

In contrast to the classic Steiner tree problem in graphs, the prize collecting variant considers, in addition to the edge-costs, also weight profits, $p(v) > 0$, associated to the nodes of G . In opposition to the edge-costs, which are minimized, the profit weights are maximized. Accordingly, the PCST objective function is

$$\max_{T \subset G} \sum_{v \in V_T} p(v) - \sum_{e \in E_T} c(e). \quad (6.5)$$

Note that PCST does not require indicating the terminals that must be connected as these are implicitly indicated by the profit weights.

Our objective, as defined in equation (6.4), maximizes a sum over positive and negative edge-weights. Consider the connected components which are strictly connected by edges with positive weight. Clearly, if any node within a positive connected component is included in the optimal subtree, then the entire component will also be part of the optimal tree. This is because the remaining nodes can be connected through positive edges, increasing the overall profit. Thus, without loss of generality, we can merge all nodes belonging to the same positive connected component, C^+ , into a single node. Furthermore, we assign a profit to this merged node, defined as the sum of weights of the positive edges within the component, expressed as

$$p(C^+) = \sum_{u,v \in C^+} w(u,v).$$

This profit reflects the benefit gained when C^+ is included in the subtree.

This merging operation generates a new graph, where each node has a profit and all edges are negatively weighted. Thus, we obtain the objective of the PCST problem

$$\begin{aligned} \max_{T_p \subset T} \text{Profit}(T_p) &= \max_{T_p \subset T} \sum_{e \in T_p} w(e) = \max_{T_p \subset T} \sum_{C^+ \in \mathcal{C}^+} \sum_{\substack{u, v \in C^+ \\ (u, v) \in T_p}} w(u, v) + \sum_{\substack{e \in T_p \\ w_e < 0}} w(e) \\ &= \max_{T_p \subset T} \sum_{C^+ \in \mathcal{C}^+} p(C^+) - \sum_{\substack{e \in T_p \\ w_e < 0}} |w(e)| \end{aligned} \quad (6.6)$$

where \mathcal{C} is the set of the positive connected components of T . Figure 6.2 illustrates how the original tree with positive and negative edge weights can be transformed to a PCST setting.

In general, solving the PCST problem is NP-hard. However, in scenarios where the underlying graph is a tree, such as in our case, an efficient solution exists. This solution can be achieved in linear time using dynamic programming, as outlined by Klau et al. in [96].

Let r be the root of the tree T . For each node $v \neq r \in T$, we define $\text{pred}(v)$ as the node that precedes v in the path from r to v and $C(v) = \{u : \text{pred}(v) = u\}$ as the set of children of v . We recursively define a value $\ell(v)$ for each node v in T

$$\ell(v) := p(v) + \sum_{u \in C(v)} \max(0, \ell(u) - |w(u, v)|) \quad (6.7)$$

The value $\ell(v)$ is the profit of the subtree $T_{V(v)} \subset T$ induced by the nodes in $V(v)$, where

$$V(v) := \{v\} \cup \{V(u) : \ell(u) > |w(u, v)|\}. \quad (6.8)$$

It can be shown by induction that $T_{V(v)}$ is the tree that maximizes the profit constrained to the subtrees of the tree induced by the descendants of v , that is the subtree with root v consisting of all vertices and edges reachable from v without passing the vertex $\text{pred}(v)$. The edge (u, v) , with $\text{pred}(u) = v$, will be included in the optimal subtree if $w(u, v) > \ell(v)$. Otherwise it does not pay off to include the edge, as it incurs a higher cost than the profit obtained through the connection to u . Hence, given the root node r , the tree that maximizes (6.6) is attained at $T_{V(r)}$ with $\text{Profit}(T_{V(r)}) = \ell(r)$.

Figure 6.2 provides a visual representation of the transformation of our problem into the PCST problem. It also outlines the sequential steps to calculate the values $\ell(\cdot)$ for each node in order to determine the optimal subtree. Pseudocode 3 offers a comprehensive description of the pruning algorithm.

Figure 6.3 illustrates the impact on the pruning process across various values of the parameter λ . This parameter regulates the balance between the bending energy and the explainability profit of the edges, as defined in (6.4). When λ is set to 0, no pruning is performed, as none of the edges has a penalization weight. In this case, we retrieve the original BCST that spans all input points. As λ increases, more edges are pruned. Notably,

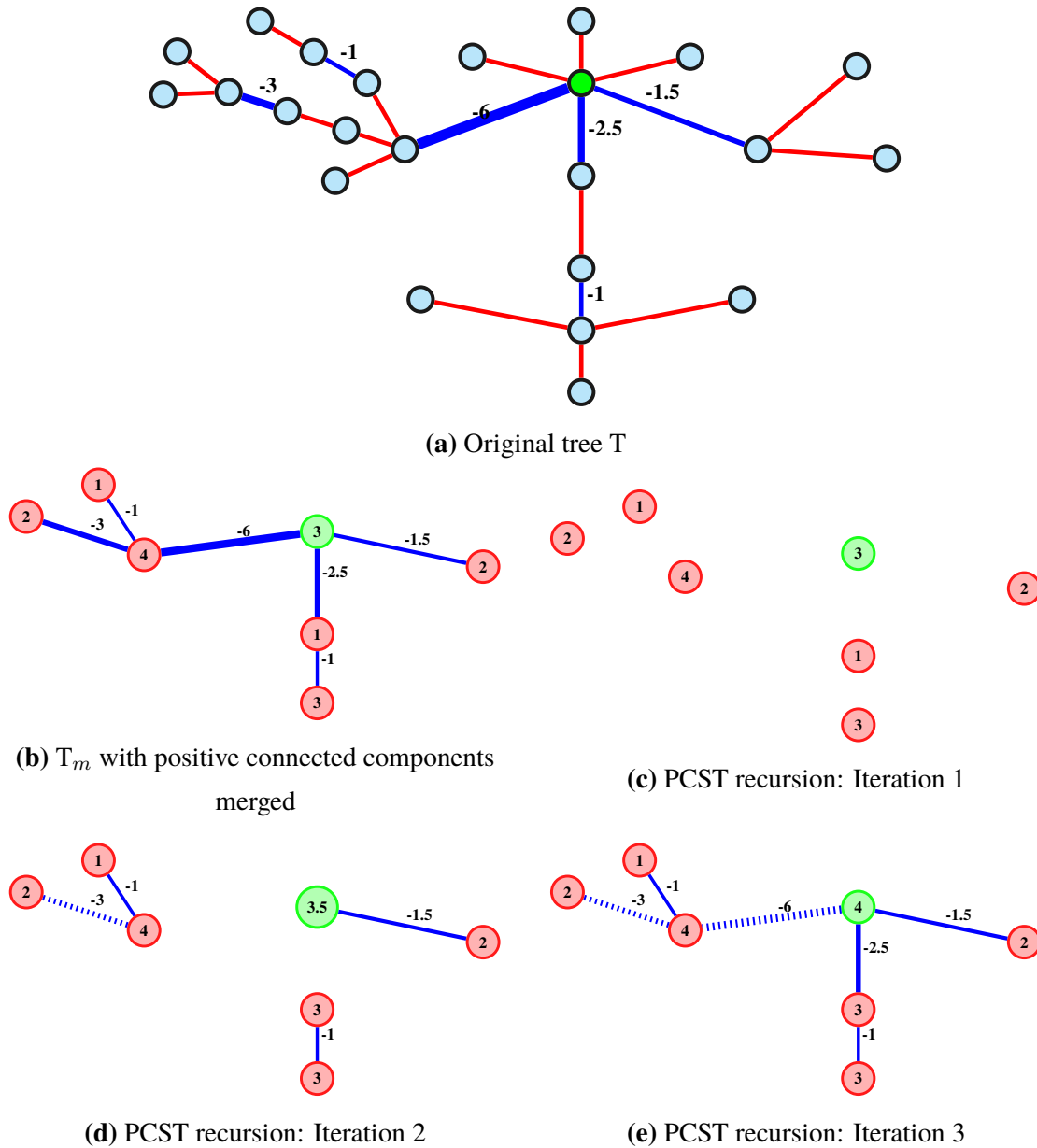


Figure 6.2. BCST Skeletonization Pruning Algorithm. 6.2a) Tree, T , with red edges signifying positive weights (uniformly set to 1) and blue edges negative weights. The green node represents the root node. 6.2b) Tree once the positive edges of T have been contracted. Maximizing (6.4) over T is equivalent to solving the Prize Collecting Steiner Tree (PCST) (6.5) problem over T_m (6.5). Node profits in T_m , indicated inside each node, equal the sum of the weights of the corresponding contracted positive edges. 6.2c-6.2e) illustrate the recursive iterations to compute values $\ell(\cdot)$ at each node, as defined in (6.7). The value inside each node represents the ℓ value at that iteration. Edges appear as visited, with dashed lines indicating edges more costly than the ℓ profit value of the connected node, suggesting their exclusion from the optimal subtree. 6.2e) showcases the optimal subtree formed by non-dashed edges with profit equal to 4 as indicated by the ℓ value of the root.

for $\lambda = 0.01$, most spurious branches have been pruned, and at $\lambda = 0.05$, none of them remains. At higher values, real branches are also pruned. In the experiments section, we consistently set $\lambda = 0.05$ for all our experiments, unless otherwise stated, as it has yielded satisfactory results.

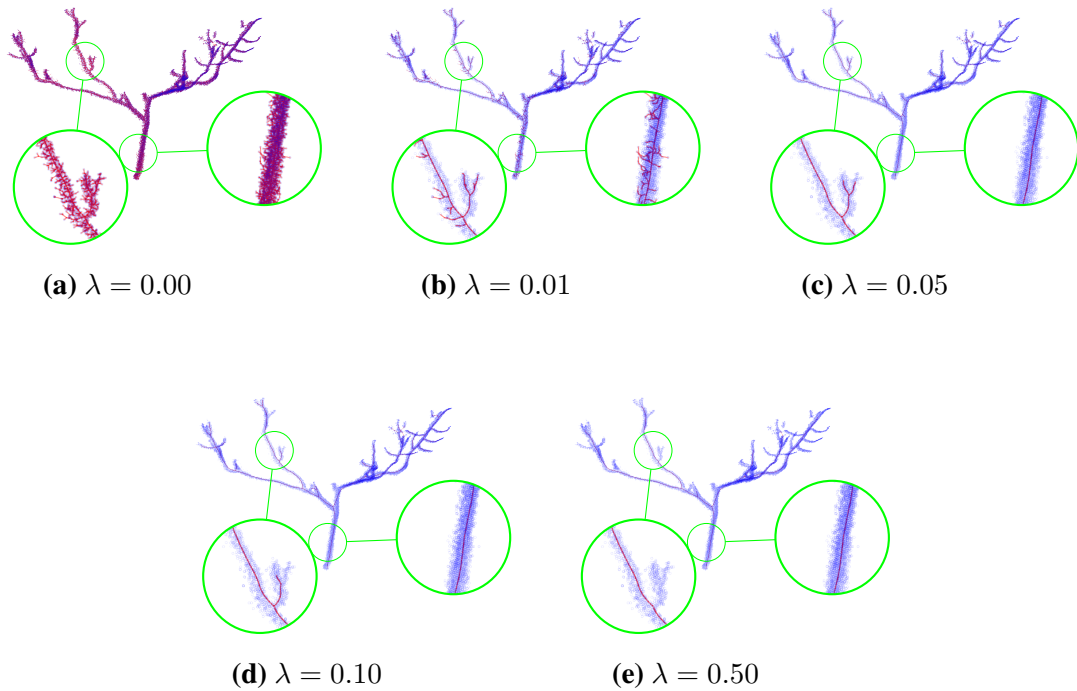


Figure 6.3. Effect of the λ Weighting Factor on Balancing the Explainability Profit and the Bending Energy in Equation (6.4). When $\lambda = 0$, no pruning occurs, resulting in the original BCST spanning all data points. As λ increases, spurious branches connecting to the data points are pruned.

6.2.3 Implementation Details

To compute the BCST and subsequently apply the pruning algorithm described in 3, it is essential to determine specific parameters and settings. These include the α parameter governing the BCST tree, the initial topology for the mSTreg heuristic used to approximate the BCST, the weighting factor λ for the profit function (6.6), and the root node determining the bending energy (6.2) and consequently the pruned tree. This section describes how to set these parameters.

In the context of a 3D point cloud representing a plant or a tree, a common choice for the root node is the one located at the bottom of the trunk—specifically, the node with the lowest z -axis value. While similar approaches have been used in other skeletonization methods,

Algorithm 3: Skeletonization Pruning Algorithm

Data: Tree graph T , λ , root
Result: Pruned tree

```

/* Compute edge weights */
1 foreach edge  $e$  in  $T$  do
2   |  $w(e) = E_p(e) - \lambda E_b(e)$ ;
3 end

/* Merge positive components and assign profit  $p$  */
4  $T_{merged} = T$ ;
5 foreach  $C^+ \in \mathcal{C}^+$  do
6   |  $C^+ = \emptyset, p(C^+) = 0$ ;
7   | foreach  $(u, v) \in E(T_{C^+})$  do
8     |  $p(C^+) = p(C^+) + w(u, v)$ ;
9     |  $T_{merged} = \text{Contract\_edge}(u, v, T_{merged}), C^+ = C^+ \cup \{u, v\}$ 
10  | end
11  | if root  $\in C^+$  then  $C_r^+ = C^+$ ;
12 end

/* Initialize variables */
13  $Q = []$ ;
14 foreach  $C^+ \in T_{merged}$  do
15   |  $\ell[C^+] = p[C^+], V[C^+] = \{C^+\}, d[C^+] = \text{deg}_{T_{merged}}(C^+)$ ;
16   | if  $\text{deg}_{T_{merged}}(C^+) == 1$  then  $Q.add(C^+)$ ;
17 end
18  $\text{pred} = \text{DFS}(C_r^+, T_{merged})$ ;

/* Compute maximum subtree */
19 while  $Q$  not empty do
20   |  $C_1^+ = Q.pop(), C_2^+ = \text{pred}[C_1^+]$ ;
21   |  $\ell[C_2^+] = \ell[C_2^+] + \max(0, \ell[C_1^+] + w(C_1^+, C_2^+))$ ;
22   | if  $\ell[C_1^+] + w(C_1^+, C_2^+) \geq 0$  then  $V[C_2^+] = V[C_2^+] \cup V[C_1^+]$ ;
23   |  $d[C_2^+] = d[C_2^+] - 1$ ;
24   | if  $d[C_2^+] == 1$  and  $C_2^+ \neq C_r^+$  then  $Q.add(C_2^+)$ ;
25 end

26 return  $T_{V(C_r^+)}$ ;

```

such as [46], our method takes a slightly different approach. Instead of directly selecting the lowest z-axis point as the root, we introduce an artificial root node to the point cloud. This artificial root is defined as the average of all points with z-axis values below the 2nd percentile of the z-coordinate values. Using this average biases the root node to be closer to the center of the trunk, in contrast to being on the surface of the trunk, as would be the case if one of the original points had been selected as the root. After determining this average point, its z-coordinate is then set to that of the bottommost point.

In our setting, the root node serves two purposes. Firstly, it determines the edge directions for computing the bending energy. Secondly, it is used to define the initial topology for the mSTreg heuristic, which approximates the BCST. While mSTreg typically uses the minimum spanning tree as its default topology, we deviate by computing a geodesic graph starting at the root node. To construct this geodesic graph, we build a connected k-nearest neighbor (kNN) graph spanning all input nodes and the artificial root node. To ensure connectivity, we initialize k to $\log(N)$, where N is the total number of input points. If the kNN graph is not initially connected, we double the value of k until connectivity is achieved. Once the connected kNN graph is established, the initial topology of the mSTreg heuristic is set as the shortest path tree rooted at the artificial root. This tree comprises all the shortest paths in the kNN graph from the root node to the remaining nodes. Figure 6.1 illustrates the initialization tree alongside the BCST tree and the pruned BCST.

Additionally, we observed that BCST-modeled skeleton is improved by increasing the point density near the root through virtual point replication. Instead of adding a single root, we add five times as many virtual points at the root location as there are original points. In practice, this involves weighting the incident edges to the root as if there were replicated collapsed terminal points at the root location.

We consistently set the exponent α regulating the impact of betweenness centrality in BCST to $\alpha = 0.5$, as it has proven effective. Similarly, unless otherwise stated, we set the weighting factor λ of the profit function (6.6) to 0.05, which has worked well in practice as illustrated in Figure 6.3. However, this parameter may require fine-tuning in other scenarios. For example, in the presence of data noise, a higher value may be necessary to remove spurious branches, as observed in Figures 6.6 and E.5.

6.3 Experiments

To assess the effectiveness of our proposal, we conducted tests using synthetic data from the smart tree dataset [48].¹ This dataset comprises point clouds simulating the surfaces of six tree types (eucalyptus, ginkgo, walnut, apple, pine and cherry trees) along with their

¹<https://github.com/uc-vision/synthetic-trees>

corresponding ground truth skeletons. We evaluate our method from the validation dataset, featuring 10 samples per each tree type (totaling 60 point clouds). In the following sections we describe the metrics employed to quantitatively assess the quality of our method, alongside detailing the experiments conducted and their corresponding outcomes.

6.3.1 Metrics

To quantitatively measure the quality of the inferred skeleton, T , we compare it with the ground truth skeleton, T_{gt} , using the following metrics:

- **Hausdorff distance (HD):** This metric identifies the skeletons as sets of points. It quantifies dissimilarity by finding the farthest point in one set to its nearest neighbor in the other set. Formally,

$$d_{HD}(T, T_{gt}) = \max \left(\max_{x \in T} \min_{y \in T_{gt}} \|x - y\|, \max_{x \in T_{gt}} \min_{y \in T} \|x - y\| \right).$$

The Hausdorff distance value is determined by the single pair of points with the maximal error, emphasizing the maximum separation between two sets.

- **Chamfer distance (CHD):** Similarly to the Hausdorff distance, it also identifies the skeletons as point clouds. It measures the dissimilarity by computing the average Euclidean distances between the points in one set to their nearest neighbors in the other set. Formally,

$$d_{CHD}(T, T_{gt}) = \frac{1}{2} \left(\frac{1}{|T|} \sum_{x \in T} \min_{y \in T_{gt}} \|x - y\| + \frac{1}{|T_{gt}|} \sum_{x \in T_{gt}} \min_{y \in T} \|x - y\| \right).$$

In contrast to the Hausdorff distance, which focuses on the maximum separation between the sets, the Chamfer distance considers the average separation.

- **Shortest path matching (SPmatch):** Contrary to the previous metrics, the shortest path matching metric interprets the skeletons as graphs, taking into account the connections between the nodes. Specifically, it quantifies the difference of the shortest path distances between pairs of points in one tree, say x and y in T , and their corresponding nearest neighbors in the other tree, represented by $m_{T \rightarrow T_{gt}}(x)$ and $m_{T \rightarrow T_{gt}}(y)$. Here $m_{T \rightarrow T_{gt}}(x) := \arg \min_{y \in T_{gt}} \|x - y\|$, and similarly, we define $m_{T_{gt} \rightarrow T}$. The shortest path matching is formally defined as

$$\begin{aligned} d_{SP}(T, T_{gt}) = & \frac{1}{2|T|} \sum_{x, y \in T} |D_T(x, y) - D_{T_{gt}}(m_{T \rightarrow T_{gt}}(x), m_{T \rightarrow T_{gt}}(y))| \\ & + \frac{1}{2|T_{gt}|} \sum_{x, y \in T_{gt}} |D_T(m_{T_{gt} \rightarrow T}(x), m_{T_{gt} \rightarrow T}(y)) - D_{T_{gt}}(x, y)|, \end{aligned} \quad (6.9)$$

where $D_T(\cdot, \cdot)$ and $D_{T_{gt}}(\cdot, \cdot)$ correspond to the shortest path distances in T and T_{gt} respectively.

6.3.2 Experimental Details and Results

We compare the BCST method against the Pypetree [46], L1-skeleton (L1skel) [79] and the Laplacian based contraction (LBC) [26] skeletonization methods. We have analyzed the performance of the various methods under different settings, namely point clouds with different density levels, noisy point clouds and point clouds with missing data. For each of the 60 point clouds of the validation subset of the smart tree dataset we compute the skeleton using each of the aforementioned methods and report the average metric scores.

N	BCST_pruned	L1skel	LBC	Pypetree
HD				
2500	1.21±1.34	2.31±2.31	3.46±3.54	2.21±2.53
5000	1.05±1.15	1.81±1.81	2.68±2.73	1.69±1.84
10000	0.94±1.02	1.60±1.63	2.13±2.21	1.34±1.35
20000	0.87±0.96	1.43±1.48	1.67±2.75	1.18±1.23
CHD				
2500	0.18±0.15	0.55±0.57	1.16±1.28	0.46±0.44
5000	0.15±0.12	0.41±0.42	0.74±0.88	0.34±0.34
10000	0.13±0.10	0.33±0.33	0.46±0.56	0.27±0.26
20000	0.11±0.09	0.26±0.26	0.31±0.34	0.22±0.22
SPmatch				
2500	0.78 ± 0.61	1.49±1.34	1.92±1.74	1.17±1.31
5000	0.73 ± 0.56	1.44±1.41	1.38±1.28	1.06±1.20
10000	0.67 ± 0.52	1.70±2.03	1.00±0.86	0.90±0.87
20000	0.65 ± 0.54	1.34±1.72	0.87±0.86	0.79±0.61

Table 6.1. Comparison of Skeletonization Methods Across Varying Density Levels. Average performance of different skeletonization methods across varying sample sizes (N), measured by HD, CHD, and SPmatch metrics, where lower values indicate superior performance. BCST_pruned consistently outperforms other algorithms across all metrics and sample sizes.

Point clouds with different density levels: For each point cloud we compute the skeletons at different levels of densities by considering uniformly sampled points of the original point clouds with $n \in \{2500, 5000, 10000, 20000\}$ points. Table 6.1 shows the average HD, CHD and SPmatch metric values over the 60 point clouds. Our method consistently outperforms

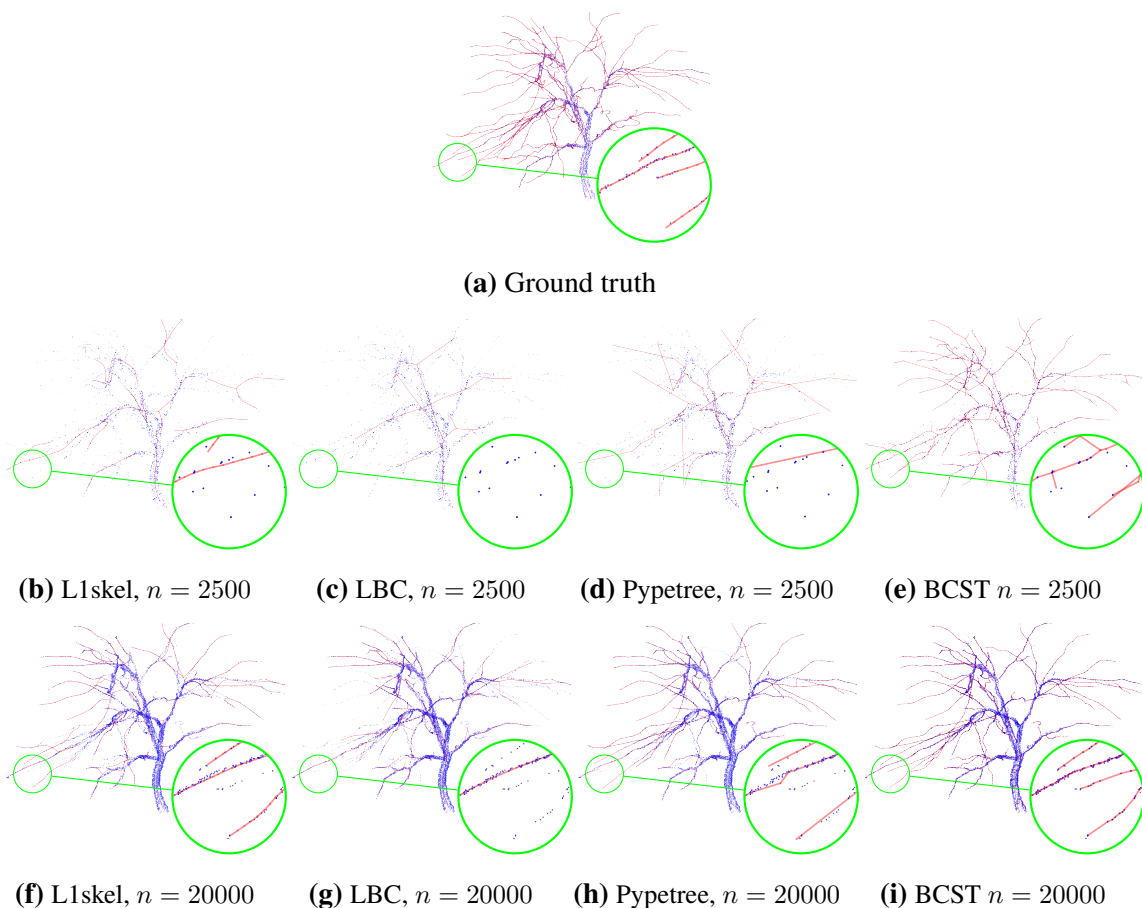


Figure 6.4. Performance Comparison Skeletons at Different Density Levels (Apple Tree). Comparative analysis of skeletonization methods applied to different density levels of an apple tree instance. Each column corresponds to a distinct technique, while each row represents varying density levels. Notably, BCST outperforms other methods by accurately modeling a majority of branches, particularly evident at lower densities (6.4e) where the other methods miss most of the branches (6.4b-6.4d).

other algorithms, demonstrating superior performance across different metrics and varying levels of density.

Figure 6.4 illustrates qualitative results of the skeletons at different density levels for an apple tree. At lower density levels LBC, L1-skeleton and Pypetree miss many more branches than our proposal, which shows that the BCST method is more robust to the number of samples. As the number of samples increases the skeletons are qualitatively comparable. Our proposed pruning algorithm effectively removes most of the spurious branches while preserving the actual branches. However, a few expendable branches can be found in the tree trunk, a pattern consistent across different experimental settings. Further fine-tuning of the λ parameter, which was fixed for all instances, may address this issue. Appendix E.1 provides further qualitative results.

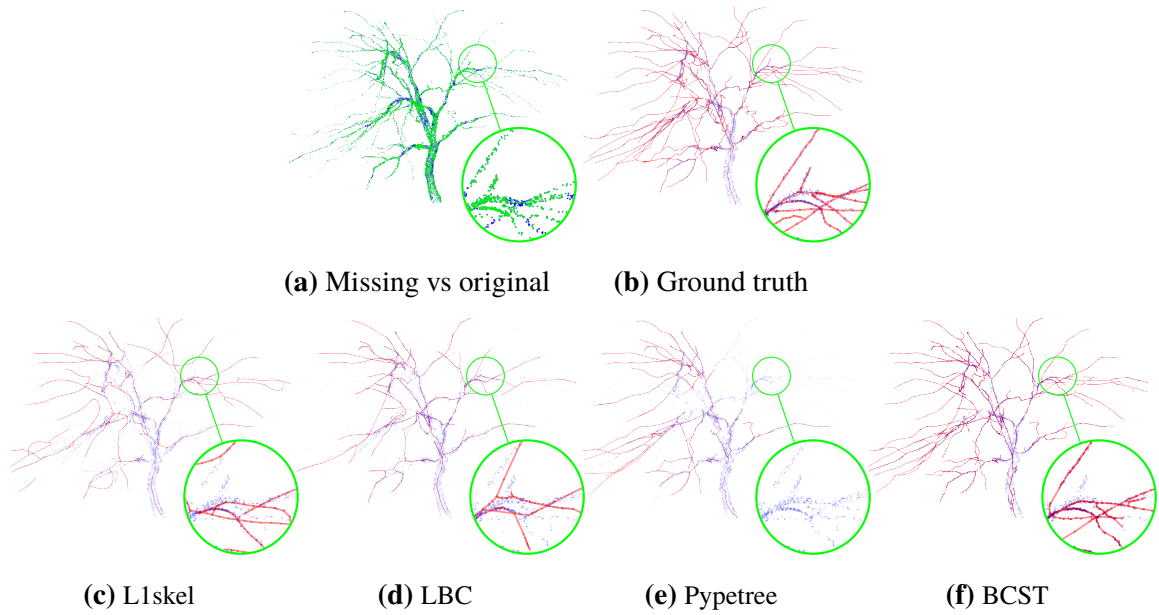


Figure 6.5. Performance Comparison Skeletons with Missing Data (Apple Tree). Comparative analysis of skeletonization methods applied to an apple tree instance with 20% missing data. The green points in 6.5a) represent the input data points, while the blue points correspond to the 20% of data points intentionally removed from the original sample with $n = 20000$. BCST demonstrates superior performance compared to other methods, accurately modeling the majority of branches even in the presence of 20% missing data.

	BCST_pruned	L1skel	LBC	pypetree
HD	1.01 ± 0.98	1.46 ± 1.48	1.73 ± 1.80	3.12 ± 3.70
SPmatch	0.83 ± 0.67	1.48 ± 1.74	1.20 ± 1.21	1.59 ± 1.65
CHD	0.14 ± 0.10	0.29 ± 0.29	0.33 ± 0.38	0.65 ± 0.84

Table 6.2. Comparison of Skeletonization Methods with Missing Data. Average performance of different skeletonization methods under 20% data absence, measured by HD, CHD, and SPmatch metrics, where lower values indicate superior performance. BCST_pruned consistently outperforms other algorithms, demonstrating superior performance in the presence of data occlusions.

Point clouds with missing data: For each point cloud we remove approximately 20% of the points by sampling uniformly at random points from the point cloud and building occlusion balls with variable radius centered at the sampled points. The points lying within the balls are removed. The radius of the ball at each sampled point, v , is set to be the q -th percentile of the sorted distances from v to the rest of points. The percentile is determined by sampling a value from an exponential distribution with mean equal to 0.001,

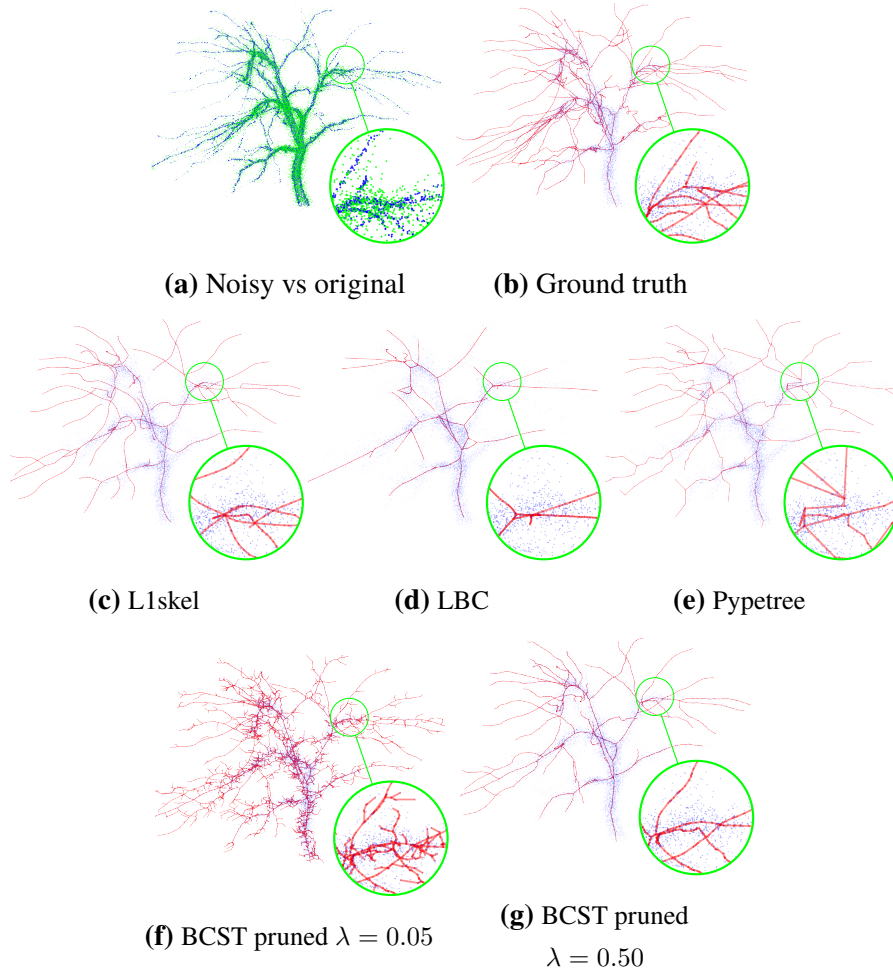


Figure 6.6. Performance Comparison Skeletons with Noisy Data (Apple Tree). Comparative analysis of skeletonization methods on an apple tree instance affected by Gaussian noise. The green points in 6.6a) represent the input data points post-application of Gaussian noise, while the blue points depict the original data points without any noise. The default weighting factor ($\lambda = 0.05$) used for pruning keeps several spurious branches. This issue can be addressed by increasing λ , as demonstrated in 6.6g).

i.e. $q \sim \text{Exp}(0.001)$. In addition, we constrain q to be in the set $[0.0001, 0.005]$ in order to bound implicitly the radius of the ball. For this experiment we consider a fixed density level with 20000 number of points. After the removal of the 20% of points, the point cloud has around 16000 samples.

Table 6.2 provides the average and standard deviation for each of the metrics over all point clouds with missing data. Analogously, to the two previous experiment settings BCST outperforms the rest of methods while having a lower decrement in performance in comparison to the performance on the unperturbed sample.

Figure 6.5 presents qualitative results for an apple tree. The BCST modeling of the skeleton demonstrates greater robustness to occlusions, resulting in more accurate representation

of the branches. This distinction becomes particularly evident when examining thinner and shorter branches, which are better preserved in the BCST case. Figure E.4 in Appendix E.1 shows the skeleton of a pine tree under the same experimental setting.

Noisy point clouds: For each point cloud, we have added Gaussian noise to each point. The variance of the Gaussian is set as the average distance to the 15th neighbor of each node. As we did for the point clouds with missing data, we consider a fixed density level with 20000 number of points. Table 6.3 provides the average and standard deviation for each of the metrics over all point clouds with noisy data. The pruned BCST skeleton obtains better score values than the rest of methods. Moreover, the decrement in performance of the BCST method with respect to the unperturbed sample is lower than the one observed for the other methods.

	BCST_pruned	L1skel	LBC	Pypetree
HD	0.81 ± 0.92	1.51 ± 1.64	2.32 ± 2.55	1.48 ± 1.59
SPmatch	0.75 ± 0.59	2.08 ± 3.10	1.30 ± 1.18	1.53 ± 1.51
CHD	0.17 ± 0.15	0.35 ± 0.49	0.64 ± 0.79	0.43 ± 0.50

Table 6.3. Comparison of Skeletonization Methods in Noisy Data. Average performance of different skeletonization methods under the presence of Gaussian noise, measured by HD, CHD, and SPmatch metrics, where lower values indicate superior performance. BCST_pruned consistently outperforms other algorithms, showcasing superior performance in noisy data scenarios.

Figure 6.6 shows qualitative results for the different skeletonization methods. LBC, L1-skeleton, and Pypetree either miss or incorrectly identify the directions of branches, unlike the BCST-based approach. However, with the default value of λ set to 0.05, numerous spurious branches remain. Increasing λ to 0.5 mitigates this issue, as demonstrated in Figure 6.6g. This suggests that adjusting the λ parameter may be necessary in noisy settings. Figure E.5 in Appendix E.1 illustrates the same pattern for a pine tree.

6.4 Conclusion

We have demonstrated the effectiveness of the BCST as a competitive alternative for 3D point cloud object skeletonization. The BCST defines a tree with a resilient geometric structure to perturbations making it well-suited for describing the skeleton of 3D objects. However, the BCST spans over all input point, causing the presence of spurious branches that need

removal. To address this, we have developed a pruning algorithm that encourages the inclusion of long branches by maximizing the proposed explainability profit, while discourages the presence of shorter branches with high curvature by minimizing its bending energy. We posed this trade-off optimization problem as an instance of the Price Collecting Steiner tree problem. The fact that the underlying graph is a tree enables the computation of the optimum solution in linear time.

We have evaluated our method on 60 simulated point clouds of different sorts of trees in different experimental settings often encountered in real-life scenarios. Our experiments included point clouds with varying levels of densities, those affected by noise, and those with missing data. Through qualitative and quantitative comparisons, we established the pruned BCST's superior performance across all experiments.

The main advantage of our method lies in its ability to model tree branches effectively, even with a limited number of points. In contrast to competing approaches, which often miss branches in low density levels, the BCST includes the majority of them. This characteristic is of special relevance as it can potentially reduce the costs of the 3D point cloud capture by reducing the number of sampled points. Moreover, our method proves superior performance in handling missing data and noisy samples. Nonetheless, for noisy data, the parameter λ , which regulates the trade-off between the explainability profit and the bending energy, may require further adjustment.

In summary, our BCST method stands out as a competitive and versatile skeletonization approach. It excels in capturing intricate details of point clouds even at low density levels, showcasing robustness against perturbations like noise and missing data.

Chapter 7

Conclusion

In this thesis we have explored various topics related to graphs:

Firstly, Chapter 2 introduced the Directed Probabilistic Watershed, a semi-supervised learning algorithm that extends the capabilities of the Probabilistic Watershed algorithm [57] to directed or asymmetric weighted graphs. We have demonstrated that the method, based on weighted counting of in-forests, can be solved analytically and is equivalent to the random walker approach that computes absorbing probabilities.

Next, Chapter 3 provided analytical expressions for the expected degree and variance of a node in a spanning tree of a weighted graph. Through polynomial manipulation and leveraging the Matrix Tree Theorem we related these statistical properties with the inverse of the Laplacian deriving in the process concise expressions. While our research has primarily delved into theoretical aspects, the exploration of practical applications remains an area for future investigation.

Inspired by the algebraic path problem framework, Chapter 4 introduced the log-norm family of graph node distances. This family of node metrics includes well-known distance metrics such as shortest path distance, potential distance, commute cost distance and mini-max distance as special cases. While the log-norm distance effectively interpolates between these distances, efficient computation remains a challenge, leaving it as future work to develop approximation algorithms. In addition, we have derived both sufficient and necessary conditions for verifying whether an algebraic path problem defines a metric on arbitrary graphs. This contribution may assist practitioners in crafting new metrics.

Chapter 5 proposed the central spanning tree (CST) problem, a parameterized family of spanning trees embedded in a metric space designed to be robust to perturbations. We have explored two variants on the Euclidean space: one permitting the inclusion of Steiner points (referred to as branched central spanning tree or BCST), and another that does not. The (B)CST is defined through a parameterized NP-hard minimization problem over edge lengths, encompassing instances such as the minimum spanning tree or the Steiner tree. In

this problem, the edge lengths are scaled by the tree edge-centralities, which are regulated by a parameter α . We have investigated the impact of α on the tree's structure, demonstrating that higher values of α result in more star-shaped trees, indicating increased robustness at the expense of data fidelity. Moreover, we discussed the $\alpha \rightarrow \pm\infty$ limit cases, proving that for $\alpha > 1$, the (B)CST becomes unsuitable to describe the data as the tree becomes a star-graph with an increasing number of terminals. We also analyzed the trees' geometry, demonstrating analytically that for α values in the range $[0, 0.5] \cup 1$, 4-degree Steiner points are infeasible when the BCST is restricted to a plane embedding. Furthermore, we provided evidence that this holds true for $\alpha \in]0.5, 1[$. In addition, we have proposed a heuristic, named mSTreg, to approximate the (B)CST optimal solution. We have shown its competitiveness on small benchmark datasets.

Finally, Chapter 6, applied the BCST problem to 3D plant skeletonization. We used the mSTreg heuristic to extract the skeleton of 3D point clouds representing plants and developed a pruning algorithm based on the Prize Collecting Steiner tree problem to remove spurious branches of the BCST approximation. We have demonstrated quantitatively and qualitatively the competitiveness of the BCST based approach in comparison to other methods. Future work may explore its application in alternative domains where a skeleton is necessary in some capacity.

Overall, this thesis has made both practical and theoretical contributions. On the one hand, our exploration of properties related to random spanning trees and forests offers valuable insights that can aid the development of algorithms for graph structure analysis. While the expected degree and variance expressions derived in Chapter 3 await specific applications, the Directed Probabilistic Watershed algorithm proposed in Chapter 2 showcases the practical utility of our findings.

On the other hand, we have proposed unifying frameworks, such as the log-norm family of distances and the central spanning tree problem, that effectively encompass various entities in their respective areas. Despite their potential, there remain significant challenges to address. The log-norm family of distances, while suggesting a means to combine the strengths of several popular metrics, remains primarily in the theoretical realm due to the absence of an algorithm to compute these distances. Thus, future research should focus on the development of an efficient algorithm that can compute or approximate the log-norm distance, enabling its application in real-world scenarios.

Similarly, the central spanning tree provides a framework for interpolating between different types of spanning trees, emphasizing the balance between data fidelity and stability. Nonetheless, the stability has only been demonstrated empirically, and though Section 5.3 proposes an approach to formalize and measure the tree stability based on Lipschitz continuity, it is still an open problem how to compute or bound analytically the Lipschitz constant.

On the practical side, we have proposed an heuristic to approximate the optimum of the minimization problem defining the CST problem. However, its lack of guarantees in approximating the optimum and limited testing on small benchmarks underscore the need for further development and validation, especially in higher dimensions. Building upon our heuristic, future work may yield improved methods with rigorous guarantees. Despite these challenges, it is worth highlighting the potential practical impact demonstrated by the experiments on single-cell inference trajectory, involving high-dimensional data, and 3D plant skeletonization. Thus, we anticipate that the CST problem will become relevant in fields where data can be effectively summarized or modeled using a spanning tree approach.

In conclusion, graphs serve as versatile structures for modeling relationships across a wide array of domains. As such, the advancement of methods and tools for analyzing such structures holds significant importance. We expect that the findings presented in this thesis will contribute to the ongoing development and understanding of these methods, fostering our ability to extract meaningful insights from graph data.

Appendix A

Directed Probabilistic Watershed

A.1 Directed Random Walker

The Random Walker proposed by Grady [66] answers the following question for undirected graphs: *What is the probability that a random walker starting at node q reaches seed s_1 before reaching s_2 ?* If we considered the seeds as a set of absorbing nodes, then the probabilities that the Random Walker computes are the absorbing probabilities of the seeds. The Random Walker can easily be extended to the directed setting if we compute these probabilities for a directed graph. In this section, we derive the absorption probabilities of a set of seeds for a directed graph. We expect this result to be well known but we reproduce it here for the convenience of the reader since we could not find any suitable reference that expressed these probabilities in terms of the Laplacian matrix. $G = (V, E, w)$ will stand for a directed graph and $S \subset V$ will be the set of seed/labeled nodes and $U = V \setminus S$ the unlabeled nodes.

Let L denote the Laplacian matrix of G as defined in Definition 3. We index the Laplacian matrix block-wise in terms of the unlabeled and labeled nodes in the following form

$$L = \begin{pmatrix} L_S & B_1 \\ B_2 & L_U \end{pmatrix}. \quad (\text{A.1})$$

Remark A.1.1. Invertibility of L_U : Assume that for any $u \in U$, there exists some $s \in S = V \setminus U$ such that there is a directed path from u to s . Let \hat{G} denote the graph formed after merging all the vertices of S into one node s^* . By assumption, any node $u \in U$ can reach at least one seed in S , therefore s^* will be reachable from any node. Consequently, there exists at least one incoming tree in \hat{G} rooted at s^* . Due to the Directed Matrix Tree Theorem, $\det(L_{\hat{G}}^{[s^*]}) = \det(L_U) \neq 0$, which implies that L_U is non-singular.

In the light of the previous remark we assume that any node $u \in U$ can reach at least one of the seeds $s \in S$ via a directed path. Otherwise, if a node could not reach any seed,

then it will not have a well defined directed random walker probability.

For the moment, let us assume that none of the seeds is absorbing, i.e., every seed has at least one out-going edge. Let

$$P := D^{-1}A = I - D^{-1}L^\top \quad (\text{A.2})$$

be the transition probability matrix where D and A are defined as in the main text. Note that D is invertible since we assume that the seeds are not absorbing and any node can reach one of the seeds, which implies that the out-degree of any node is greater than zero. The entry P_{ij} denotes the probability of transitioning from node i to node j in one hop. Note that $\sum_{j \in V} P_{ij} = 1$ for all i .

We can express, in conjunction with equation (A.2), the transpose of the transition probability matrix as

$$P^\top = \begin{pmatrix} I_S - L_S D_S^{-1} & -B_1 D_U^{-1} \\ -B_2 D_S^{-1} & I_U - L_U D_U^{-1} \end{pmatrix}.$$

Now consider the set of seeds S as absorbing nodes, i.e., once a random walker reaches one of the seeds the random walker will vanish. Hence, the transition probability matrix will have the following form

$$\bar{P}^\top = \begin{pmatrix} I_S & -B_1 D_U^{-1} \\ 0 & I_U - L_U D_U^{-1} \end{pmatrix}$$

Theorem A.1.3 provides a closed formula for $\lim_{n \rightarrow \infty} (\bar{P}^n)^\top$ which will give us the absorbing probabilities of the seeds. In order to prove it, we need to state a series of definitions and results that we state without proof [see 77, Chapter 5.6].

Definition 7. Given an arbitrary matrix $A \in \mathbb{R}^{m \times m}$, we define the 1-norm matrix of A as

$$\|A\|_1 = \max_j \sum_i |A_{ij}|.$$

Lemma A.1.2. [see 77, Corollary 5.6.16] Given an arbitrary matrix $A \in \mathbb{R}^{m \times m}$, if $\|A\|_1 < 1$ then $(A - I)$ is invertible and

$$(I - A)^{-1} = \sum_{i=0}^{\infty} A^i. \quad (\text{A.3})$$

Now we can prove the following result.

Theorem A.1.3. Let \bar{P} be defined as before then

$$\lim_{n \rightarrow \infty} (\bar{P}^\top)^n = \begin{pmatrix} I_S & -B_1 L_U^{-1} \\ 0 & 0 \end{pmatrix}.$$

Proof: Let us first prove by induction that

$$(\bar{P}^\top)^n = \begin{pmatrix} I_S & -B_1 D_U^{-1} \sum_{i=0}^{n-1} (I_U - L_U D_U^{-1})^i \\ 0 & (I_U - L_U D_U^{-1})^n \end{pmatrix}.$$

It naturally holds for $n = 1$. By induction

$$\begin{aligned} (\bar{P}^\top)^{n+1} &= (\bar{P}^\top)^n \bar{P}^\top = \begin{pmatrix} I_S & -B_1 D_U^{-1} \sum_{i=0}^{n-1} (I_U - L_U D_U^{-1})^i \\ 0 & (I_U - L_U D_U^{-1})^n \end{pmatrix} \begin{pmatrix} I_S & -B_1 D_U^{-1} \\ 0 & I_U - L_U D_U^{-1} \end{pmatrix} \\ &= \begin{pmatrix} I_S & -B_1 D_U^{-1} - B_1 D_U^{-1} \sum_{i=0}^{n-1} (I_U - L_U D_U^{-1})^{i+1} \\ 0 & (I_U - L_U D_U^{-1})^{n+1} \end{pmatrix} \\ &= \begin{pmatrix} I_S & -B_1 D_U^{-1} \sum_{i=0}^n (I_U - L_U D_U^{-1})^i \\ 0 & (I_U - L_U D_U^{-1})^{n+1} \end{pmatrix} \end{aligned}$$

Now we will prove the case when $n \rightarrow \infty$. Let us assume that the limit exist. We express it blockwise as

$$\lim_{n \rightarrow \infty} (\bar{P}^\top)^n = \begin{pmatrix} I_S & \bar{P}_1^\top \\ 0 & \bar{P}_2^\top \end{pmatrix}.$$

Since every node $u \in U$ can reach a seed node, s , in a finite number of hops, there exists a number of steps, $k' > 0$, such that for all $k \geq k'$, the probability, $P_{su}(k)$, of being in seed s at step k , is greater than 0. Hence, $M_k := (I_U - L_U D_U^{-1})^k$ is a substochastic matrix and we have

$$\left\| (I_U - L_U D_U^{-1})^k \right\|_1 = \max_j \sum_i \left[(I_U - L_U D_U^{-1})^k \right]_{ij} \underbrace{=}_{\text{maximizer}} 1 - \underbrace{\sum_{s \in S} P_{sj^*}(k)}_{>0} < 1.$$

Therefore,

$$\begin{aligned} \lim_{n \rightarrow \infty} \|M_k^n\|_1 &= \lim_{n \rightarrow \infty} \left\| (I_U - L_U D_U^{-1})^{n \cdot k} \right\|_1 = \lim_{n \rightarrow \infty} \left\| (I_U - L_U D_U^{-1})^n \right\|_1 \\ &\leq \lim_{n \rightarrow \infty} \left(\|I_U - L_U D_U^{-1}\|_1 \right)^n = 0 \Rightarrow \bar{P}_2^\top = \lim_{n \rightarrow \infty} (I_U - L_U D_U^{-1})^n = 0. \end{aligned}$$

Furthermore, as a consequence of Lemma A.1.2

$$\sum_{i=0}^{\infty} (I_U - L_U D_U^{-1})^i = (I_U - I_U + L_U D_U^{-1})^{-1} = (L_U D_U^{-1})^{-1} = D_U L_U^{-1}.$$

Finally,

$$\begin{aligned}\bar{P}_1^\top &= \lim_{n \rightarrow \infty} -B_1 D_U^{-1} \sum_{i=0}^n (I_U - L_U D_U^{-1})^i = -B_1 D_U^{-1} \sum_{i=0}^{\infty} (I_U - L_U D_U^{-1})^i \\ &= -B_1 D_U^{-1} D_U L_U^{-1} = -B_1 L_U^{-1}\end{aligned}$$

□

The entry $[\bar{P}_1^\top]_{su}$, where $s \in S$ and $u \in U$, is the probability that a random walker starting at node u will be absorbed by s . That is because $[\bar{P}_1^\top]_{su} = \sum_{k=1}^{\infty} P_{su}(k)$ where $P_{su}(k)$ is the probability of being at node s at k th step if you started at u .

Remark A.1.4. Note that $\bar{P}_1^\top = -B_1 L_U^{-1}$ is the transposed version of the linear system solved by the undirected Random Walker and equivalently the ProbWS [57, 66], i.e., $\bar{P}_1 = -(L_U^\top)^{-1} B_1^\top$. Since [57, 66] consider an undirected graph, which can be interpreted as a directed graph where each undirected edge has been replaced by a pair of directed edges in opposite directions with the same weight, the equality $(L_U^\top)^{-1} = L_U^{-1}$ holds and the transposition becomes irrelevant.

A.2 Efficient Computation of the DProbWStrw Probabilities

In this section, we will prove Theorem 2.4.4. Firstly, we demonstrate that in the DProbWStrw variant, using a TRW with self-loops is equivalent to using the TRW without self-loops. This equivalence enables us to leverage the Sherman-Morrison formula (Lemma A.2.2) to establish the equality in Theorem 2.4.4, allowing us to efficiently exploit the sparsity of the graph when solving the linear systems determining the DProbWStrw probabilities.

In Theorem 2.4.3 we have proven the equivalence between the random walker and the in-forest approaches. Leveraging this equivalence, we can readily demonstrate that self-loops do not affect the DProbWS probabilities. Since any in-forest does not contain cycles, and therefore self-loops, adding self-loops to the graph does not alter the set of in-forests. Consequently, the total weight of the set of in-forests remains unchanged when self-loops are added to the graph. Therefore, the DProbWS probability remains unaffected (refer to Definition 4). In light of Theorem 2.4.3, this implies that the random walker absorption probabilities of the seeds are also independent of the presence of self-loops. Intuitively, we can argue that in the long run the number of steps the random walker remains immobile at a node (number of steps that the random walker traverses a self-loop) is irrelevant to the seed absorption probability.

Formally, the graph with self-loops and without self-loops define the same Laplacian matrix. Let A be the adjacency matrix of the graph without self-loops. If we add self-loops

to the graph, the adjacency matrix of the graph with self-loops becomes $\hat{A} = A + d$, where d is a diagonal matrix indicating the weights of the self-loops edges. Therefore,

$$L = D - A^\top = \underbrace{D + d}_{\hat{D}} - \underbrace{(A^\top + d)}_{\hat{A}^\top} = \hat{D} - \hat{A}^\top = \hat{L},$$

which means the Laplacian of the graph without self-loops, L , is equal to the Laplacian of the graph with self-loops, \hat{L} . As a consequence of Theorem 2.3.6, the DProbWS probabilities are also equal for both graphs.

Note that, although the Laplacians are unaffected by adding self-loops, the random walker transition probabilities at each node will differ when self-loops are added. However, a random walker defined on a graph with self-loops will determine the same absorption probabilities as a random walker on the same graph without the self-loops. This equivalence holds because the transition probabilities of the graph with self-loops, at each node, conditioned to not traverse the self-loop of the node are equal to the transition probabilities in the graph without the self-loops.

The following lemma proposes a variant of the TRW with self-loops which defines the same DProbWS probabilities as the TRW without self-loops defined in equation (2.15).

Lemma A.2.1. The TRW without self-loops proposed in (2.15) determines the same DProbWS probabilities as the TRW with self-loops defined by the following transition probability matrix

$$\hat{P}_{TRW} = (1 - \hat{\eta})P + \hat{\eta} \frac{1}{n} \mathbb{1}\mathbb{1}^\top, \quad (\text{A.4})$$

where $\hat{\eta} = \frac{n\eta}{\eta+n-1}$.

Proof: As argued in the previous paragraphs, we just need to show that the transition probabilities of the TRW with self-loops at each node conditioned to not traverse the self-loop are equal to the transition probabilities in the graph without the self-loops. The probability of traversing a self-loop in the TRW with self-loops is equal to

$$\hat{P}_{TRW}(\text{self-loop}) = \frac{\hat{\eta}}{n}.$$

Consequently, the transition probability of the TRW with self-loops from node i to node j conditioned to not traverse the self-loop is given by

$$\hat{P}_{TRW}(j|i, \neg\text{self-loop}) = \begin{cases} \frac{\hat{P}_{TRW}(j|i)}{1 - \hat{P}_{TRW}(\text{self-loop})} = \frac{\hat{P}_{TRW}(j|i)}{1 - \frac{\hat{\eta}}{n}} & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}.$$

Hence, the transition probability of the TRW with self-loops conditioned to not traverse any self-loop will be equal to

$$\hat{P}_{TRW, \neg\text{self-loop}} = \frac{(1 - \hat{\eta})}{1 - \frac{\hat{\eta}}{n}} P + \frac{\hat{\eta}}{1 - \frac{\hat{\eta}}{n}} \frac{1}{n} (\mathbb{1}\mathbb{1}^\top - I).$$

Substituting $\hat{\eta}$ by $\frac{n\eta}{\eta+n-1}$ we obtain

$$\hat{P}_{TRW, \neg \text{self-loop}} = (1 - \eta)P + \frac{\eta}{n-1} (\mathbb{1}\mathbb{1}^\top - I) = P_{TRW},$$

i.e., we retrieve the transition probability matrix of the TRW without self-loops (2.15). \square

As we showed in equation (2.16), the linear system that determines the DProbWS probability can be reformulated in terms of the transition probability matrix as:

$$(I - P_U) x_U^s = - \left[\hat{B}_1^\top \right]_s, \quad (\text{A.5})$$

where I is the identity matrix with the appropriate size and $\hat{B}_1^\top = D_U^{-1} B_1^\top$, with D the diagonal matrix whose entries are the out-degree of each node.¹

If we use the TRW with self-loops, as specified by equation (A.4), equation (A.5) becomes

$$\left((1 - \hat{\eta}) (I - P_U^\top) + \frac{\hat{\eta}}{n} \mathbb{1}\mathbb{1}^\top \right) \tilde{x}_U^s = -(1 - \hat{\eta}) \left[\hat{B}_1^\top \right]_s - \frac{\hat{\eta}}{n} \mathbb{1}. \quad (\text{A.6})$$

or alternatively

$$\left((I - P_U^\top) + \frac{\hat{\eta}}{n(1 - \hat{\eta})} \mathbb{1}\mathbb{1}^\top \right) \tilde{x}_U^s = - \left[\hat{B}_1^\top \right]_s - \frac{\hat{\eta}}{n(1 - \hat{\eta})} \mathbb{1}. \quad (\text{A.7})$$

Note that we use x_U^s and \tilde{x}_U^s to refer to the DProbWS and DProbWStrw probabilities respectively.

The difference between the matrices of the linear systems in equations (A.5) and (A.7) is equal to a 1-rank matrix

$$(I - P_U) - \left((I - P_U) + \frac{\hat{\eta}}{n(1 - \hat{\eta})} \mathbb{1}\mathbb{1}^\top \right) = - \frac{\hat{\eta}}{n(1 - \hat{\eta})} \mathbb{1}\mathbb{1}^\top.$$

Therefore, we can use the Sherman-Morrison formula [72].

Lemma A.2.2 (Sherman-Morrison formula [72]). Given an invertible matrix $A \in \mathbb{R}^{m \times m}$ and $u, v \in \mathbb{R}^m$ we have

$$(A + uv^\top)^{-1} = A^{-1} - \frac{A^{-1}uv^\top A^{-1}}{1 + v^\top A^{-1}u}.$$

if and only if $v^\top A^{-1}u \neq -1$.

¹We assume that the out-degree is distinct of zero for all nodes in the set of unlabeled nodes U .

Let $N = I - P_U$, then applying Lemma A.2.2 to (A.7) we obtain

$$\begin{aligned}
\tilde{x}_U^s &= \left((I - P_U) + \frac{\hat{\eta}}{n(1-\hat{\eta})} \mathbb{1}\mathbb{1}^\top \right)^{-1} \left(- [\hat{B}_1^\top]_s - \frac{\hat{\eta}}{n(1-\hat{\eta})} \mathbb{1} \right) \\
&= \left(N + \frac{\hat{\eta}}{n(1-\hat{\eta})} \mathbb{1}\mathbb{1}^\top \right)^{-1} \left(- [\hat{B}_1^\top]_s - \frac{\hat{\eta}}{n(1-\hat{\eta})} \mathbb{1} \right) \\
&\stackrel{\text{Lemma A.2.2}}{=} \left(N^{-1} - \frac{\hat{\eta}}{n(1-\hat{\eta})} \frac{N^{-1} \mathbb{1}\mathbb{1}^\top N^{-1}}{1 + \frac{\hat{\eta}}{n(1-\hat{\eta})} \mathbb{1}^\top N^{-1} \mathbb{1}} \right) \left(- [\hat{B}_1^\top]_s - \frac{\hat{\eta}}{n(1-\hat{\eta})} \mathbb{1} \right) \quad (\text{A.8}) \\
&= -N^{-1} [\hat{B}_1^\top]_s - \frac{\hat{\eta}}{n(1-\hat{\eta})} N^{-1} \mathbb{1} \\
&\quad + \frac{\hat{\eta}}{n(1-\hat{\eta})} \frac{N^{-1} \mathbb{1}\mathbb{1}^\top N^{-1} [\hat{B}_1^\top]_s}{1 + \frac{\hat{\eta}}{n(1-\hat{\eta})} \mathbb{1}^\top N^{-1} \mathbb{1}} + \left(\frac{\hat{\eta}}{n(1-\hat{\eta})} \right)^2 \frac{N^{-1} \mathbb{1}\mathbb{1}^\top N^{-1} \mathbb{1}}{1 + \frac{\hat{\eta}}{n(1-\hat{\eta})} \mathbb{1}^\top N^{-1} \mathbb{1}}
\end{aligned}$$

As stated in (2.16), we have that

$$-N^{-1} [\hat{B}_1^\top]_s = -(I - P_U)^{-1} D_U^{-1} [B_1^\top]_s = x_U^s.$$

Defining $y := N^{-1} \mathbb{1}$, equation (A.8) becomes

$$\begin{aligned}
\tilde{x}_U^s &= x_U^s - \frac{\hat{\eta}}{n(1-\hat{\eta})} y - \frac{\hat{\eta}}{n(1-\hat{\eta})} \frac{y \mathbb{1}^\top x_U^s}{1 + \frac{\hat{\eta}}{n(1-\hat{\eta})} \mathbb{1}^\top y} + \frac{\hat{\eta}}{n(1-\hat{\eta})} \frac{y \mathbb{1}^\top y}{1 + \frac{\hat{\eta}}{n(1-\hat{\eta})} \mathbb{1}^\top y} \\
&= x_U^s - \frac{\frac{\hat{\eta}}{n(1-\hat{\eta})}}{1 + \frac{\hat{\eta}}{n(1-\hat{\eta})} \mathbb{1}^\top y} y \left(\left(1 + \frac{\hat{\eta}}{n(1-\hat{\eta})} \mathbb{1}^\top y \right) + \mathbb{1}^\top x_U^s - \frac{\frac{\hat{\eta}}{n(1-\hat{\eta})}}{1 + \frac{\hat{\eta}}{n(1-\hat{\eta})} \mathbb{1}^\top y} \mathbb{1}^\top y \right)
\end{aligned}$$

Applying now the equality $\hat{\eta} = \frac{n\eta}{\eta+n-1}$ stated in Lemma A.2.1 we finally obtain

$$\tilde{x}_U^s = x_U^s - \frac{\eta(\eta+1)}{1 + \eta \mathbb{1}^\top y} y \left(\left(1 + \frac{\eta}{(n-1)(1-\eta)} \right) + \mathbb{1}^\top x_U^s - \frac{\eta(\eta+1)}{1 + \eta \mathbb{1}^\top y} \mathbb{1}^\top y \right), \quad (\text{A.9})$$

which proves Theorem 2.4.4. Hence, we have shown that DProbWStrw probabilities given by \tilde{x}_U^s can easily be computed via (A.9), if we compute x_U^s and y . Moreover, the computation of the DProbWStrw probabilities require to solve one additional sparse linear system in comparison to the ones needed to compute the DProbWS probabilities, which costs much less effort than solving a dense linear system per seed.

A.3 Further Experiment Details

A.3.1 Reference Methods

ARW method The method proposed in [45] is closely related to ours, due to the use of Absorbing Random Walks (ARW). Conceptually, this method adds an absorbing meta node to the original graph, which is connected to every node through an edge with certain weight \bar{w} .² The algorithm computes the random walker expected accumulated number of visits to

²The weight \bar{w} is implicitly determined by a parameter α in the algorithm, which we set equal to 0.1 for all our experiments. See [45] for more details.

the seeds starting at a query node before being absorbed by the meta node. The expected accumulated number of visits provides a notion of affinity between the nodes. The algorithm assigns the label of the seed that maximizes the expected accumulated number of visits from the unlabeled query node.

LLUD method The method proposed in [194] also uses the random walker. The LLUD method considers an optimization problem over the space of classification functions $\mathcal{H}(V)$, which assigns a real value $f(v)$ to each vertex $v \in V$. The optimization problem is the following

$$\arg \min_{f \in \mathcal{H}(V)} \{\Omega(f) + \gamma \|f - y\|\} \quad (\text{A.10})$$

where y denotes the function in $\mathcal{H}(V)$ defined by $y(v) = 1$ or -1 if vertex v has been labeled as positive or negative respectively, and 0 if it is unlabeled. The functional Ω is defined as

$$\Omega(f) := \frac{1}{2} \sum_{(u,v) \in E} \pi(u) P_{uv} \left(\frac{f(u)}{\sqrt{\pi(u)}} - \frac{f(v)}{\sqrt{\pi(v)}} \right), \quad (\text{A.11})$$

where π is the stationary distribution of the random walker, which is assumed to be unique independently of the starting point.³ The functional Ω forces the classification function to be smooth, while the second term in (A.10) forces the function to preserve the label of the seeds. The balance between these two terms is regulated by the parameter γ .⁴

GTG method The method in [54] interprets the classification of the nodes as a transductive game where each player (node) can choose a strategy among a set of strategies (labels). The proposed transduction game always has a Nash equilibrium which will define the final labeling. Partial payoffs between two nodes are defined based on the weight of the edge connecting these nodes. The total payoff of a node is the sum of its partial payoffs. The Nash equilibrium is computed iteratively till convergence.

A.3.2 Datasets

Digits The *Digits* dataset⁵ [50, 183] consists of 8×8 pixel images of digits. The dataset contains a total of 1797 images divided in 10 classes corresponding to the different digits. We use this dataset to construct a kNN graph. The kNN graph is formed by 1797 nodes and 8985 edges.

³In practice, the uniqueness is ensured thanks to the use of the teleporting random walk.

⁴In the algorithm, the parameter γ is implicitly determined by another parameter μ , which we set equal to 0.9 for all our experiments. See [194] for more details.

⁵https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html

20Newsgroups The *20Newsgroups* dataset⁶ [50, 100] is a collection of 11314 newsgroup documents, partitioned across 20 different newsgroups. We only use the train data. We define a kNN graph out of it. The kNN graph contains 11314 nodes and 56570 edges.

Email-EU The *Email-EU* dataset⁷ [105, 106, 187] is a directed unweighted graph that was generated using email data from a large European research institution. An edge (u, v) is present in the graph if person u sent an email to person v . There is a total of 1005 nodes and 25571 edges. It has a total of 42 classes representing the departments at the research institute.

Cora The *Cora* dataset⁸ [120] is a directed graph where each node represents a scientific publication. An edge (u, v) is present in the graph if paper u cites v . There is a total of 2708 nodes and 5429 edges. It has a total of 7 classes representing the topics of the publications.

CiteseerX The *CiteseerX* dataset⁹ [146] is a directed graph where each node represents a scientific publication. An edge (u, v) is present in the graph if paper u cites v . There is a total of 3264 nodes and 4536 edges. It has a total of 6 classes representing the topics of the publications.

A.3.3 k-Nearest Neighbor Graph Construction

The *Digits* and *20Newsgroups* datasets are not graph datasets. To process them with the DProbWS, we construct k-Nearest Neighbor (kNN) graphs out of them. Directed edges of the kNN graphs are obtained as follows: an edge from node u to node v is present if and only if v is among the k nearest neighbors of u . In our experiments we set $k = 5$ as in [45].

To generate the kNN graph first we need to embed the data points into a metric space. In the case of the *Digits* dataset, since they are images, we just flatten the 8x8 images into a vector of 64 dimensions. The *20Newsgroups* is a text dataset. Via the `TfidfVectorizer` class implemented in `scikit-learn` in Python¹⁰, we embed the datapoints into \mathbb{R}^{130107} . `TfidfVectorizer` maps a collection of raw documents to a matrix of TF-IDF features, where TF-IDF stands for “Term Frequency - Inverse Document Frequency”. TF-IDF is a

⁶https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_20newsgroups.html

⁷<http://snap.stanford.edu/data/email-Eu-core.html>

⁸<https://web.archive.org/web/20151007064508/http://linqs.cs.umd.edu/projects/projects/lbc/>

⁹<https://networkrepository.com/citeseer.php>

¹⁰https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

statistic that aims to better define how important a word is for a document, while also taking into account the relation to other documents from the same corpus [22].

Once the data points have been embedded, the weight of the assigned edge (u, v) is given by $w_{uv} = \exp(-\|f(u) - f(v)\|)$, where $f(u)$ is the image of the node u in the embedding space. Note that the structure of the graph and the weights depend on the distances between the nodes in the embedding space. If the representation of the data in the embedding space is poor, so will be the labeling.

Appendix B

Expected Degree and Variance in Random Spanning Trees

B.1 Proof of Theorem 3.5.2

Here we prove Theorem 3.5.2, a well-known result for which we were unable to locate a suitable reference.

Theorem B.1.1 (Restatement of Theorem 3.5.2). Given a connected edge-weighted undirected graph $G = (V, E, w)$, let \mathcal{T}_G stand for the set of all spanning trees of G . For an edge $e = (u, v) \in E$, the probability that e belongs to spanning tree $T \in \mathcal{T}_G$ is given by

$$\Pr(e \in T) = \begin{cases} w(e) \left(\ell_{uu}^{-1,[r]} + \ell_{vv}^{-1,[r]} - 2\ell_{uv}^{-1,[r]} \right) & \text{if } r \neq u, v \\ w(e)\ell_{uu}^{-1,[v]} & \text{if } r = v \\ w(e)\ell_{vv}^{-1,[u]} & \text{if } r = u \end{cases}, \quad (\text{B.1})$$

where $\ell_{ij}^{-1,[r]}$ denotes the entry ij of the inverse of the matrix $L_G^{[r]}$ (the Laplacian L_G after removing the row and the column corresponding to node r).

Proof: Consider the graph $G_{\setminus e} = (V_G, E_G \setminus \{e\})$ formed from G once edge e has been removed. Note that the Laplacians of G and $G_{\setminus e}$ are related as follows:

$$L_{G_{\setminus e}} = L_G - w(e)b_e(b_e)^\top,$$

where $b_e = \mathbf{1}_u - \mathbf{1}_v$, with $\mathbf{1}_i$ representing the i th column of the identity matrix.

The set of spanning trees, denoted \mathcal{T}_G , can be divided into two subsets: trees that exclude edge e (equivalent to $\mathcal{T}_{G_{\setminus e}}$ since $G_{\setminus e}$ lacks e) and those that include it. By applying the Matrix Tree Theorem (Theorem 1.2.1) and the Determinant Lemma (Lemma 2.3.4), we can derive the probability that a tree will contain edge e . We differentiate three different scenarios based on the removal of node r from the Laplacian when employing the Matrix Tree Theorem.

- Case $r \neq u, v$:

$$\begin{aligned}
\Pr(e \in \mathbf{T}) &= \frac{w(\mathcal{T}_G) - w(\mathcal{T}_{G \setminus e})}{w(\mathcal{T}_G)} \stackrel{\text{Theorem 1.2.1}}{=} \frac{\det(L_G^{[r]}) - \det\left(\overbrace{L_G^{[r]} - w(e)b_e^{[r]}(b_e^{[r]})^\top}^{L_{G \setminus e}^{[r]}}\right)}{\det(L_G^{[r]})} \\
&\stackrel{\text{Lemma 2.3.4}}{=} \frac{\det(L_G^{[r]}) - \det(L_G^{[r]}) \left(1 - w(e) (b_e^{[r]})^\top (L_G^{[r]})^{-1} b_e^{[r]}\right)}{\det(L_G^{[r]})} \\
&= w(e) (\ell_{uu}^{-1, [r]} + \ell_{vv}^{-1, [r]} - 2\ell_{uv}^{-1, [r]})
\end{aligned}$$

Note that we can use Lemma 2.3.4 since $L_G^{[r]}$ is invertible. Indeed, since G is connected it contains at least one spanning tree. Thus $\det(L_G^{[r]}) = w(\mathcal{T}_G) \neq 0$ as a consequence of the Matrix Tree Theorem.

- Case $r = u$:

$$\begin{aligned}
\Pr(e \in \mathbf{T}) &= \frac{w(\mathcal{T}_G) - w(\mathcal{T}_{G \setminus e})}{w(\mathcal{T}_G)} \stackrel{\text{Theorem 1.2.1}}{=} \frac{\det(L_G^{[u]}) - \det\left(\overbrace{L_G^{[u]} - w(e)\mathbf{1}_v^{[u]}(\mathbf{1}_v^{[u]})^\top}^{L_{G \setminus e}^{[u]}}\right)}{\det(L_G^{[u]})} \\
&\stackrel{\text{Lemma 2.3.4}}{=} \frac{\det(L_G^{[u]}) - \det(L_G^{[u]}) \left(1 - w(e) (\mathbf{1}_v^{[u]})^\top (L_G^{[u]})^{-1} \mathbf{1}_v^{[u]}\right)}{\det(L_G^{[u]})} \\
&= w(e) (\ell_{uu}^{-1, [v]})
\end{aligned}$$

- Case $r = v$ is analogous to case $r = u$.

□

Appendix C

Algebraic Path Problem for Graph Metrics

C.1 Min-Norm Semiring

Lemma C.1.1. Given $r > 0$, the min-norm semiring $(S, \oplus, \otimes_r, \infty, 0)$, where $S = \mathbb{R}^+ \cup \{\infty\}$ and

$$x \oplus y = \min(x, y), \quad x \otimes_r y = \sqrt[r]{x^r + y^r}. \quad (\text{C.1})$$

is a commutative selective semiring with $\bar{0} = \infty$ and $\bar{1} = 0$ as its neutral elements.

Proof: We will only show the the associativity of \otimes_r and the distributivity of \otimes_r over \oplus .

- Associativity \otimes_r

$$\begin{aligned} (x \otimes_r y) \otimes_r z &= \sqrt[r]{x^r + y^r} \otimes_r z = \sqrt[r]{(\sqrt[r]{x^r + y^r})^r + z^r} = \sqrt[r]{x^r + y^r + z^r} \\ &= \sqrt[r]{x^r + (\sqrt[r]{y^r + z^r})^r} = x \otimes_r \sqrt[r]{y^r + z^r} = x \otimes_r (y \otimes_r z) \end{aligned} \quad (\text{C.2})$$

- Distributivity:

$$\begin{aligned} (x \oplus y) \otimes_r z &= \sqrt[r]{(\min(x, y))^r + z^r} = \sqrt[r]{\min(x^r, y^r) + z^r} = \sqrt[r]{\min(x^r + z^r, y^r + z^r)} \\ &= \min(\sqrt[r]{x^r + z^r}, \sqrt[r]{y^r + z^r}) = x \otimes_r z \oplus y \otimes_r z \end{aligned} \quad (\text{C.3})$$

The left-distributivity is a consequence of the commutativity of \otimes .

□

Lemma C.1.2. Let $x, y \geq 0$. Then

$$\lim_{r \rightarrow \infty} (x^r + y^r)^{1/r} = \max(x, y).$$

Proof: If $x = 0$ or $y = 0$, then the limit is trivial. Without loss of generality, we will assume $x \geq y > 0$. First note that

$$\sqrt[r]{x^r + y^r} = (x^r + y^r)^{1/r} = \exp\left(\frac{1}{r} \log(x^r + y^r)\right).$$

Then

$$\lim_{r \rightarrow \infty} (x^r + y^r)^{1/r} = \lim_{r \rightarrow \infty} \exp\left(\frac{1}{r} \log(x^r + y^r)\right) = \exp\left(\lim_{r \rightarrow \infty} \frac{1}{r} \log(x^r + y^r)\right).$$

Applying L'Hôpital's rule we obtain

$$\lim_{r \rightarrow \infty} \frac{1}{r} \log(x^r + y^r) = \lim_{r \rightarrow \infty} \frac{x^r \log(x) + y^r \log(y)}{x^r + y^r} = \lim_{r \rightarrow \infty} \frac{\log(x) + \left(\frac{y}{x}\right)^r \log(y)}{1 + \left(\frac{y}{x}\right)^r} \underbrace{=}_{\frac{y}{x} \leq 1} \log(x). \quad (\text{C.4})$$

Therefore $\exp\left(\lim_{r \rightarrow \infty} \frac{1}{r} \log(x^r + y^r)\right) = \exp(\log(x)) = x = \max(x, y)$. \square

As a consequence of Lemma C.1.2, $(S, \oplus, \otimes_r, \infty, 0)$ interpolates between the min-plus and minimax semiring. Therefore, the min-norm distance also interpolates between the shortest path and minimax distance.

C.2 APP of the Eisner Semiring Recovers the First Hitting Cost

We show that the second entry of the APP associated with the Eisner semiring restricted to hitting paths is equal to the first hitting cost $\mathcal{H}(s, t)$:

$$\begin{aligned} \bigoplus_{\varphi \in \mathcal{P}_{st}^h} \bigotimes_{e \in \varphi} (p_e, p_e c_e) &= \bigoplus_{\varphi \in \mathcal{P}_{st}^h} \left(\prod_{e \in \varphi} p_e, \left(\prod_{e \in \varphi} p_e \right) \left(\sum_{e \in \varphi} c_e \right) \right) \\ &= \bigoplus_{\varphi \in \mathcal{P}_{st}^h} \left(\Pr(\varphi), \Pr(\varphi) c(\varphi) \right) = \left(\sum_{\varphi \in \mathcal{P}_{st}^h} \Pr(\varphi), \sum_{\varphi \in \mathcal{P}_{st}^h} \Pr(\varphi) c(\varphi) \right) \quad (\text{C.5}) \\ &= \left(\mathbb{E}_{\varphi \sim \mathcal{P}_{st}^h} [1], \mathbb{E}_{\varphi \sim \mathcal{P}_{st}^h} [c(\varphi)] \right) = \left(1, \mathbb{E}_{\varphi \sim \mathcal{P}_{st}^h} [c(\varphi)] \right) \\ &= (1, \mathcal{H}(s, t)) \end{aligned}$$

C.3 APP of the Log-Semiring Recovers the Potential Distance

We show that the APP associated with the log-semiring, when restricted to hitting paths, recovers the first summand of the potential distance [61, 95], provided that the edge costs

are defined as $p_e \exp(-\mu c_e)$.

$$\begin{aligned}
\bigoplus_{\wp \in \mathcal{P}_{st}^h} \bigotimes_{e \in \wp} -\frac{1}{\mu} \log(p_e \exp(-\mu c_e)) &= \bigoplus_{\wp \in \mathcal{P}_{st}^h} \sum_{e \in \wp} \frac{1}{\mu} \log(p_e \exp(-\mu c_e)) \\
&= \bigoplus_{\wp \in \mathcal{P}_{st}^h} -\frac{1}{\mu} \log\left(\prod_{e \in \wp} p_e \exp(-\mu \sum_{e \in \wp} c_e)\right) \\
&= \bigoplus_{\wp \in \mathcal{P}_{st}^h} -\frac{1}{\mu} \log\left(\Pr(\wp) \exp(-\mu c(\wp))\right) \quad (\text{C.6}) \\
&= -\frac{1}{\mu} \log\left(\sum_{\wp \in \mathcal{P}_{st}^h} \Pr(\wp) \exp(-\mu c(\wp))\right) \\
&= -\frac{1}{\mu} \log\left(\mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} \left[\exp(-\mu c(\wp))\right]\right)
\end{aligned}$$

C.4 Log-Norm Strong Bimonoid

In this section, we prove that the operations defined in equation (4.11), and restated here in (C.7), form a strong bimonoid over $\mathbb{R}^+ \times \mathbb{R}^+ \cup \{\bar{0}\}$, where $\bar{0}$ represents the neutral element of the operation \oplus_μ . Since there is not a natural neutral element in $\mathbb{R}^+ \times \mathbb{R}^+$ for \oplus_μ , we explicitly need to define an ad hoc neutral element.

Lemma C.4.1. Let $r > 1, \mu > 0$. The log-norm algebraic structure

$$\left(\mathbb{R}^+ \times \mathbb{R}^+ \cup \{\bar{0}\}, \oplus_\mu, \otimes_r, \bar{0}, \bar{1}\right),$$

where $\bar{1} = (1, 0)$ and

$$\begin{aligned}
(a, b) \oplus_\mu (c, d) &= \left(1, -\frac{1}{\mu} \log(ae^{-\mu b} + ce^{-\mu d})\right) \\
(a, b) \oplus_\mu \bar{0} &= \bar{0} \oplus_\mu (a, b) = (a, b) \\
(a, b) \otimes_r (c, d) &= \left(ac, \sqrt[r]{c^r + d^r}\right) \\
(a, b) \otimes_r \bar{0} &= \bar{0} \otimes_r (a, b) = \bar{0},
\end{aligned} \quad (\text{C.7})$$

defines a strong bimonoid.

Proof: Note that in (C.7) we define $\bar{0}$ to be absorbing. Thus, we just have left to show that \oplus_μ and \otimes_r define monoids over $\mathbb{R}^+ \times \mathbb{R}^+$. The associativity and commutativity of \otimes_r follow from the associativity and commutativity of the usual product operation in \mathbb{R}^+ and the product operation of the min-norm semiring (Appendix C.2, equation (C.2)). It is also trivial to show that the neutral element of \otimes_r is $\bar{1} = (1, 0)$. Next we show the associativity

of \oplus_μ .

$$\begin{aligned}
((a_1, a_2) \oplus_\mu (b_1, b_2)) \oplus_\mu (c_1, c_2) &= \left(1, -\frac{1}{\mu} \log \left(a_1 e^{-\mu a_2} + b_1 e^{-\mu b_2}\right)\right) \oplus_\mu (c_1, c_2) \\
&= \left(1, -\frac{1}{\mu} \log \left(a_1 e^{-\mu a_2} + b_1 e^{-\mu b_2} + c_1 e^{-\mu c_2}\right)\right) \\
&= (a_1, a_2) \oplus_\mu \left(1, -\frac{1}{\mu} \log \left(b_1 e^{-\mu b_2} + c_1 e^{-\mu c_2}\right)\right) \\
&= (a_1, a_2) \oplus_\mu ((b_1, b_2) \oplus_\mu (c_1, c_2)).
\end{aligned} \tag{C.8}$$

The commutativity of \oplus_μ follows from the commutativity of the common sum. \square

Lemma C.4.2. Let $r > 1, \mu > 0$. The associated APP of the log-norm strong bimonoid defines the log-norm distance when the costs are set equal to (p_e, c_e) , i.e.

$$\begin{aligned}
\text{LN}(s, t) &= \left[\bigoplus_{\varphi \in \mathcal{P}_{st}^h} \bigotimes_{e \in \varphi}^\mu (p_e, c_e) + \bigoplus_{\varphi \in \mathcal{P}_{ts}^h} \bigotimes_{e \in \varphi}^\mu (p_e, p_e c_e) \right]_2 \\
&= -\frac{1}{\mu} \log \left(\mathbb{E}_{\varphi \sim \mathcal{P}_{st}^h} [\exp(-\mu \|c(\varphi)\|_r)] \right) - \frac{1}{\mu} \log \left(\mathbb{E}_{\varphi \sim \mathcal{P}_{ts}^h} [\exp(-\mu \|c(\varphi)\|_r)] \right).
\end{aligned} \tag{C.9}$$

Proof:

$$\begin{aligned}
\bigoplus_{\varphi \in \mathcal{P}_{st}^h} \bigotimes_{e \in \varphi}^\mu (p_e, p_e c_e) &= \bigoplus_{\varphi \in \mathcal{P}_{st}^h} \left(\prod_{e \in \varphi} p_e, \sqrt[r]{\sum_{e \in \varphi} c_e^r} \right) \\
&= \bigoplus_{\varphi \in \mathcal{P}_{st}^h} \left(\Pr(\varphi), \|c(\varphi)\|_r \right) \\
&= \left(1, -\frac{1}{\mu} \log \left(\sum_{\varphi \in \mathcal{P}_{st}^h} \Pr(\varphi) \exp(-\mu \|c(\varphi)\|_r) \right)\right) \\
&= \left(1, -\frac{1}{\mu} \log \left(\mathbb{E}_{\varphi \sim \mathcal{P}_{st}^h} [\exp(-\mu \|c(\varphi)\|_r)] \right)\right)
\end{aligned} \tag{C.10}$$

\square

Corollary C.4.3. Let $\mu > 0$.

- The log-max algebraic structure $(\mathbb{R}^+ \times \mathbb{R}^+ \cup \{\bar{0}\}, \oplus_\mu, \otimes_\infty, \bar{0}, \bar{1} = (1, 0))$, where

$$\begin{aligned}
(a, b) \oplus_\mu (c, d) &= \left(1, -\frac{1}{\mu} \log \left(a e^{-\mu b} + c e^{-\mu d}\right)\right) \\
(a, b) \oplus_\mu \bar{0} &= \bar{0} \oplus_\mu (a, b) = (a, b) \\
(a, b) \otimes_\infty (c, d) &= (ac, \max(c, d)) \\
(a, b) \otimes_\infty \bar{0} &= \bar{0} \otimes_r (a, b) = \bar{0},
\end{aligned} \tag{C.11}$$

defines a strong bimonoid.

- The associated APP of the Log-max strong bimonoid defines the log-norm distance when the costs are equal to (p_e, c_e) , i.e.

$$\begin{aligned} \text{LM}(s, t) &= \left[\bigoplus_{\wp \in \mathcal{P}_{st}^h}^{\mu} \bigotimes_{e \in \wp}^{\infty} (p_e, c_e) + \bigoplus_{\wp \in \mathcal{P}_{ts}^h}^{\mu} \bigotimes_{e \in \wp}^{\infty} (p_e, p_e c_e) \right]_2 \\ &= -\frac{1}{\mu} \left(\log \left(\mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} \left[e^{-\mu \max_{e \in \wp} c(e)} \right] \right) + \log \left(\mathbb{E}_{\wp \sim \mathcal{P}_{ts}^h} \left[e^{-\mu \max_{e \in \wp} c(e)} \right] \right) \right). \end{aligned} \quad (\text{C.12})$$

Proof: Consequence of the fact that $\lim_{r \rightarrow \infty} \otimes_r = (\times, \max)$, Lemma C.4.1, and Lemma C.4.2 (see Lemma C.1.2). \square

Analogously, we define the strong bimonoids that would define the exp-norm and exp-max distances presented in Table 4.1.

Lemma C.4.4.

1. Let $r > 0$. The exp-norm algebraic structure $(\mathbb{R}^+ \times \mathbb{R}, \oplus, \otimes_r, (0, 0), (1, 0))$ defines an strong bimonoid, where

$$\begin{aligned} (a, b) \oplus_{\mu} (c, d) &= (a + c, b + d) \\ (a, b) \otimes_r (c, d) &= \left(ac, ac \sqrt[r]{\left(\frac{b}{a}\right)^r + \left(\frac{d}{c}\right)^r} \right). \end{aligned} \quad (\text{C.13})$$

2. The exp-max algebraic structure $(\mathbb{R}^+ \times \mathbb{R}, \oplus, \otimes_{\infty}, (0, 0), (1, 0))$ defines an strong bimonoid, where

$$\begin{aligned} (a, b) \oplus_{\mu} (c, d) &= (a + c, b + d) \\ (a, b) \otimes_r (c, d) &= \left(ac, ac \max \left(\frac{b}{a}, \frac{d}{c} \right) \right). \end{aligned} \quad (\text{C.14})$$

Moreover the associated APP of the exp-norm and exp-max strong bimonoids define the exp-norm and exp-max distances respectively when the costs are equal to $(p_e, p_e c_e)$, i.e.

$$\left[\bigoplus_{\wp \in \mathcal{P}_{st}^h}^{\mu} \bigotimes_{e \in \wp}^r (p_e, p_e c_e) + \bigoplus_{\wp \in \mathcal{P}_{ts}^h}^{\mu} \bigotimes_{e \in \wp}^r (p_e, p_e c_e) \right]_2 = \mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} [\|c(\wp)\|_r] + \mathbb{E}_{\wp \sim \mathcal{P}_{ts}^h} [\|c(\wp)\|_r], \quad (\text{C.15})$$

and

$$\left[\bigoplus_{\wp \in \mathcal{P}_{st}^h}^{\mu} \bigotimes_{e \in \wp}^{\infty} (p_e, p_e c_e) + \bigoplus_{\wp \in \mathcal{P}_{ts}^h}^{\mu} \bigotimes_{e \in \wp}^{\infty} (p_e, p_e c_e) \right]_2 = \mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} \left[\max_{e \in \wp} c(e) \right] + \mathbb{E}_{\wp \sim \mathcal{P}_{ts}^h} \left[\max_{e \in \wp} c(e) \right]. \quad (\text{C.16})$$

Proof: We will only proof the exp-norm case, since the exp-max case is analogous. To prove that the exp-norm is an strong bimonoid we will just show that the \otimes_r is associative. The rest of properties are trivial.

$$\begin{aligned}
((a_1, a_2) \otimes_r (b_1, b_2)) \otimes_r (c_1, c_2) &= \left(a_1 b_1, a_1 b_1 \sqrt[r]{\left(\frac{a_2}{a_1}\right)^r + \left(\frac{b_2}{b_1}\right)^r} \right) \otimes_r (c_1, c_2) \\
&= \left(a_1 b_1 c_1, a_1 b_1 c_1 \sqrt[r]{\left(\frac{a_1 b_1 \sqrt[r]{\left(\frac{a_2}{a_1}\right)^r + \left(\frac{b_2}{b_1}\right)^r}}{a_1 b_1}\right)^r + \left(\frac{c_2}{c_1}\right)^r} \right) \\
&= \left(a_1 b_1 c_1, a_1 b_1 c_1 \sqrt[r]{\left(\frac{a_2}{a_1}\right)^r + \left(\frac{b_2}{b_1}\right)^r + \left(\frac{c_2}{c_1}\right)^r} \right) \\
&= (a_1, a_2) \otimes_r \left(b_1 c_1, b_1 c_1 \sqrt[r]{\left(\frac{b_2}{b_1}\right)^r + \left(\frac{c_2}{c_1}\right)^r} \right) \\
&= (a_1, a_2) \otimes_r ((b_1, b_2) \otimes_r (c_1, c_2)).
\end{aligned} \tag{C.17}$$

The APP of the exp-norm strong bimonoid follows from

$$\begin{aligned}
\bigoplus_{\wp \in \mathcal{P}_{st}^h}^{\mu} \bigotimes_{e \in \wp}^r (p_e, p_e c_e) &= \bigoplus_{\wp \in \mathcal{P}_{st}^h}^{\mu} \left(\prod_{e \in \wp} p_e, \left(\prod_{e \in \wp} p_e \right) \sqrt[r]{\sum_{e \in \wp} c_e^r} \right) \\
&= \bigoplus_{\wp \in \mathcal{P}_{st}^h}^{\mu} \left(\Pr(\wp), \Pr(\wp) \|c(\wp)\|_r \right) = \left(\sum_{\wp \in \mathcal{P}_{st}^h} \Pr(\wp), \sum_{\wp \in \mathcal{P}_{ts}^h} \Pr(\wp) \|c(\wp)\|_r \right) \\
&= \left(\mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} [1], \mathbb{E}_{\wp \sim \mathcal{P}_{ts}^h} [\|c(\wp)\|_r] \right) = \left(1, \mathbb{E}_{\wp \sim \mathcal{P}_{ts}^h} [\|c(\wp)\|_r] \right).
\end{aligned} \tag{C.18}$$

□

C.5 Log-Norm Metric Limits

In this section, we prove the limits of the log-norm distance shown in Table 4.1. First, in Lemma C.5.1 we prove the limits when $\mu \rightarrow 0^+$ and $\mu \rightarrow \infty$ for a finite r .

Lemma C.5.1.

1.

$$\begin{aligned}
\lim_{\mu \rightarrow 0^+} -\frac{1}{\mu} \log \left(\mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} \left[\exp \left(-\mu \|c(\wp)\|_r \right) \right] \right) &- \frac{1}{\mu} \log \left(\mathbb{E}_{\wp \sim \mathcal{P}_{ts}^h} \left[\exp \left(-\mu \|c(\wp)\|_r \right) \right] \right) \\
&= \mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} [\|c(\wp)\|_r] + \mathbb{E}_{\wp \sim \mathcal{P}_{ts}^h} [\|c(\wp)\|_r]
\end{aligned} \tag{C.19}$$

2.

$$\begin{aligned}
& \lim_{\mu \rightarrow \infty} -\frac{1}{\mu} \log \left(\mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} \left[\exp \left(-\mu \|c(\wp)\|_r \right) \right] \right) - \frac{1}{\mu} \log \left(\mathbb{E}_{\wp \sim \mathcal{P}_{ts}^h} \left[\exp \left(-\mu \|c(\wp)\|_r \right) \right] \right) \\
&= 2 \min_{\wp \in \mathcal{P}_{st}^h} \|c(\wp)\|_r
\end{aligned} \tag{C.20}$$

Proof: By showing the limit of the first summands we can derive the limit of the second in an analogous way.

1.

$$\begin{aligned}
& \lim_{\mu \rightarrow 0^+} -\frac{1}{\mu} \log \left(\mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} \left[\exp \left(-\mu \|c(\wp)\|_r \right) \right] \right) = \\
& \lim_{\mu \rightarrow 0^+} -\frac{1}{\mu} \log \left(\sum_{\wp \in \mathcal{P}_{st}^h} (\Pr(\wp) \exp(-\mu \|c(\wp)\|_r)) \right) \stackrel{\text{L'Hôpital's rule}}{=} \\
& \lim_{\mu \rightarrow 0^+} \frac{\sum_{\wp \in \mathcal{P}_{st}^h} (\Pr(\wp) \|c(\wp)\|_r \exp(-\mu \|c(\wp)\|_r))}{\sum_{\wp \in \mathcal{P}_{st}^h} (\Pr(\wp) \exp(-\mu \|c(\wp)\|_r))} = \sum_{\wp \in \mathcal{P}_{st}^h} \Pr(\wp) \|c(\wp)\|_r = \mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} [\|c(\wp)\|_r]
\end{aligned} \tag{C.21}$$

2.

$$\begin{aligned}
& \lim_{\mu \rightarrow \infty} -\frac{1}{\mu} \log \left(\mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} \left[\exp \left(-\mu \|c(\wp)\|_r \right) \right] \right) = \\
& \lim_{\mu \rightarrow \infty} -\frac{1}{\mu} \log \left(\sum_{\wp \in \mathcal{P}_{st}^h} \left(\Pr(\wp) \exp \left(-\mu \sum_{e \in \wp} c_e \right) \right) \right) \stackrel{\text{L'Hôpital's rule}}{=} \\
& \lim_{\mu \rightarrow \infty} \frac{\sum_{\wp \in \mathcal{P}_{st}^h} (\Pr(\wp) \|c(\wp)\|_r \exp(-\mu \|c(\wp)\|_r))}{\sum_{\wp \in \mathcal{P}_{st}^h} (\Pr(\wp) \exp(-\mu \|c(\wp)\|_r))} = \min_{\wp \in \mathcal{P}_{st}^h} \|c(\wp)\|_r
\end{aligned} \tag{C.22}$$

□

Since $\lim_{r \rightarrow \infty} x \otimes_r y = \lim_{r \rightarrow \infty} \sqrt[r]{x^r + y^r} = \max(x, y)$ as a consequence of Lemma C.1.2, we obtain all the limits exposed in Table 4.1. Note that for $r = 1$ we retrieve the potential distance [61, 95] and their limits.

C.6 Proofs of Section 4.4

C.6.1 Proof Lemma 4.4.2

Lemma C.6.1 (Lemma 4.4.2). Given a graph G and arbitrary nodes s, t and q then

1.

$$\text{APP}^h(s, t) = \alpha^h \oplus \beta^h \otimes \text{APP}^h(q, t). \tag{C.23}$$

2.

$$\text{APP}^h(s, q) = \beta^h \oplus \alpha^h \otimes \text{APP}^h(t, q), \quad (\text{C.24})$$

where

$$\alpha^h := \bigoplus_{\substack{\wp \in \mathcal{P}_{st}^h \\ q \notin \wp}} \bigotimes_{e \in \wp} c(e), \quad \beta^h := \bigoplus_{\substack{\wp_1 \in \mathcal{P}_{sq}^h \\ t \notin \wp_1}} \bigotimes_{e \in \wp_1} c(e).$$

Proof:

1.

$$\begin{aligned} \text{APP}^h(s, t) &= \bigoplus_{\wp \in \mathcal{P}_{st}^h} \bigotimes_{e \in \wp} c(e) \\ &= \bigoplus_{\substack{\wp \in \mathcal{P}_{st}^h \\ q \notin \wp}} \bigotimes_{e \in \wp} c(e) \oplus \bigoplus_{\substack{\wp \in \mathcal{P}_{st}^h \\ q \in \wp}} \bigotimes_{e \in \wp} c(e) \\ &= \bigoplus_{\substack{\wp \in \mathcal{P}_{st}^h \\ q \notin \wp}} \bigotimes_{e \in \wp} c(e) \oplus \bigoplus_{\substack{\wp_1 \in \mathcal{P}_{sq}^h \\ t \notin \wp_1 \\ \wp_2 \in \mathcal{P}_{qt}^h}} \left(\bigotimes_{e \in \wp_1} c(e) \right) \otimes \left(\bigotimes_{e \in \wp_2} c(e) \right) \\ &= \underbrace{\bigoplus_{\substack{\wp \in \mathcal{P}_{st}^h \\ q \notin \wp}} \bigotimes_{e \in \wp} c(e)}_{\alpha^h} \oplus \underbrace{\left(\bigoplus_{\substack{\wp_1 \in \mathcal{P}_{sq}^h \\ t \notin \wp_1}} \bigotimes_{e \in \wp_1} c(e) \right)}_{\beta^h} \otimes \underbrace{\left(\bigoplus_{\wp_2 \in \mathcal{P}_{qt}^h} \bigotimes_{e \in \wp_2} c(e) \right)}_{\text{APP}^h(q, t)} \\ &= \alpha^h \oplus \beta^h \otimes \text{APP}^h(q, t). \end{aligned} \quad (\text{C.25})$$

2. The second equality is proven analogously to the previous one once the following permutation is done:

$$s \rightarrow s, q \rightarrow t, t \rightarrow q$$

□

C.6.2 Proof Theorem 4.4.6

Theorem C.6.2 (Theorem 4.4.6). Let $G = (V, E)$ be an S-graph. If

1. \mathbf{g} is \otimes -subadditive, i.e. $\mathbf{g}(a \otimes b) \leq \mathbf{g}(a) + \mathbf{g}(b)$, $\forall a, b \in S$,
2. \mathbf{g} is increasing in $S \setminus \{\bar{0}\}$ with respect to the order defined in (4.1), i.e., $a \preceq b \rightarrow \mathbf{g}(a) \leq \mathbf{g}(b) \forall a, b \in S \setminus \{\bar{0}\}$,
3. $a \preceq a \otimes \text{APP}^h(t, q) \otimes \text{APP}^h(q, t) \forall a \in S, q, t \in V$, i.e., aggregating the cost of the cycles starting at an arbitrary node q and traversing a node t , does not decrease the cost.¹

¹And also the distance, since \mathbf{g} is increasing.

then d , as defined in (4.12), satisfies the triangle inequality over the nodes of G .

Remark C.6.3. We need to consider that \mathfrak{g} is increasing in $S \setminus \{\bar{0}\}$ because $\bar{0} \preceq s$, $\forall s \in S$ since $s \oplus \bar{0} = s$. However, by assumption $\mathfrak{g}(\bar{0}) = \infty$, thus $\mathfrak{g}(\bar{0}) \geq \mathfrak{g}(s)$. If we did not exclude $\bar{0}$, \mathfrak{g} would map all elements of S to ∞ .

Proof: We need to prove that the triangle inequality (4.15) holds. Let s , q and t be arbitrary nodes of G . Due to the subadditivity of \mathfrak{g} , we have

$$\mathfrak{g}(\text{APP}^h(s, q) \otimes \text{APP}^h(q, t)) \leq \mathfrak{g}(\text{APP}^h(s, q)) + \mathfrak{g}(\text{APP}^h(q, t)).$$

Therefore, as a consequence of (C.23), it will be enough to show

$$\begin{aligned} d_L(s, t) &= \mathfrak{g}(\text{APP}^h(s, t)) = \mathfrak{g}(\alpha^h \oplus \beta^h \otimes \text{APP}^h(q, t)) \\ &\leq \mathfrak{g}(\text{APP}^h(s, q) \otimes \text{APP}^h(q, t)) = d_L(s, q) + d_L(q, t), \end{aligned}$$

which will follow from

$$\alpha^h \oplus \beta^h \otimes \text{APP}^h(q, t) \preceq \text{APP}^h(s, q) \otimes \text{APP}^h(q, t). \quad (\text{C.26})$$

since \mathfrak{g} is increasing. As a consequence of equation (C.24), it suffices to show the following inequality

$$\begin{aligned} \alpha^h \oplus \beta^h \otimes \text{APP}^h(q, t) &\preceq \text{APP}^h(s, q) \otimes \text{APP}^h(q, t) \\ &= (\alpha^h \otimes \text{APP}^h(t, q) \oplus \beta^h) \otimes \text{APP}^h(q, t) \\ &= \alpha^h \otimes \text{APP}^h(t, q) \otimes \text{APP}^h(q, t) \oplus \beta^h \otimes \text{APP}^h(q, t), \end{aligned} \quad (\text{C.27})$$

which holds if

$$\alpha^h \preceq \alpha^h \otimes \text{APP}^h(t, q) \otimes \text{APP}^h(q, t). \quad (\text{C.28})$$

Indeed, (C.28) holds thanks to our third assumption. \square

C.7 Use Case of the Results in Section 4.4

Corollaries from Theorem 4.4.5 and Theorem 4.4.6

As a consequence of Theorem 4.4.5 we can show that the Min-norm distances and the potential distance define proper metrics (Corollary C.7.1 and (Corollary C.7.2)). Analogously, from Theorem 4.4.6 we can show that the commute Cost Distance defines a metric (Corollary C.7.3).

Corollary C.7.1. Min-norm distances, including the shortest path and minimax distances, are graph node metrics.

Proof: We will apply Theorem 4.4.5. Since \mathfrak{g} is equal to the identity function, the subadditivity follows from the subadditivity of r -th roots:

$$a \otimes_r b = \sqrt[r]{a^r + b^r} \leq a + b, \quad \forall r \geq 1, i. \quad (\text{C.29})$$

Trivially, the subadditivity also holds for the max operation.

Furthermore, the identity function \mathfrak{g} is a decreasing function since

$$a \preceq b \iff \exists c \in S \text{ s.t. } \min(a, c) = a \oplus c = b \iff b \leq a. \quad (\text{C.30})$$

Finally, the third assumption is a consequence of the increasing nature of the \otimes_r and max operations: for $a, b \in \mathbb{R}^+$ we have

$$a \otimes_r b = \sqrt[r]{a^r + b^r} \geq a \Rightarrow a \otimes_r b \preceq a; \quad \max(a, b) \geq a \Rightarrow a \otimes_\infty b \preceq a$$

Therefore, $a \otimes \text{APP}^h(q, t) \otimes \text{APP}^h(t, q) \preceq a$.

□

Corollary C.7.2. The potential distance [61, 95] defines a metric.

Proof: We will apply Theorem 4.4.5. The potential distance can be retrieved by the APP associated with the log-semiring. In this case \otimes coincides with $+$. Therefore, \mathfrak{g} is the identity \otimes -homomorphism and the subadditivity follows trivially.

The function \mathfrak{g} is a decreasing function since

$$\begin{aligned} a \preceq b &\iff \exists c \in S \text{ s.t. } a \oplus_\mu c = -\frac{1}{\mu} \log(e^{-\mu a} + e^{-\mu c}) = b \\ &\iff \exists c \text{ s.t. } c = -\frac{1}{\mu} \log(e^{-\mu b} - e^{-\mu a}) \iff b \leq a \end{aligned} \quad (\text{C.31})$$

The third assumption of Theorem 4.4.5 follows from the fact that the cost of a path is strictly positive and that $a \otimes b = a + b \preceq a$ since $a \leq a + b$ for $b \geq 0$. □

Corollary C.7.3. The Commute Cost Distance defines a metric.

Proof: Note that the values of the semiring that we consider lie in $\{1\} \times \mathbb{R}^+$, since the first entry of $\text{APP}(s, t)^h$ is equal to $\sum_{\varphi \in \mathcal{P}_{st}} \Pr(\varphi) = 1$ (see Remark 4.4.4). Thus we just need to show the properties of Theorem 4.4.6 for this subset of elements.

In this case, \mathfrak{g} is the projection of the second entry of $\mathbb{R}^+ \times \mathbb{R}^+$. Hence, the subadditivity follows from

$$\mathfrak{g}((1, c_1) \otimes (1, c_2)) = \mathfrak{g}((1, c_1 + c_2)) = c_1 + c_2 = \mathfrak{g}((1, c_1)) + \mathfrak{g}((1, c_2)). \quad (\text{C.32})$$

The function \mathfrak{g} is an increasing function since

$$\begin{aligned}
(p_1, c_1) \preceq (p_2, c_2) &\iff \\
\exists (p_3, c_3) \in S \text{ s.t. } (p_1, c_1) \oplus_\mu (p_3, c_3) &= (p_1 + p_3, c_1 + c_3) = (p_2, c_2) \\
\iff \exists p_3, c_3 > 0 \text{ s.t. } p_1 + p_3 = p_2 \ \& \ c_1 + c_3 = c_2 \\
\Rightarrow c_1 &\leq c_2.
\end{aligned} \tag{C.33}$$

The third assumption of Theorem 4.4.6 follows from

$$(p_1, c_1) \preceq (p_1, c_1) \otimes \underbrace{\text{APP}^h(t, q)}_{=(1, c_2)} \otimes \underbrace{\text{APP}^h(q, t)}_{=(1, c_3)} = (p_1, c_1) \otimes (1, c_2) \otimes (1, c_3) = (p_1, c_1 + p_1(c_2 + c_3)),$$

since $c_1 \leq c_1 + p_1(c_2 + c_3)$ because $p_1, c_2, c_3 \geq 0$. \square

Novel metric

In order to illustrate the application of the results exposed in Section 4.4, we will define a graph distance that can be easily verified to be a metric thanks to Theorem 4.4.5.

Let $p_{ij} \in [0, 1)$ be the transition probabilities of a vanishing random walker, i.e. a random walker for which there exist at least one node k where $\sum_j p_{kj} < 1$. That is, there is a non-zero probability that the random walker "vanishes". Alternatively, one could interpret that there exists an absorbing node, i^* , to which every node, i , can transition with probability $p_{ii^*} = 1 - \sum_{j \neq i^*} p_{ij}$. In such case, there exists at least one node k such that $p_{ki^*} > 0$.

Let

$$d(s, t) := \begin{cases} \overbrace{-\frac{1}{r} \log \left(\sum_{\emptyset \in \mathcal{P}_{st}^h} \prod_{(i,j) \in E} p_{ij}^r \right)}^{d_L(s,t)} \overbrace{-\frac{1}{r} \log \left(\sum_{\emptyset \in \mathcal{P}_{ts}^h} \prod_{(i,j) \in E} p_{ij}^r \right)}^{d_R(s,t)} & \text{if } s \neq t \\ 0 & \text{otherwise} \end{cases} \tag{C.34}$$

We claim that (C.34) defines a metric. To prove it we will apply Theorem 4.4.5.

First, consider the semiring $S = \{\mathbb{R}^+, \oplus_r, \cdot, 0, 1\}$ with $x \oplus_r y = \sqrt[r]{x^r + y^r}$, $r \geq 1$ and $g(x) = -\log(x)$. Note that if $s \neq t$, then

$$\bigoplus_{\emptyset \in \mathcal{P}_{st}^h} \bigotimes_{e \in \emptyset} p_e = \prod_{\emptyset \in \mathcal{P}_{st}^h} \sqrt[r]{\sum_{e \in \emptyset} p_e^r} = \sqrt[r]{\prod_{\emptyset \in \mathcal{P}_{st}^h} \sum_{e \in \emptyset} p_e^r}. \tag{C.35}$$

If we apply the function $\mathfrak{g}(x) = -\log(x)$ to (C.35), we retrieve (C.34):

$$\mathfrak{g} \left(\bigoplus_{\emptyset \in \mathcal{P}_{st}^h} \bigotimes_{e \in \emptyset} p_e \right) = -\log \left(\sqrt[r]{\prod_{\emptyset \in \mathcal{P}_{st}^h} \sum_{e \in \emptyset} p_e^r} \right) = -\frac{1}{r} \log \left(\sum_{\emptyset \in \mathcal{P}_{st}^h} \prod_{(i,j) \in E} p_{ij}^r \right) = d_L(s, t). \tag{C.36}$$

From the definition of (C.34), it is clear that $d(s, t)$ satisfies the symmetry and indiscernible properties. It is only left to prove the triangle inequality to show that (C.34) is indeed a metric. This property follows easily from Theorem 4.4.5, since:

- $\mathfrak{g}(x) = -\log(x)$ is \otimes -subadditive:

$$\begin{aligned}\mathfrak{g}(a \otimes b) &= -\log(a \otimes b) = \log\left(\sqrt[r]{a^r + b^r}\right) = -\frac{1}{r}\log(a^r + b^r) \\ &= -\frac{1}{r}\log(a^r) - \frac{1}{r}\log(b^r) = -\log(a) - \log(b) = \mathfrak{g}(a) + \mathfrak{g}(b)\end{aligned}$$

- $\mathfrak{g}(x) = -\log(x)$ is decreasing: Note that $a \preccurlyeq b \iff \exists c \in \mathbb{R}^+$ such that $a^r + c^r = b^r$. Consequently,

$$a \preccurlyeq b \iff a \leq b. \quad (\text{C.37})$$

Thus, if $a \preccurlyeq b$, then

$$\mathfrak{g}(b) = -\log(b) = -\log\left(\sqrt[r]{a^r + c^r}\right) = -\frac{1}{r}\log(a^r + c^r) \leq -\frac{1}{r}\log(a^r) = -\log(a) = \mathfrak{g}(a).$$

- A similar argument as was used in [61] (see Remark 4.4.4) can be used to show that $\text{APP}^h(t, q) = \bigoplus_{\varphi \in \mathcal{P}_{st}^h} \bigotimes_{e \in \varphi} p_e \leq 1$. Hence, $a \otimes \text{APP}^h(t, q) \otimes \text{APP}^h(q, t) \preccurlyeq a$, $\forall a \in S$, $q, t \in V$ follows from this fact together with (C.37).

We have just proven the following corollary.

Corollary C.7.4. Given the transition probabilities of a vanishing random walker, $p_{ij} \in [0, 1)$, over an arbitrary graph G , the function $d(s, t)$ defines a metric.

$$d(s, t) := \begin{cases} -\frac{1}{r}\log\left(\sum_{\varphi \in \mathcal{P}_{st}^h} \prod_{(i,j) \in E} p_{ij}^r\right) - \frac{1}{r}\log\left(\sum_{\varphi \in \mathcal{P}_{ts}^h} \prod_{(i,j) \in E} p_{ij}^r\right) & \text{if } s \neq t \\ 0 & \text{otherwise} \end{cases}. \quad (\text{C.38})$$

We provide a bit of intuition about this metric: when $r = 1$, the expression in the brackets of the logarithm is equal to the absorbing probability of t from s before the random walker vanishes.

$$\lim_{r \rightarrow 1} \sum_{\varphi \in \mathcal{P}_{st}^h} \prod_{(i,j) \in E} p_{ij}^r = \sum_{\varphi \in \mathcal{P}_{st}^h} \prod_{(i,j) \in E} p_{ij} = \sum_{\varphi \in \mathcal{P}_{st}^h} \text{Pr}(\varphi)$$

Thus, when $r \rightarrow 1$, nodes are closer if they have higher absorbing probability before the RW vanishes. On the other extreme, when $r \rightarrow \infty$, then the distance focuses on the path of maximum probability between two nodes. If we consider $p_{ij} = \exp(-c_{ij})$,² the limit case

²In this case we are assuming that c_{ij} are high enough for all i, j such that $\sum_k \exp(-c_{ik}) < 1$.

$r \rightarrow \infty$ would be equivalent to twice the shortest path cost with edge costs equal to c_{ij} :

$$\begin{aligned} \lim_{r \rightarrow \infty} d_L(s, t) &= \lim_{r \rightarrow \infty} -\frac{1}{r} \log \left(\sum_{\varphi \in \mathcal{P}_{st}^h} \prod_{(i,j) \in E} \exp(-r \cdot c_{ij}) \right) \\ &= \lim_{r \rightarrow \infty} -\log \left(\sqrt[r]{\sum_{\varphi \in \mathcal{P}_{st}^h} \exp(-r \cdot c(\varphi))} \right) = -\log \left(\max_{\varphi \in \mathcal{P}_{st}^h} \exp(-c(\varphi)) \right) \\ &= \min_{\varphi \in \mathcal{P}_{st}} c(\varphi). \end{aligned}$$

C.8 Log-Norm Distance

In this section we will prove that the log-norm distance defines a metric over the nodes of a graph with positive edge-costs. Since the log-norm operations \otimes_r and \oplus_μ do not define a semiring, we can not use the results developed in Section 4.4. Therefore, we present an additional proof for the log-norm distance.

Lemma C.8.1. Given $r \geq 1$, $\mu > 0$ and $c : E \mapsto \mathbb{R}^+$ then

$$\text{LN}(s, t) = -\frac{1}{\mu} \left(\log \left(\mathbb{E}_{\varphi \sim \mathcal{P}_{st}^h} \left[\exp(-\mu c(\|\varphi\|_r)) \right] \right) + \log \left(\mathbb{E}_{\varphi \sim \mathcal{P}_{ts}^h} \left[\exp(-\mu c(\|\varphi\|_r)) \right] \right) \right) \quad (\text{C.39})$$

defines a distance over the vertices of the graph.

Proof: The symmetry, non-negativity and the equality $\text{LN}(s, s) = 0$ are trivial consequences of the definition. Moreover, if we assign to each $e \in E$ a cost $c(e) > 0$, then $c(\|\varphi\|_r) > 0$ for any path φ . Consequently, $\text{LN}(s, t) > 0$ for $s \neq t$.

From now on, we focus on the triangle inequality. We will show the triangle inequality for the terms with the paths in \mathcal{P}_{st}^h since the case for \mathcal{P}_{ts}^h is analogous. Hence, we claim

$$\begin{aligned} & -\frac{1}{\mu} \log \left(\mathbb{E}_{\varphi \sim \mathcal{P}_{st}^h} \left[\exp(-\mu \|c(\varphi)\|_r) \right] \right) \\ & \leq -\frac{1}{\mu} \log \left(\mathbb{E}_{\varphi \sim \mathcal{P}_{sq}^h} \left[\exp(-\mu \|c(\varphi)\|_r) \right] \right) - \frac{1}{\mu} \log \left(\mathbb{E}_{\varphi \sim \mathcal{P}_{qt}^h} \left[\exp(-\mu \|c(\varphi)\|_r) \right] \right) \end{aligned} \quad (\text{C.40})$$

Equation (C.40) is equivalent to

$$\mathbb{E}_{\varphi \sim \mathcal{P}_{st}^h} \left[\exp(-\mu \|c(\varphi)\|_r) \right] \geq \mathbb{E}_{\varphi \sim \mathcal{P}_{sq}^h} \left[\exp(-\mu \|c(\varphi)\|_r) \right] \mathbb{E}_{\varphi \sim \mathcal{P}_{qt}^h} \left[\exp(-\mu \|c(\varphi)\|_r) \right] \quad (\text{C.41})$$

after applying $-\frac{1}{\mu} \log(\cdot)$ to both sides of (C.41). Next, in order to isolate the terms on the right-hand side of (C.41), we separate the set of hitting paths from s to t into those that cross the third node q and those that do not.

$$\begin{aligned}
& \mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} \left[\exp \left(-\mu \|c(\wp)\|_r \right) \right] = \sum_{\wp \in \mathcal{P}_{st}^h} \Pr(\wp) \exp \left(-\mu \|c(\wp)\|_r \right) \\
& \stackrel{*}{\geq} \sum_{\substack{\wp_1 \in \mathcal{P}_{sq}^h \\ t \notin \wp_1}} \sum_{\wp_2 \in \mathcal{P}_{qt}^h} \Pr(\wp_1) \Pr(\wp_2) \exp \left(-\mu \|c(\wp_1)\|_r \right) \exp \left(-\mu \|c(\wp_2)\|_r \right) \\
& + \sum_{\substack{\wp \in \mathcal{P}_{st}^h \\ q \notin \wp}} \Pr(\wp) \exp \left(-\mu \|c(\wp)\|_r \right) \\
& = \left(\sum_{\substack{\wp_1 \in \mathcal{P}_{sq}^h \\ t \notin \wp_1}} \Pr(\wp_1) \exp \left(-\mu \|c(\wp_1)\|_r \right) \right) \left(\sum_{\wp_2 \in \mathcal{P}_{qt}^h} \Pr(\wp_2) \exp \left(-\mu \|c(\wp_2)\|_r \right) \right) \\
& + \sum_{\substack{\wp \in \mathcal{P}_{st}^h \\ q \notin \wp}} \Pr(\wp) \exp \left(-\mu \|c(\wp)\|_r \right) \\
& = \left(\sum_{\wp_1 \in \mathcal{P}_{sq}^h} \Pr(\wp_1) \exp \left(-\mu \|c(\wp_1)\|_r \right) - \sum_{\substack{\wp_1 \in \mathcal{P}_{sq}^h \\ t \in \wp_1}} \Pr(\wp_1) \exp \left(-\mu \|c(\wp_1)\|_r \right) \right) \\
& \times \left(\sum_{\wp_2 \in \mathcal{P}_{qt}^h} \Pr(\wp_2) \exp \left(-\mu \|c(\wp_2)\|_r \right) \right) + \sum_{\substack{\wp \in \mathcal{P}_{st}^h \\ q \notin \wp}} \Pr(\wp) \exp \left(-\mu \|c(\wp)\|_r \right) \quad (\text{C.42}) \\
& = \left(\sum_{\wp_1 \in \mathcal{P}_{sq}^h} \Pr(\wp_1) \exp \left(-\mu \|c(\wp_1)\|_r \right) \right) \left(\sum_{\wp_2 \in \mathcal{P}_{qt}^h} \Pr(\wp_2) \exp \left(-\mu \|c(\wp_2)\|_r \right) \right) \\
& - \left(\sum_{\substack{\wp_1 \in \mathcal{P}_{sq}^h \\ t \in \wp_1}} \Pr(\wp_1) \exp \left(-\mu \|c(\wp_1)\|_r \right) \right) \left(\sum_{\wp_2 \in \mathcal{P}_{qt}^h} \Pr(\wp_2) \exp \left(-\mu \|c(\wp_2)\|_r \right) \right) \\
& + \sum_{\substack{\wp \in \mathcal{P}_{st}^h \\ q \notin \wp}} \Pr(\wp) \exp \left(-\mu \|c(\wp)\|_r \right) \\
& = \mathbb{E}_{\wp \sim \mathcal{P}_{sq}^h} \left[\exp \left(-\mu \|c(\wp)\|_r \right) \right] \mathbb{E}_{\wp \sim \mathcal{P}_{qt}^h} \left[\exp \left(-\mu \|c(\wp)\|_r \right) \right] \\
& - \left(\sum_{\substack{\wp_1 \in \mathcal{P}_{sq}^h \\ t \in \wp_1}} \Pr(\wp_1) \exp \left(-\mu \|c(\wp_1)\|_r \right) \right) \left(\sum_{\wp_2 \in \mathcal{P}_{qt}^h} \Pr(\wp_2) \exp \left(-\mu \|c(\wp_2)\|_r \right) \right) \\
& + \sum_{\substack{\wp \in \mathcal{P}_{st}^h \\ q \notin \wp}} \Pr(\wp) \exp \left(-\mu \|c(\wp)\|_r \right)
\end{aligned}$$

In * we have used the fact that

$$\sqrt[r]{\sum_{i \in I_1} x_i^r + \sum_{i \in I_2} y_i^r} \leq \sqrt[r]{\sum_{i \in I_1} x_i^r} + \sqrt[r]{\sum_{i \in I_2} y_i^r}.$$

and that $\exp(-x)$ is a decreasing function. Note that for $r = 1$ equality holds.

In order to show (C.41) from (C.42) we need to prove

$$\begin{aligned} & \sum_{\substack{\wp \in \mathcal{P}_{st}^h \\ q \notin \wp}} \Pr(\wp) \exp\left(-\mu \|c(\wp)\|_r\right) \\ & \geq \left(\sum_{\substack{\wp \in \mathcal{P}_{sq}^h \\ t \in \wp}} \Pr(\wp) \exp\left(-\mu \|c(\wp)\|_r\right) \right) \left(\sum_{\wp \in \mathcal{P}_{qt}^h} \Pr(\wp) \exp\left(-\mu \|c(\wp)\|_r\right) \right) \end{aligned} \quad (\text{C.43})$$

Let $\wp_1 \odot \wp_2$ denote the concatenation of paths. Since $c(e) > 0$ we deduce

$$c(\wp_1 \odot \wp_2) = \sqrt[r]{\sum_{e \in \wp_1 \odot \wp_2} (c(e))^r} \geq \sqrt[r]{\sum_{e \in \wp_1} (c(e))^r} = c(\wp_1).$$

Thus,

$$\begin{aligned} & \sum_{\substack{\wp \in \mathcal{P}_{sq}^h \\ t \in \wp}} \Pr(\wp) \exp\left(-\mu \|c(\wp)\|_r\right) = \left(\sum_{\substack{\wp_1 \in \mathcal{P}_{st}^h \\ q \notin \wp_1}} \sum_{\wp_2 \in \mathcal{P}_{tq}^h} \Pr(\wp_1) \Pr(\wp_2) \exp\left(-\mu c(\wp_1 \odot \wp_2)\right) \right) \\ & \leq \left(\sum_{\substack{\wp_1 \in \mathcal{P}_{st}^h \\ q \notin \wp_1}} \sum_{\wp_2 \in \mathcal{P}_{tq}^h} \Pr(\wp_1) \Pr(\wp_2) \exp\left(-\mu c(\wp_1)\right) \right) = \sum_{\substack{\wp_1 \in \mathcal{P}_{st}^h \\ q \notin \wp_1}} \Pr(\wp_1) \exp\left(-\mu c(\wp_1)\right) \end{aligned} \quad (\text{C.44})$$

Since $\|c(\wp)\|_r > 0$ for any $\wp \in \mathcal{P}_{ij}^h$ for any vertices $i \neq j$. Therefore,

$$\sum_{\wp \in \mathcal{P}_{st}^h} \Pr(\wp) \exp\left(-\mu \|c(\wp)\|_r\right) < \sum_{\wp \in \mathcal{P}_{st}^h} \Pr(\wp) = 1. \quad (\text{C.45})$$

Then equation (C.43) follows from (C.44) and (C.45). \square

Note that, as a consequence of the previous theorem, the triangle inequality of the exp-max, log-max and exp-norm limits exposed in Table 4.1 is also satisfied. Indeed, by taking the corresponding limits in both sides of the log-norm triangle inequality, the inequality will still hold.

C.9 Log-Norm Distance and the Randomized Shortest Paths

In this section we will relate the log-norm distance with the Helmholtz free energy, following the same reasoning that related the potential distance with the free energy in [95]. Let s and

t be two arbitrary but fixed nodes in the graph G . As defined in [95], the free energy of a thermodynamical system modelled by the probability distribution \Pr_{st} and with temperature $T = 1/\mu$ is given by

$$\Phi(\Pr_{st}) = \sum_{\wp \in \mathcal{P}_{st}^h} \Pr_{st}(\wp) c(\wp) + \frac{1}{\mu} \text{KL}(\Pr_{st}, \Pr^{\text{ref}}), \quad (\text{C.46})$$

where $\Pr^{\text{ref}}(\wp) = \prod_{e \in \wp} p_e$ is the probability that a path $\wp \in \mathcal{P}_{st}^h$ is generated by a random walker and KL is the Kullback-Leibler divergence. In our setting, the cost of a path will be given by $\|c(\wp)\|_r$. Therefore, we use the following expression for the free energy

$$\Phi_r(\Pr_{st}) = \sum_{\wp \in \mathcal{P}_{st}^h} \Pr_{st}(\wp) \|c(\wp)\|_r + \frac{1}{\mu} \text{KL}(\Pr_{st}, \Pr^{\text{ref}}). \quad (\text{C.47})$$

We will show that the symmetrized minimum free energy between two nodes s and t (free energy distance in [95]) coincides with the log-norm distance, i.e.

$$\text{LN}(s, t) = \Phi_r(\Pr_{st}) + \Phi_r(\Pr_{ts}).$$

First, we define the probability distribution over the hitting paths from s to t as the one that minimizes the free energy

$$\Pr_{st}^*(\cdot) := \arg \min_{\Pr(\cdot)} \sum_{\wp \in \mathcal{P}_{st}^h} \Pr(\wp) \|c(\wp)\|_r + \frac{1}{\mu} \text{KL}(\Pr_{st}, \Pr^{\text{ref}}). \quad (\text{C.48})$$

It can be easily checked that the minimizer is given by the following Gibbs probability distribution [95]

$$\Pr_{st}^*(\wp) = \frac{\Pr^{\text{ref}}(\wp) \exp(-\mu \|c(\wp)\|_r)}{\sum_{\hat{\wp} \in \mathcal{P}_{st}^h} \Pr^{\text{ref}}(\hat{\wp}) \exp(-\mu \|c(\hat{\wp})\|_r)}. \quad (\text{C.49})$$

If we now compute the KL-divergence between \Pr_{st}^* and \Pr^{ref} we obtain

$$\begin{aligned}
\text{KL}(\Pr_{st}^*, \Pr^{\text{ref}}) &= \sum_{\wp \in \mathcal{P}_{st}^h} \Pr_{st}^*(\wp) \log \left(\frac{\Pr_{st}^*(\wp)}{\Pr^{\text{ref}}(\wp)} \right) \\
&= \sum_{\wp \in \mathcal{P}_{st}^h} \Pr_{st}^*(\wp) \log \left(\frac{\Pr^{\text{ref}}(\wp) \exp(-\mu \|c(\wp)\|_r)}{\sum_{\hat{\wp} \in \mathcal{P}_{st}^h} \Pr^{\text{ref}}(\hat{\wp}) \exp(-\mu \|c(\hat{\wp})\|_r)} \right) \\
&\quad - \sum_{\wp \in \mathcal{P}_{st}^h} \Pr_{st}^*(\wp) \log(\Pr^{\text{ref}}(\wp)) \\
&= \sum_{\wp \in \mathcal{P}_{st}^h} \Pr_{st}^*(\wp) \log(\Pr^{\text{ref}}(\wp)) - \mu \sum_{\wp \in \mathcal{P}_{st}^h} \Pr_{st}^*(\wp) \|c(\wp)\|_r \\
&\quad - \log \left(\sum_{\hat{\wp} \in \mathcal{P}_{st}^h} \Pr^{\text{ref}}(\hat{\wp}) \exp(-\mu \|c(\hat{\wp})\|_r) \right) - \sum_{\wp \in \mathcal{P}_{st}^h} \Pr_{st}^*(\wp) \log(\Pr^{\text{ref}}(\wp)) \\
&= -\mu \sum_{\wp \in \mathcal{P}_{st}^h} \Pr_{st}^*(\wp) \|c(\wp)\|_r - \log \left(\sum_{\wp \in \mathcal{P}_{st}^h} \Pr^{\text{ref}}(\wp) \exp(-\mu \|c(\wp)\|_r) \right)
\end{aligned} \tag{C.50}$$

Combining this result with (C.47) it follows that

$$\Phi_r(\Pr_{st}^*) = -\frac{1}{\mu} \log \left(\sum_{\wp \in \mathcal{P}_{st}^h} \Pr^{\text{ref}}(\wp) \exp(-\mu \|c(\wp)\|_r) \right).$$

Finally, symmetrizing this expression we obtain the log-norm distance.

C.10 Exp-Max and Log-Max Metric Computation

Currently, there does not exist any efficient algorithm to compute the log-norm distance, LN, in its general form. Nonetheless, we briefly sketch here a possible algorithm to compute the novel exp-max distance (EM) that arises as a limit case of LN (see Table 4.1).

$$\text{EM}(s, t) = \underbrace{\mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} \left[\max_{e \in \wp} c(e) \right]}_{\text{EM}_L(s, t)} + \underbrace{\mathbb{E}_{\wp \sim \mathcal{P}_{ts}^h} \left[\max_{e \in \wp} c(e) \right]}_{\text{EM}_R(s, t)} \tag{C.51}$$

Let $G = (V, E)$ be a graph, $\ell(E)$ be the set of edge costs instantiated by the graph G , and $\mathcal{P}_{st}^h(c)$ be the set of paths with maximum cost equal to c :

$$\ell(E) := \{c(e) : e \in E\} \tag{C.52}$$

$$\mathcal{P}_{st}^h(c) := \{\wp \in \mathcal{P}_{st}^h : c = \max_{e \in \wp} c(e)\} \tag{C.53}$$

We can decompose the left summand of EM as

$$\begin{aligned} \text{EM}_L(s, t) &= \mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} \left[\max_{e \in \wp} c(e) \right] = \sum_{\wp \in \mathcal{P}_{st}^h} \Pr(\wp) \max_{e \in \wp} c(e) \\ &= \sum_{c \in \ell(E)} \sum_{\wp \in \mathcal{P}_{st}^h(c)} \Pr(\wp) \max_{e \in \wp} c(e) = \sum_{c \in \ell(E)} c \Pr(\wp \in \mathcal{P}_{st}^h(c)). \end{aligned}$$

Let $P_{\square c} := \Pr(\wp \in \cup_{c' \square c} \mathcal{P}_{st}^h(c'))$ with $\square \in \{<, \leq\}$. Thus,

$$\Pr(\wp \in \mathcal{P}_{st}^h(c)) = P_{\leq c} - P_{< c} \quad (\text{C.54})$$

can be computed in closed form, since $P_{< c}$ ($P_{\leq c}$) is the probability of reaching t from s without traversing an edge with lower (or equal) cost than c . This is equal to the absorption probability of t , which can be computed analytically by solving a linear system (see Theorem 2.4.1), once extra absorbing nodes have been set on the edges with higher (or equal) cost than c (see Figure C.1). The computational cost of this algorithm scales with $|\ell(c)|$. To reduce the computational cost, we suggest to bin the edge costs coarsely.

Analogously, one can decompose the left summand of the log-max distance (LM) (see Table 4.1) and approximate it in a similar form:

$$\text{LM}_L(s, t) = -\frac{1}{\mu} \log \left(\mathbb{E}_{\wp \sim \mathcal{P}_{st}^h} \left[e^{-\mu \max_{e \in \wp} c(e)} \right] \right) = -\frac{1}{\mu} \log \left(\sum_{c \in \ell(E)} e^{-\mu c} \Pr(\wp \in \mathcal{P}_{st}^h(c)) \right). \quad (\text{C.55})$$

We have shown that one can compute these particular limit instances of the log-norm distance.

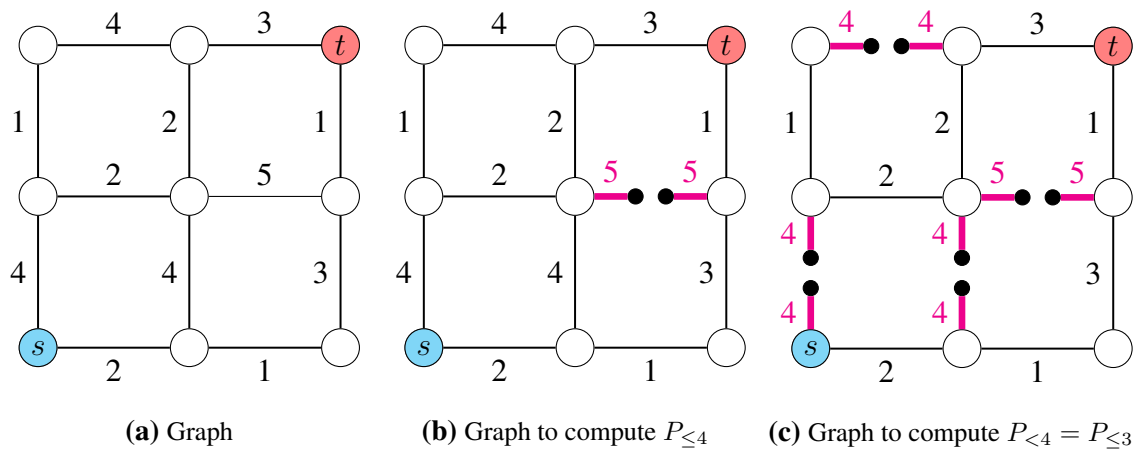


Figure C.1. Computation Ranked Absorption Probability. C.1a) Graph with edge costs and two marked nodes s and t . C.1b) Graph with two absorbing nodes (black nodes). Each path connecting s and t that has a cost $\|c(\varphi)\|_{\infty} = \max_{e \in \varphi} c_e$ higher than 4 must contain the edge with cost equal to 5. By adding artificial absorbing nodes to this edge, any random walker that crosses this edge will be absorbed. We add two absorbing nodes per edge to account for the directions of the edges, which could have different costs. Absorption probability at node t by a random walker starting at node s in this graph is equal to the probability of sampling a path with cost lower or equal than 4, i.e., $P_{\leq 4}$. C.1c) Analogously we can compute $P_{<4}$ if we add absorbing nodes in all edges with cost higher or equal than 4. Note that since there are no edges with cost in between 3 and 4, we have that $P_{<4} = P_{\leq 3}$.

Appendix D

Central Spanning Tree

D.1 Stability Examples

In this section, we show how stable the BCST and CST for different α values are. We sample 1000 points uniformly from uniform distributions over different supports and perturb them by adding zero centered Gaussian noise. We generate two perturbations and show how the tree structure evolves across different values. See Figures D.1 D.2 and D.3. As we increase the value of α , the trees exhibit a more pronounced "star-shaped" pattern and enhanced stability. The parameter α provides a trade-off mechanism between preserving the structure of the data and ensuring the stability of the resulting tree.

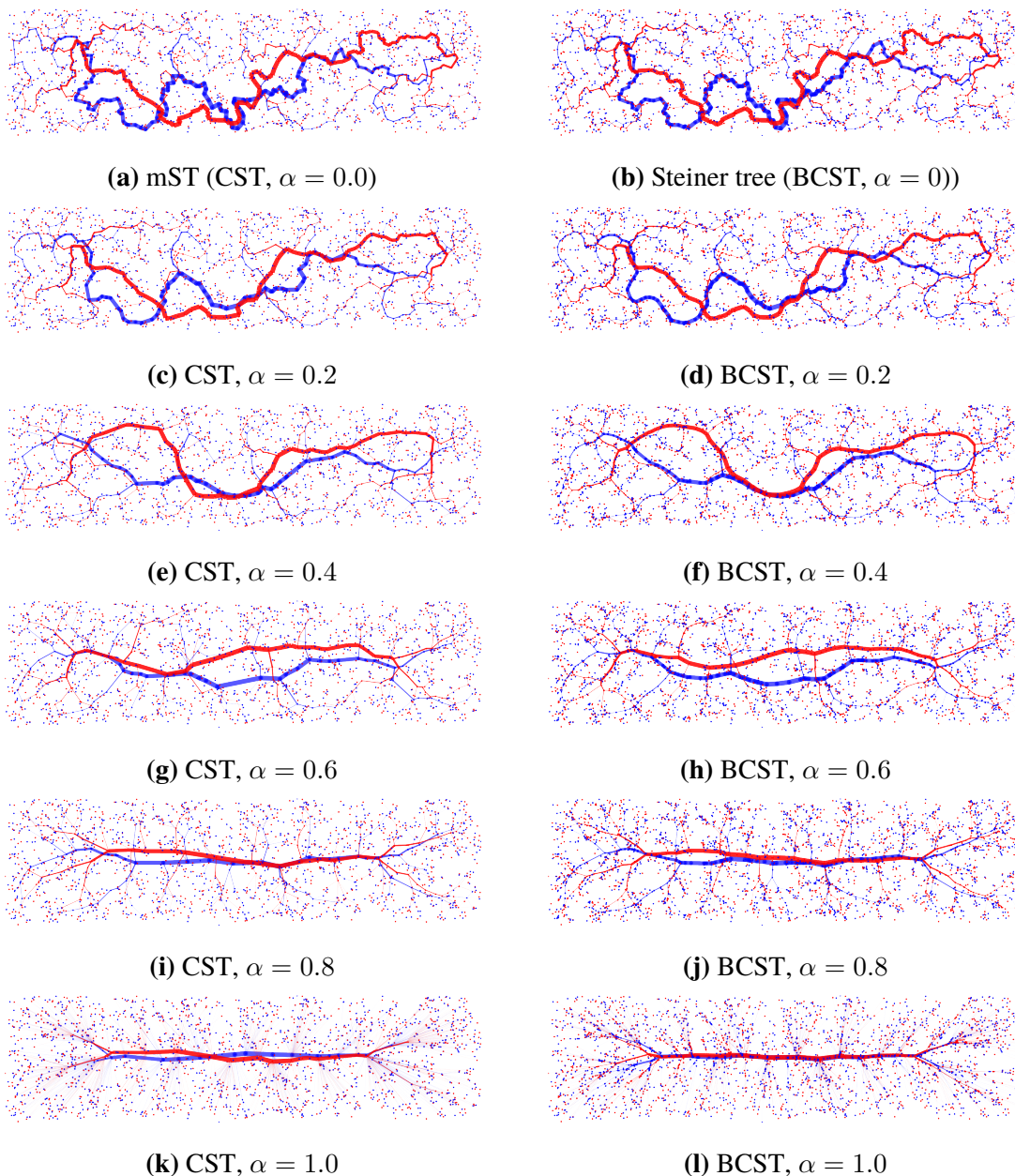


Figure D.1. (B)CST Examples from Rectangle Uniform Distribution. CST and BCST are computed for two perturbed instances generated by adding zero-centered Gaussian noise to points derived from a common sample uniformly taken within a rectangle. (B)CST for higher α values are more robust to noise and adhere to large scale structure in the data better. The width of each edge is proportional to its centrality. All trees except for the mST were computed using the heuristic proposed in Section 5.6.2.

D.2 Reinterpreting CST as a Minimum Concave Cost Flow

In this section, we will pose the central spanning tree (CST) problem as a minimum concave cost network flow (MCCNF) problem. The MCCNF problem minimizes the transportation cost of a commodity from sources to sinks. Here, the edge costs are modeled by concave

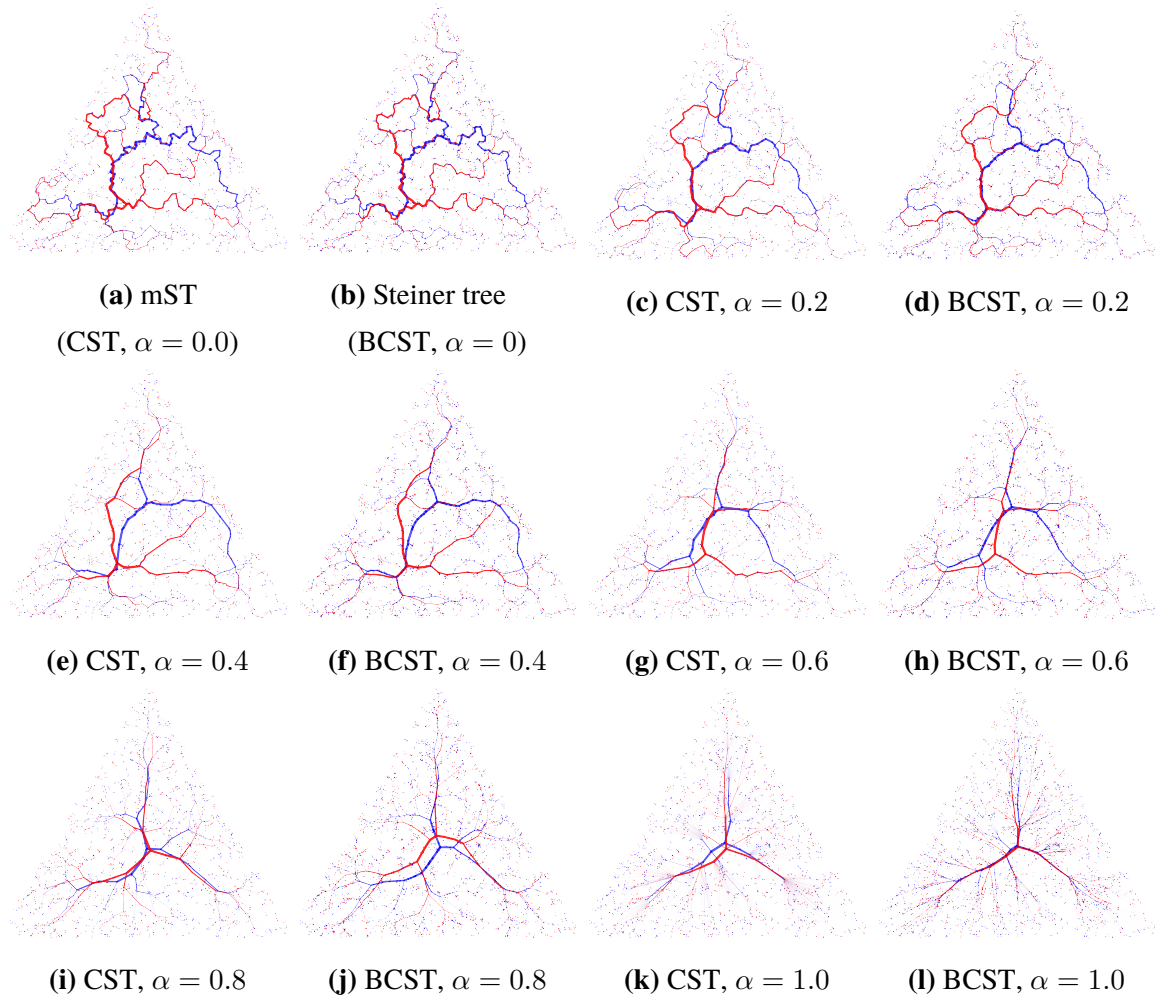


Figure D.2. (B)CST Examples from Triangle Uniform Distribution. CST and BCST are computed for two perturbed instances generated by adding zero-centered Gaussian noise to points derived from a common sample uniformly taken within a triangle. (B)CST for higher α values are more robust to noise and adhere to large scale structure in the data better. The width of each edge is proportional to its centrality. All trees except for the mST were computed using the heuristic proposed in Section 5.6.2.

functions that depend on the edge flow. Formally, given a demand vector $\mu \in \mathbb{R}^N$ with $\sum_{i=1}^N \mu_i = 0$ and a network $G = (V, E)$ with N nodes, we define the MCCNF problem as

$$\begin{aligned}
 & \min_f \sum_{ij \in E} C_{ij}(f_{ij}), \quad \text{subject to} \\
 & \sum_{(i,j) \in E} f_{ij} - \sum_{(j,i) \in E} f_{ji} = \mu_i, \quad \forall i \in V. \\
 & f_{ij} \geq 0
 \end{aligned} \tag{D.1}$$

In the equation, f_{ij} represents the flow associated with edge (i, j) and C_{ij} is a concave function dependent on f_{ij} which determines the cost of the edge (i, j) . Note that the network

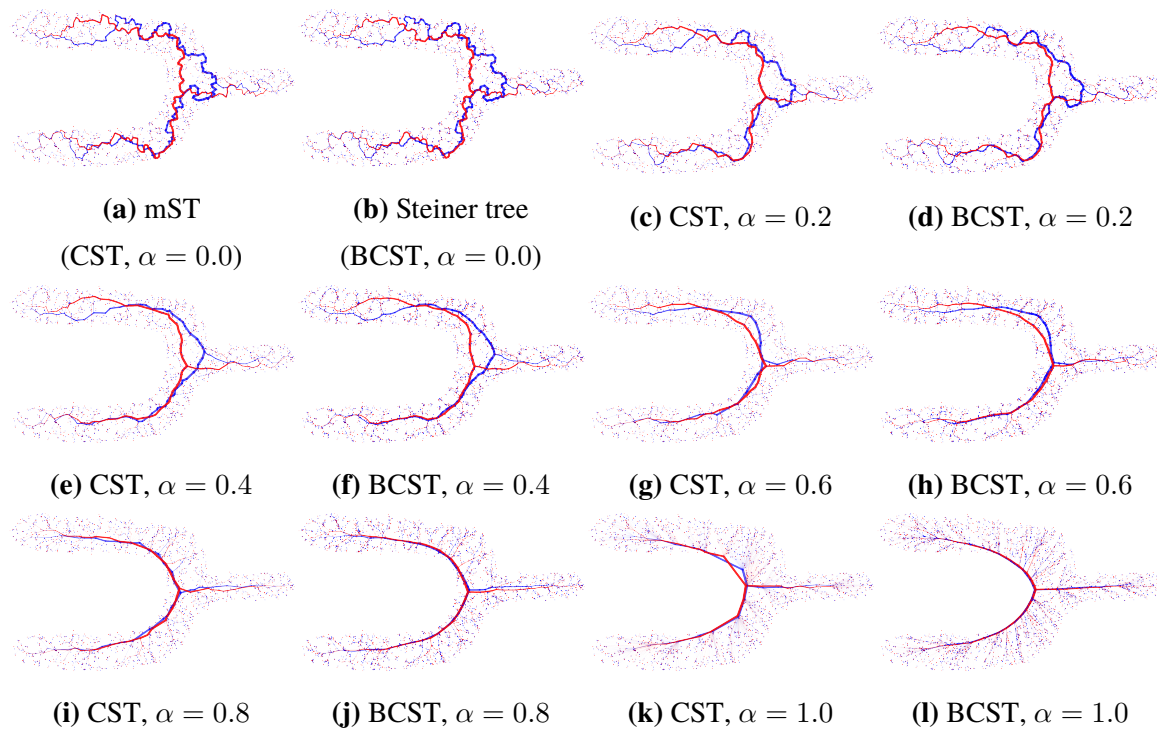


Figure D.3. (B)CST Examples from Non-Convex Uniform Distribution. CST and BCST are computed for two perturbed instances generated by adding zero-centered Gaussian noise to points derived from a common sample uniformly taken within a non convex shape. (B)CST for higher α values are more robust to noise and adhere to large scale structure in the data better. The width of each edge is proportional to its centrality. All trees except for the mST were computed using the heuristic proposed in Section 5.6.2.

defined by the flow, i.e. by the edges with $f_{ij} > 0$, does not necessarily have to be a tree. We will refer to nodes with negative demands as sources and nodes with positive demands as sinks.

To be able to represent the CST problem as an MCCNF, we need to identify the terms m_e as flows. Since the function $(m_e(1 - m_e))^\alpha c_e$ is concave for $\alpha \in [0, 1]$, it will follow that the CST is an instance of the MCCNF problem.

Next, we will show how the m_e can be interpreted as the flow along an edge of a particular single source flow problem. Consider a graph with N nodes, where there is one source node s with a mass $(N - 1)/N$ that needs to be transported to the rest of the nodes. Each sink

node has a demand of $1/N$ mass. Thus, (D.1) becomes

$$\begin{aligned}
& \min_x \sum_{ij \in E} c_{ij} (f_{ij}(1 - f_{ij}))^\alpha, \quad \text{subject to} \\
& \sum_{(j,s) \in E} f_{js} - \sum_{(s,j) \in E} f_{sj} = \frac{N-1}{N}, \\
& \sum_{(i,j) \in E} f_{ij} - \sum_{(j,i) \in E} f_{ji} = \frac{1}{N}, \quad \forall i \neq s \\
& f_{ij} \geq 0
\end{aligned} \tag{D.2}$$

For single source flow problems, it is well-known that the optimal solutions can only form trees [188]. We will show that for this particular problem, f_{ij} is equal to m_{ij} for any tree. Recall that for a given tree T , the value m_e associated with an edge e was defined as the number of nodes that lie on one of the sides of e divided by the total number of nodes. Although for our purposes the chosen side of the edge is arbitrary, since the objective function is symmetrized thanks to being multiplied by $(1 - m_e)$, that is not the case when we want to interpret it as a flow. However, which side to choose will be canonically determined by the flow direction.

Let T be a feasible solution of (D.2), i.e. a tree. The flow f_{ij} at edge (i, j) of T indicates the outgoing mass that is transported through the edge. This mass is equal to the sum of the demands of the nodes that lie in the side of the edge (i, j) indicated by the flow. Given that each node has a demand of $1/N$, then the flow f_{ij} is equal to $1/N$ multiplied by the number of nodes in the side in question, that is $f_{ij} = m_{ij}$.

D.2.1 Relation to the Branched Optimal Transport Problem

The branched or irrigation optimal transport (BOT) problem is also a particular MCCNF instance. In the BOT problem, the nodes are embedded in a Euclidean space and also allows for the inclusion of additional Steiner points. It is an extension of the optimal transport problem, distinguished by its diminishing costs which lead to a branching effect by promoting the joint transportation of mass.

Formally, the BOT problem minimizes

$$\begin{aligned}
& \min_{x_B, m_E} \sum_{(i,j) \in E} m_{ij}^\alpha \|x_i - x_j\|_2, \quad \text{subject to} \\
& \sum_{(i,j) \in E} m_{ij} - \sum_{(j,i) \in E} m_{ji} = \mu_i, \quad \forall i \in V \cup B, \\
& m_{ij} \geq 0
\end{aligned} \tag{D.3}$$

where m_{ij} is the flow transported along an edge (i, j) and B and x_B are the set of SPs and

their coordinates, respectively. As before, the vector μ represents the demands of the nodes. The demands of the SPs are set to zero.

In the scenario where there is a single source and all nodes share the same demand, the BCST and BOT problems differ only in the factors that multiply the distances. In the BOT problem, these factors correspond to m_{ij}^α , representing the mass transported along an edge raised to the power of α . In the BCST problem, the factors are given by $(m_{ij}(1 - m_{ij}))^\alpha$, representing the centralities of the edges raised to the power of α . It is worth mentioning, that both problems converge to the Steiner tree problem when $\alpha = 0$.

The primary distinction between the two problems lies in the selection of a source node. In the BOT problem, the selection of the source node determines the optimal topology of the network. However, that is not the case for the BCST problem. Indeed, for the BCST problem, the specific source node chosen is irrelevant due to the symmetrization effect caused by the term $m_e(1 - m_e)$. In other words, the location of the source determines the edge orientation, which then defines the value of m_e . Nonetheless, this effect is nullified when multiplied by $(1 - m_e)$. This independence of the source node choice makes the BCST a more natural extension of the Steiner tree problem, since it does not require the choice of sources and sinks, unlike the BOT problem.

D.3 Equivalence of the CST Problem with $\alpha = 1$ and the Minimum Routing Cost Tree Problem

As mentioned in the main text, the term $m_{ij}(1 - m_{ij})$ is proportional to the betweenness centrality of the edge (i, j) . This centrality quantifies the number of shortest paths that traverse the given edge. Thus, the multiplication of the length of each edge by its frequency in a shortest path is a rearrangement of the sum over all shortest path costs, i.e. the MRCT cost. Formally,

$$\begin{aligned}
 \sum_{i,j \in V \times V} d_T(i, j) &= \sum_{i,j \in V \times V} \sum_{(u,v) \in \wp_{ij}} \|x_u - x_v\| \\
 &= \sum_{(u,v) \in T} \underbrace{|\{\wp_{ij} : (u, v) \in \wp_{ij}, i, j \in V \times V\}|}_{(u,v) \text{ betweenness centrality}} \|x_u - x_v\| \quad (\text{D.4}) \\
 &\propto \sum_{(u,v) \in T} m_{uv}(1 - m_{uv}) \|x_u - x_v\|,
 \end{aligned}$$

where $d_T(i, j)$ is the shortest path distance in tree T between i and j realized by the path \wp_{ij} .

D.4 Limit Cases of the CST/BCST Problems Beyond the Range $\alpha \in [0, 1]$

In this section, we investigate the topologies of the limit cases of the CST as α approaches $\pm\infty$. We will use the following notation.

- N will represent the number of terminals.
- For a given tree T containing edge (x, y) , m_{xy}^T indicates the proportion of nodes (normalized by N) that are reachable from x , once edge (x, y) is removed from T . That is, the normalized number of nodes that lie in the side of x .
- For a given tree T , the term \mathcal{N}_x denotes the set of neighbors of x in T

D.4.1 Proof Theorem 5.2.2

Theorem 5.2.2 states that when a “stronger” variant of the triangle inequality holds, then the optimum solution of the BCST problem is a star tree. We divide the proof into two lemmas, Lemma *D.4.1* that proves it for the CST problem; and Lemma *D.4.2* for the BCST case.

Lemma D.4.1. Given a complete graph G with N nodes, let c_{ij} be the edge-costs of any pair of nodes i, j in the graph. If there exists

$$t \leq \min_{\ell \in [2, N/2]} \frac{\left(\frac{\ell(N-\ell)}{N-1}\right)^\alpha - 1}{\ell - 1}$$

such that

$$c_{kv} + tc_{uv} \geq c_{ku} \tag{D.5}$$

for all triangles in the graph, then there exists an optimum CST evaluated at α which is a star tree.

Proof: We will show that given a tree T , we can always increase the degree of a particular node without increasing the CST-cost. Without loss of generality, we can assume $n \geq 4$, otherwise, any possible tree is a star-tree and the result holds trivially.

Let us assume that T is not a star-tree. Thus, there exist at least two nodes, u and v , with degrees higher than 1, such that they are adjacent to each other. Moreover, we can assume that one of them, say v , is an extreme inner node, meaning that all its neighbors (except u) have degree 1. Without loss of generality, we can assume that $\ell := |\mathcal{N}_v| \leq N/2$, where \mathcal{N}_v is the set of neighbors of v in T . Otherwise, we could have chosen a different extreme inner node. Note that the centrality of the edge (u, v) is $m_{uv}(1 - m_{uv}) = \frac{\ell(N-\ell)}{N^2}$.

We will show that the topology T' which connects all $k \in \mathcal{N}_v \setminus \{u\}$ to u instead of v has a lower CST cost. The only edge centralities affected by this change are those associated

with the edges (u, v) , (u, k) , and (k, v) for all $k \in \mathcal{N}_v \setminus \{u\}$. To compare the costs of the topologies, it suffices to focus on these specific edges.

First, let's determine the values of the centralities for the edges in both trees:

- Normalized centrality of edge (u, v) in T:

$$m_{uv}^T(1 - m_{uv}^T) = \frac{\ell(N - \ell)}{N^2}$$

- Normalized centrality of edge (u, v) in T':

$$m_{uv}^{T'}(1 - m_{uv}^{T'}) = \frac{N - 1}{N^2}$$

Note that v has become a leaf of T'.

- Normalized centrality of edge (k, v) in T and centrality of edge (k, u) in T' for all $k \in \mathcal{N}_v \setminus \{u\}$:

$$m_{kv}^T(1 - m_{kv}^T) = \frac{N - 1}{N^2} = m_{ku}^{T'}(1 - m_{ku}^{T'})$$

These nodes are leaves in both trees.

The difference between the costs of the topologies is

$$\begin{aligned} \text{CST}(T) - \text{CST}(T') &= c_{uv} \left(\frac{\ell(N - \ell)}{N^2} \right)^\alpha + \sum_{\substack{k \in \mathcal{N}_v \\ k \neq u}} c_{kv} \left(\frac{N - 1}{N^2} \right)^\alpha \\ &\quad - c_{uv} \left(\frac{N - 1}{N^2} \right)^\alpha - \sum_{\substack{k \in \mathcal{N}_v \\ k \neq u}} c_{ku} \left(\frac{N - 1}{N^2} \right)^\alpha \\ &= c_{uv} \left(\left(\frac{\ell(N - \ell)}{N^2} \right)^\alpha - \left(\frac{N - 1}{N^2} \right)^\alpha \right) + \sum_{\substack{k \in \mathcal{N}_v \\ k \neq u}} (c_{kv} - c_{ku}) \left(\frac{N - 1}{N^2} \right)^\alpha \\ &= \sum_{\substack{k \in \mathcal{N}_v \\ k \neq u}} \left[\frac{c_{uv} \left(\left(\frac{\ell(N - \ell)}{N^2} \right)^\alpha - \left(\frac{N - 1}{N^2} \right)^\alpha \right)}{\ell - 1} + (c_{kv} - c_{ku}) \left(\frac{N - 1}{N^2} \right)^\alpha \right] \\ &= \left(\frac{N - 1}{N^2} \right)^\alpha \sum_{\substack{k \in \mathcal{N}_v \\ k \neq u}} \left[\frac{c_{uv} \left(\left(\frac{\ell(N - \ell)}{N - 1} \right)^\alpha - 1 \right)}{\ell - 1} + (c_{kv} - c_{ku}) \right] \end{aligned} \quad (\text{D.6})$$

Thus the decrease in cost will be positive if each term in the summand of the last equality of (D.6) is positive, namely

$$c_{kv} + \frac{\left(\left(\frac{\ell(N - \ell)}{N - 1} \right)^\alpha - 1 \right)}{\ell - 1} c_{uv} \geq c_{ku}, \quad (\text{D.7})$$

which holds by assumption. Therefore, T' will have a lower cost. Repeating this process, we can always decrease the cost till we form a star tree. \square

The next result extends Lemma D.4.1 to be applicable to the BCST problem. In contrast to the CST case, the BCST involves Steiner points, which must be treated differently. Lemma D.4.2 shows that if the "strong" triangle inequality holds, we can collapse sequentially all Steiner points while decreasing the BCST cost of a tree T.

Lemma D.4.2. Consider a solution T of the BCST problem with N terminals. Let c_{ij} be the edge-costs of any pair of nodes i, j in the graph (Steiner or terminals). If there exists

$$t \leq \min_{\ell \in [2, N/2]} \frac{\left(\frac{\ell(N-\ell)}{N-1}\right)^\alpha - 1}{\ell - 1}$$

such that

$$c_{kv} + tc_{uv} \geq c_{ku} \quad (\text{D.8})$$

for all triangles, then there exists a star tree with lower cost.

Proof: Analogously to Lemma D.4.1 we will show that given a tree T , we can always increase the degree of a particular node without increasing the BCST-cost.

Let us assume that T is not a star-tree. Thus, there exist at least two nodes, u and v , with degree higher than 1 which are adjacent to each other. Moreover, we can assume that one of them, say v , is an extreme inner node, meaning that all its neighbors (except u) have degree 1. Without loss of generality, we can assume that $m_{uv} := \frac{\ell}{N^2} \leq \frac{1}{2N}$. Otherwise, we could have chosen a different extreme inner node.

If v is a terminal node, we can apply the same reasoning as in Lemma D.4.1 to increase the degree of u . Let us assume then that v is a Steiner point. In this case, we will construct a new topology T' by collapsing v with u . This implies that the edge (u, v) will disappear and that all $k \in \mathcal{N}_v \setminus \{u\}$ will be connected to u . In this case the normalized centralities of the edges are not changed.

- Normalized centrality of edge (u, v) in T :

$$m_{uv}^T(1 - m_{uv}^T) = \frac{\ell(N - \ell)}{N^2}$$

- Edge (u, v) is not anymore in T' :
- Normalized centrality of edge (k, v) in T and centrality of edge (k, u) in T' for all $k \in \mathcal{N}_v \setminus \{u\}$:

$$m_{kv}^T(1 - m_{kv}^T) = \frac{N - 1}{N^2} = m_{ku}^{T'}(1 - m_{ku}^{T'})$$

These nodes are leaves in both trees.

The difference between the costs of the topologies is

$$\begin{aligned} \text{CST}(T) - \text{CST}(T') &= c_{uv} \left(\frac{\ell(N - \ell)}{N^2}\right)^\alpha + \sum_{\substack{k \in \mathcal{N}_v \\ k \neq u}} c_{kv} \left(\frac{N - 1}{N^2}\right)^\alpha - \sum_{\substack{k \in \mathcal{N}_v \\ k \neq u}} c_{ku} \left(\frac{N - 1}{N^2}\right)^\alpha \\ &= c_{uv} \left(\frac{\ell(N - \ell)}{N^2}\right)^\alpha + \sum_{\substack{k \in \mathcal{N}_v \\ k \neq u}} (c_{kv} - c_{ku}) \left(\frac{N - 1}{N^2}\right)^\alpha \\ &= \sum_{\substack{k \in \mathcal{N}_v \\ k \neq u}} \left[\frac{c_{uv} \left(\frac{\ell(N - \ell)}{N^2}\right)^\alpha}{\ell - 1} + (c_{kv} - c_{ku}) \left(\frac{N - 1}{N^2}\right)^\alpha \right] \\ &= \left(\frac{N - 1}{N^2}\right)^\alpha \sum_{\substack{k \in \mathcal{N}_v \\ k \neq u}} \left[\frac{c_{uv} \left(\frac{\ell(N - \ell)}{N - 1}\right)^\alpha}{\ell - 1} + (c_{kv} - c_{ku}) \right] \end{aligned} \quad (\text{D.9})$$

Thus the decrease in cost will be positive if each term in the summand of the last equality of (D.9) is positive, namely

$$c_{kv} + \frac{\left(\frac{\ell(N-\ell)}{N-1}\right)^\alpha}{\ell-1} c_{uv} \geq c_{ku}, \quad (\text{D.10})$$

which holds by assumption, since

$$\frac{\left(\frac{\ell(N-\ell)}{N-1}\right)^\alpha}{\ell-1} \geq \min_{\ell \in [2, N/2]} \frac{\left(\frac{\ell(N-\ell)}{N-1}\right)^\alpha}{\ell-1} > \min_{\ell \in [2, N/2]} \frac{\left(\frac{\ell(N-\ell)}{N-1}\right)^\alpha - 1}{\ell-1}.$$

Therefore, T' will have a lower cost. Repeating this process, we can always decrease the cost till we form a star tree. \square

Remark D.4.3. Note that Lemma D.4.1 and Lemma D.4.2 state only a sufficient condition, which means that the optimum can be a star tree even if the strong triangle inequality does not hold. Additionally, it is worth to highlight that Lemma D.4.1 also holds true for the CST problem even when the nodes lack embedding in any specific space, allowing for edge costs with arbitrary values.

D.4.2 Proof $h_1(\ell, N, \alpha) > 1$ as N Approaches Infinity, for $\alpha > 1$

Recall that h_1 is defined as

$$h_1(\ell, N, \alpha) := \frac{\left(\left(\frac{\ell(N-\ell)}{N-1}\right)^\alpha - 1\right)}{\ell-1} = \frac{\left(1 + \frac{(\ell-1)(N-\ell-1)}{N-1}\right)^\alpha - 1}{\ell-1}.$$

We will show that for high enough N , $\min_{\ell \in [2, N/2]} h_1(\ell, N, \alpha) > 1$. By leveraging the Mathematica software [83], we can establish that the function $h_1(\ell, N, \alpha)$ exhibits concavity concerning ℓ within the interval $[2, N/2]$ under the conditions $\alpha > 1$ and $N > 3$. Consequently, for fixed values of $\alpha > 1$ and N , the minimum of h is achieved either at $\ell = 2$ or at $\ell = N/2$. Next we show that as N tends to infinity, both evaluations tend towards a value greater than 1.

- When $\ell = 2$ we have

$$h_1(2, N, \alpha) = \left(1 + \frac{N-3}{N-1}\right)^\alpha - 1 \xrightarrow{N \rightarrow \infty} 2^\alpha - 1 > 1 \quad (\text{D.11})$$

- When $\ell = N/2$ we have

$$h_1(N/2, N, \alpha) = \frac{\left(1 + \frac{(N/2-1)^2}{N-1}\right)^\alpha - 1}{N/2-1} \xrightarrow{N \rightarrow \infty} \infty \quad (\text{D.12})$$

Therefore, $h_1(\ell, N, \alpha) > 1$ as N approaches infinity and $\alpha > 1$.

Combining this inequality with Theorem 5.2.2, we conclude that the optimum (B)CST will be a tree as N approaches infinity and $\alpha > 1$.

D.4.3 Computation $\alpha^*(N)$

Given N , recall that $\alpha^*(N)$ is the minimum α at which $h_1(\ell, N, \alpha) > 1$ for all $\ell \in [2, N/2]$. Since h_1 is concave with respect to ℓ when $\alpha > 1$, our focus narrows down to investigating the cases $\ell = 2$ and $\ell = N/2$ —the values where the minima can be attained.

- When $\ell = 2$ we have

$$\begin{aligned} h_1(2, N, \alpha) = \left(1 + \frac{N-3}{N-1}\right)^\alpha - 1 > 1 &\iff \left(1 + \frac{N-3}{N-1}\right)^\alpha > 2 \\ &\iff \alpha \log\left(1 + \frac{N-3}{N-1}\right) > \log(2) \\ &\iff \alpha > \frac{\log(2)}{\log\left(1 + \frac{N-3}{N-1}\right)} \end{aligned} \quad (\text{D.13})$$

- When $\ell = N/2$ we have

$$\begin{aligned} h_1(N/2, N, \alpha) = \frac{\left(1 + \frac{(N/2-1)^2}{N-1}\right)^\alpha - 1}{N/2-1} > 1 &\iff \left(1 + \frac{(N/2-1)^2}{N-1}\right)^\alpha > N/2 \\ &\iff \alpha \log\left(1 + \frac{(N/2-1)^2}{N-1}\right) > \log(N/2) \\ &\iff \alpha > \frac{\log(N/2)}{\log\left(1 + \frac{(N/2-1)^2}{N-1}\right)} \end{aligned} \quad (\text{D.14})$$

Therefore,

$$\alpha^*(N) := \max\left(\frac{\log(2)}{\log\left(1 + \frac{N-3}{N-1}\right)}, \frac{\log(N/2)}{\log\left(1 + \frac{(N/2-1)^2}{N-1}\right)}\right) \quad (\text{D.15})$$

D.4.4 Proof Theorem 5.2.5

In this section we prove Theorem 5.2.5 which states that if a variant of the triangle inequality holds, then the optimum (B)CST tree will be a path as α approaches infinity. First, Lemma D.4.4 will show that if the triangle inequality holds strictly, then for α negative enough the optimum CST will be a path. Corollary D.4.6 demonstrates that when the nodes are embedded in a geodesic space, the triangle inequality does not need to hold strictly for Lemma D.4.4 to be true. Theorem 5.2.5 will also be derived as corollary (Corollary D.4.7).

Lemma D.4.4. Let G be a complete graph with edge-costs satisfying the condition of the strict triangle inequality for every triplet of nodes (u, v, k) , defined as

$$c_{uv} + c_{kv} < c_{ku}$$

As the parameter α approaches negative infinity ($\alpha \rightarrow -\infty$), there exists a Hamiltonian path T_\star in G with a lower CST cost than any other tree T that is not a path.

Proof: We will show that for any node v with degree higher than 2, we can always decrease its degree such that the CST cost of T decreases. By iteratively applying this process, we ensure that the degrees of all nodes will eventually be reduced to at most 2, culminating in the formation of a path.

Let v be a node with degree higher than 3. Let (k, v) and (u, v) be the two edges adjacent to v and assume w.l.o.g that $m_{uv}^T = \min_{i \in \mathcal{N}_v} m_{iv}^T$. Since $m_{kv}^T \geq m_{uv}^T$, we have

$$(m_{kv}^T(1 - m_{kv}^T)) \geq (m_{uv}^T(1 - m_{uv}^T)).$$

We will now demonstrate that the modified topology T' , where node k is connected to node u instead of node v , results in a lower CST cost. The only edge centralities affected by this change are those associated with edges (u, v) , (u, k) , and (k, v) . To compare the costs of the topologies, it suffices to focus on these specific edges.

First, we determine the values of the centralities for these edges in both trees.

- Normalized centrality of edge (u, v) in tree T :

$$m_{uv}^T(1 - m_{uv}^T)$$

- Normalized centrality of edge (u, v) in tree T' :

$$m_{uv}^{T'}(1 - m_{uv}^{T'}) = (m_{uv}^T + m_{kv}^T)(1 - m_{uv}^T - m_{kv}^T)$$

The equality is due to the fact that once k is a neighbor of u , all nodes that were in the same side as k will be now in the same side as u .

- Normalized centrality edge k, v in T and normalized centrality edge k, u in T' :

$$m_{kv}^T(1 - m_{kv}^T) = m_{ku}^{T'}(1 - m_{ku}^{T'})$$

Both u and v lie in the same side of the edges, hence the equality of their normalized centralities.

Note that $m_{uv}^{T'}(1 - m_{uv}^{T'}) > m_{uv}^T(1 - m_{uv}^T)$. Otherwise, it would imply that

$$\begin{aligned} m_{uv}^{T'}(1 - m_{uv}^{T'}) &< m_{uv}^T(1 - m_{uv}^T) \iff \\ (m_{uv}^T + m_{kv}^T)(1 - m_{uv}^T - m_{kv}^T) &< m_{uv}^T(1 - m_{uv}^T) \iff \\ \min(m_{uv}^T + m_{kv}^T, 1 - m_{uv}^T - m_{kv}^T) &< \min(m_{uv}^T, 1 - m_{uv}^T) = m_{uv}^T \end{aligned}$$

Trivially, $m_{uv}^T + m_{kv}^T < m_{uv}^T$ leads to a contradiction since $m_{kv}^T > 0$. Thus, the only possibility is

$$\begin{aligned} 1 - m_{uv}^T - m_{kv}^T < m_{uv}^T &\iff 1 - m_{kv}^T < 2m_{uv}^T \\ &\iff 1 - m_{kv}^T = m_{uv}^T + \sum_{\substack{i \in \mathcal{N}_v \\ i \neq k, i \neq u}} m_{iv}^T < 2m_{uv}^T \\ &\iff \sum_{\substack{i \in \mathcal{N}_v \\ i \neq k, i \neq u}} m_{iv}^T < m_{uv}^T \end{aligned}$$

which is also a contradiction since by assumption $m_{uv}^T = \min_{i \in \mathcal{N}_v} m_{iv}^T$. Now we are able to show that the cost of T' is lower than the one of T

$$\begin{aligned} \text{CST}(T) - \text{CST}(T') &= c_{uv} (m_{uv}^T (1 - m_{uv}^T))^\alpha + c_{kv} (m_{kv}^T (1 - m_{kv}^T))^\alpha \\ &\quad - c_{uv} ((m_{uv}^T + m_{kv}^T) (1 - m_{uv}^T - m_{kv}^T))^\alpha - c_{ku} (m_{kv}^T (1 - m_{kv}^T))^\alpha \\ &= c_{uv} \left((m_{uv}^T (1 - m_{uv}^T))^\alpha - ((m_{uv}^T + m_{kv}^T) (1 - m_{uv}^T - m_{kv}^T))^\alpha \right) + (c_{kv} - c_{ku}) (m_{kv}^T (1 - m_{kv}^T))^\alpha \\ &= (m_{uv}^T (1 - m_{uv}^T))^\alpha \left(c_{uv} - c_{uv} \left(\underbrace{\frac{(m_{uv}^T + m_{kv}^T) (1 - m_{uv}^T - m_{kv}^T)}{m_{uv}^T (1 - m_{uv}^T)}}_{>1} \right)^\alpha + (c_{kv} - c_{ku}) \left(\underbrace{\frac{m_{kv}^T (1 - m_{kv}^T)}{m_{uv}^T (1 - m_{uv}^T)}}_{\geq 1} \right)^\alpha \right). \end{aligned}$$

We can differentiate two cases. If $m_{kv}^T (1 - m_{kv}^T) = m_{uv}^T (1 - m_{uv}^T)$ then

$$\frac{\text{CST}(T) - \text{CST}(T')}{(m_{uv}^T (1 - m_{uv}^T))^\alpha} \xrightarrow{\alpha \rightarrow -\infty} (c_{uv} + c_{kv} - c_{ku}) > 0, \quad (\text{D.16})$$

where we have used the strict triangle inequality. Otherwise, the limit tends to

$$\frac{\text{CST}(T) - \text{CST}(T')}{(m_{uv}^T (1 - m_{uv}^T))^\alpha} \xrightarrow{\alpha \rightarrow -\infty} c_{uv} > 0.$$

Hence, for sufficiently negative values of α , the difference $\text{CST}(T) - \text{CST}(T')$ will be positive. By repeating this process, we can continue reducing the degree of nodes with degree higher than 2 until all nodes have degree at most 2. This process will eventually lead to the formation of a Hamiltonian path with a lower cost than the original tree T . \square

Remark D.4.5. Due to equation (D.16), Lemma D.4.4 required the triangle inequality to hold strictly. However, the strict triangle inequality is not an indispensable for the validity of Lemma D.4.4. Corollary D.4.6 demonstrates that, when the nodes of the graph are embedded in a geodesic metric space (e.g. Euclidean space), the strict triangle inequality becomes unnecessary. This result extends also to the BCST problem.

Nevertheless, the scenario portrayed in Figure D.4 serves as an example where the strict triangle inequality is not satisfied, leading to a non-optimal Hamiltonian path. This illustrates that while the strict triangle inequality may be dispensable under certain conditions, there are instances of arbitrary graphs, as demonstrated in the figure, where it cannot be abandoned.

Corollary D.4.6. Consider the BCST and CST problem where the nodes are embedded in a geodesic metric space. As α tends to negative infinity ($\alpha \rightarrow -\infty$) there exists a Hamiltonian path T_\star with a lower CST/BCST cost than any other tree T that is not a path.

Proof: The reasoning aligns with the exposition in Lemma D.4.4, proving that for any node v with degree exceeding 2, we can always decrease its degree such that the CST/BCST cost of T decreases. Through the iterative application of this process, the degrees of all nodes

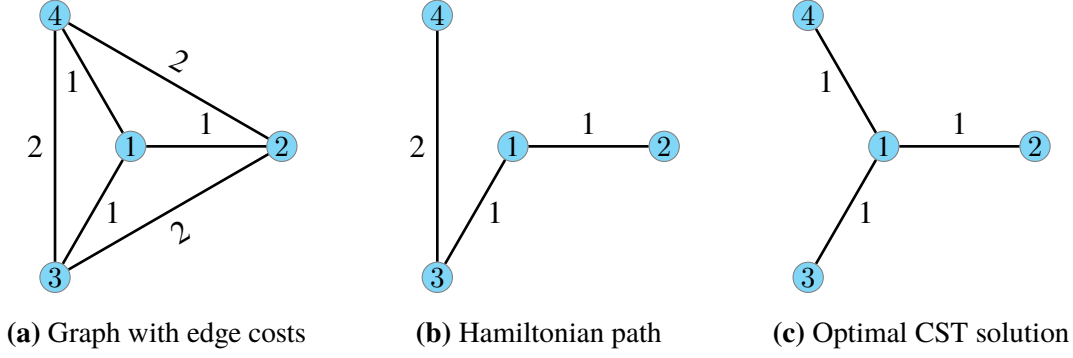


Figure D.4. Necessity of the Triangle Inequality for Path Graph to be (B)CST Optimum as $\alpha \rightarrow -\infty$. D.4a) Graph with edge costs depicted, where the triangle inequality is not strictly satisfied. D.4b) Optimal hamiltonian path with CST cost equal to $\frac{3^\alpha \cdot 3 + 4^\alpha}{16^\alpha}$. D.4c) Optimal CST with cost equal to $\frac{3 \cdot 3^\alpha}{16^\alpha}$. Thus, if the triangle inequality does not strictly hold, the Hamiltonian path will not necessarily be optimal even for sufficiently negative α values

are systematically decreased, ultimately converging to a state where each node has at most a degree of 2, thereby resulting in the formation of a path.

Consider a node v with a degree higher than 2. To apply the logic presented in Theorem D.4.4, it is essential to ensure the ability to select two neighbors such that the triangle inequality holds strictly. Let u , k , and ℓ represent three distinct neighbors of v , and assume that the triangle inequality is an equality between each pair of neighbors and the node v . In other words, we have

$$c_{uv} + c_{vk} = c_{uk}, \quad (\text{D.17})$$

$$c_{uv} + c_{v\ell} = c_{u\ell}, \quad (\text{D.18})$$

$$c_{\ell v} + c_{kv} = c_{\ell k}. \quad (\text{D.19})$$

Starting with (D.17), we conclude that v lies on the geodesic path between u and k . Similarly, from (D.18) and (D.19), we deduce that v is positioned between u and ℓ and also between ℓ and k . Consequently, ℓ is established to be between u and k , as it is situated between u and v and also between k and v .

As a result, we can infer that $c_{\ell k} + c_{\ell u} = c_{uk}$. Summing (D.17) and (D.18) and utilizing the derived equality $c_{\ell k} + c_{\ell u} = c_{uk}$ (owing to the position of ℓ between u and k), we obtain:

$$c_{uv} + c_{vk} + c_{uv} + c_{v\ell} = c_{u\ell} + c_{uk} \rightarrow 2c_{uv} + c_{\ell k} = c_{\ell k} \rightarrow c_{uv} = 0$$

The deduction that $c_{uv} = 0$ implies that nodes u and v occupy the same position and can effectively be considered as a single node. Consequently, we can systematically remove neighbors of v by repeating this process until v attains a degree of 2.

Alternatively, if the triangle inequality must hold strictly for a pair of nodes u , k , and node v , we can, w.l.o.g., assume that node u is chosen such that $m_{uv} = \min_{i \in \mathcal{N}_v} m_{iv}$. In this

scenario, applying the same reasoning as the one presented in Lemma *D.4.4*, we demonstrate that the modified topology T' —where node k is connected to node u instead of node v —results in a lower BCST/CST cost. \square

Lemma *D.4.4* resembles Lemma *D.4.1* in the sense that both lemmas require the satisfaction of a weighted triangle inequality. The following corollary reformulates Lemma *D.4.4*, mirroring the structure found in Lemma *D.4.1*, and accentuates the correlation between the weighted triangle inequality and the number of terminals denoted by N .

Corollary D.4.7 (Theorem 5.2.5). Given a complete graph G with N nodes, let c_{ij} be the edge-costs of any pair of nodes (i, j) . If there exists

$$t \leq \min_{\substack{1 \leq s \leq N-3 \\ 1 \leq \ell \leq \min(s, (N-s)/2-1)}} \frac{(\ell(N-\ell))^\alpha - ((\ell+s)(N-\ell-s))^\alpha}{(s(N-s))^\alpha}$$

such that

$$c_{kv} + tc_{uv} \geq c_{ku}$$

for all triangles in the graph, then there exists an optimum CST evaluated at α which is a Hamiltonian path.

Proof: Lemma *D.4.4* holds if

$$\begin{aligned} & \left(c_{uv} - c_{uv} \left(\frac{(m_{uv}^T + m_{kv}^T)(1 - m_{uv}^T - m_{kv}^T)}{(m_{uv}^T(1 - m_{uv}^T))} \right)^\alpha + (c_{kv} - c_{ku}) \left(\frac{m_{kv}^T(1 - m_{kv}^T)}{m_{uv}^T(1 - m_{uv}^T)} \right)^\alpha \right) > 0 \\ \iff & c_{uv} \left(\frac{(m_{uv}^T(1 - m_{uv}^T))^\alpha - ((m_{uv}^T + m_{kv}^T)(1 - m_{uv}^T - m_{kv}^T))^\alpha}{(m_{kv}^T(1 - m_{kv}^T))^\alpha} \right) + c_{kv} > c_{ku} \quad (\text{D.20}) \end{aligned}$$

where it is assumed that $m_{uv} = \min_{i \in \mathcal{N}_v} m_{iv}$. Thus, let $m_{uv} = \frac{\ell}{N}$ and $m_{kv} = \frac{s}{N}$. Substituting these values into (D.20), we derive the following inequality

$$c_{kv} + \frac{(\ell(N-\ell))^\alpha - ((\ell+s)(N-\ell-s))^\alpha}{(s(N-s))^\alpha} c_{uv} \geq c_{ku}.$$

Note that $\ell \leq \min(s, (N-s)/2-1)$ since $m_{uv} = \min_{i \in \mathcal{N}_v} m_{iv}$. Thus, $m_{uv} = \frac{\ell}{N} \leq m_{kv} = \frac{s}{N}$. On the other hand, since in the proof of Lemma *D.4.4*, v has at least three neighbors (u , k and say p), the $N-s$ nodes not lying in the side corresponding to k of edge (k, v) must be distributed in the side of v . At exception of node v , one part of the remaining $N-s-1$ nodes will be in the side corresponding to u of edge (u, v) and the other part in the side corresponding to p of edge (p, v) . Since m_{uv} must be minimum, the side corresponding to u of edge (u, v) can at most be equal to $\frac{N-s-1}{2N}$.

Notice also that $1 \leq s \leq N-3$, since v has degree at least three and therefore there are at least three nodes (v, u, p) lying in the same side of v with respect to edge (k, v) . Thus, $m_{vk}^T = (1 - m_{kv}^T) = 1 - \frac{s}{N} \geq \frac{3}{N}$, or equivalently $s \leq N-3$. \square

D.5 Exploring the Number of Derivable Topologies from CST and BCST Topologies

D.5.1 Number of BCST Topologies Derivable from a CST Topology

In this section, we explicitly determine the number of topologies of the BCST problem that can be derived from a single CST topology.

To derive a full tree topology T_{BCST} from a CST topology T_{CST} with N terminals, we need to add $N - 2$ SP. In particular, for each terminal node, v , with degree $d_v \geq 2$, we need to spawn $d_v - 1$ SP. Since for k terminal nodes, there exist a total $(2k - 5)!!$ of full tree topologies [151], there are $(2(d_v + 1) - 5)!! = (2d_v - 3)!!$ ways to connect the added SP to the neighbors of v and v itself. Thus the total number of full tree topologies is equal to the number of possible combinations of subtopologies engendered per terminal neighborhood for terminals with degree higher than 2. Formally, this number is equal

$$\prod_{v : d_v \geq 2} (2d_v - 3)!! \quad (\text{D.21})$$

Note that, on the one hand, if all nodes have degree lower or equal than 2, i.e. the tree is a path, then a single full tree topology can be derived. On the other hand, if the original T_{CST} is a star graph, then there is a single graph with degree higher than 2, which is equal to $N - 1$. Thus, the total number of topologies derived from it is equal to $(2N - 5)!!$. This is the total number of possible full tree topologies, hence a star graph can generate any full tree topology. In general, the higher the degree of the nodes in T_{CST} , the higher the number of derivable full tree topologies.

D.5.2 Number of CST Topologies Derivable from a BCST Topology

In this case, we need to collapse each SP to a terminal. The collapse process can be carried out sequentially, where each SP is collapsed to one of its neighboring nodes, until no SP remain. Naively, we might assume that there are 3^{N-2} possible topologies, given that each SP has 3 neighbors available for collapse and there are $N - 2$ SPs. However, this is not the case because some combinations may result in non-valid topologies. For instance, if all SPs collapse with neighbors that are also SP, none of the SP will be collapsed with a terminal node.

Before providing the formula for the number of CST topologies that can be derived from a full tree topology, let's introduce the All Minors Matrix Tree Theorem [30], which is necessary to derive the formula. The All Minors Matrix Tree Theorem generalizes the Matrix Tree Theorem Theorem 1.2.1. We state a simplified version of the theorem without providing a proof.

Theorem D.5.1. Given a graph $G = (V, E)$ and a subset U of nodes in G , let $W = V \setminus U$. We define $L_{W,W}$ as the submatrix of the Laplacian matrix of G , which includes the rows and columns indexed by the nodes in W . In this context, the determinant of $L_{W,W}$, denoted as $\det(L_{W,W})$, provides a count of the number of spanning forests of G that consist of $|U|$ disjoint trees, with the nodes in U being disconnected across these trees.

Proof: See Chaiken [30]. □

Now we are ready to present the main result of this subsection. Consider a full tree topology T_{BCST} . The number of topologies for the CST problem that can be derived from T_{BCST} is given by

$$\det L_{SP_s, SP_s}, \tag{D.22}$$

where L_{SP_s, SP_s} represents the submatrix of the Laplacian matrix L of T_{BCST} . This submatrix is formed by selecting the rows and columns associated with the SP. By virtue of Theorem D.5.1, equation (D.22) counts the number of spanning forests of the T_{BCST} which disconnect the terminal nodes. To demonstrate that this count of forests coincides with the number of topologies that can be derived from the full tree topology T_{BCST} , we will establish a bijection.

Indeed, if we have a forest that disconnects all the terminals, each SP within the forest must belong to a component with a single terminal. In this scenario, we can unambiguously collapse each SP to its corresponding terminal. Once we have collapsed the SP, we still need to reconnect the terminals between them to form a valid CST topology. Now, notice that in the original full tree topology T_{BCST} , each terminal is uniquely adjacent to a SP. We can connect the terminals between them based on the collapse process of the SP. Specifically, a terminal v_t is connected to another terminal u_t if the neighboring SP of v_t in the original T_{BCST} has been collapsed to u_t . Similarly, we can reverse these steps to map a CST topology back to a unique forest that disconnects the terminals. Figure D.5 illustrates the individual steps of this bijection using two examples. We have proven the following theorem

Theorem D.5.2. Let T_{BCST} be a full tree topology with N terminals. Consider the Laplacian matrix L of T_{BCST} . The number of CST topologies that can be derived from T_{BCST} is equal to the determinant of L_{SP_s, SP_s} , which is the submatrix of the Laplacian obtained by selecting the rows and columns indexed by the SPs. Hence, the number of CST topologies derived from T_{BCST} can be calculated as $\det L_{SP_s, SP_s}$.

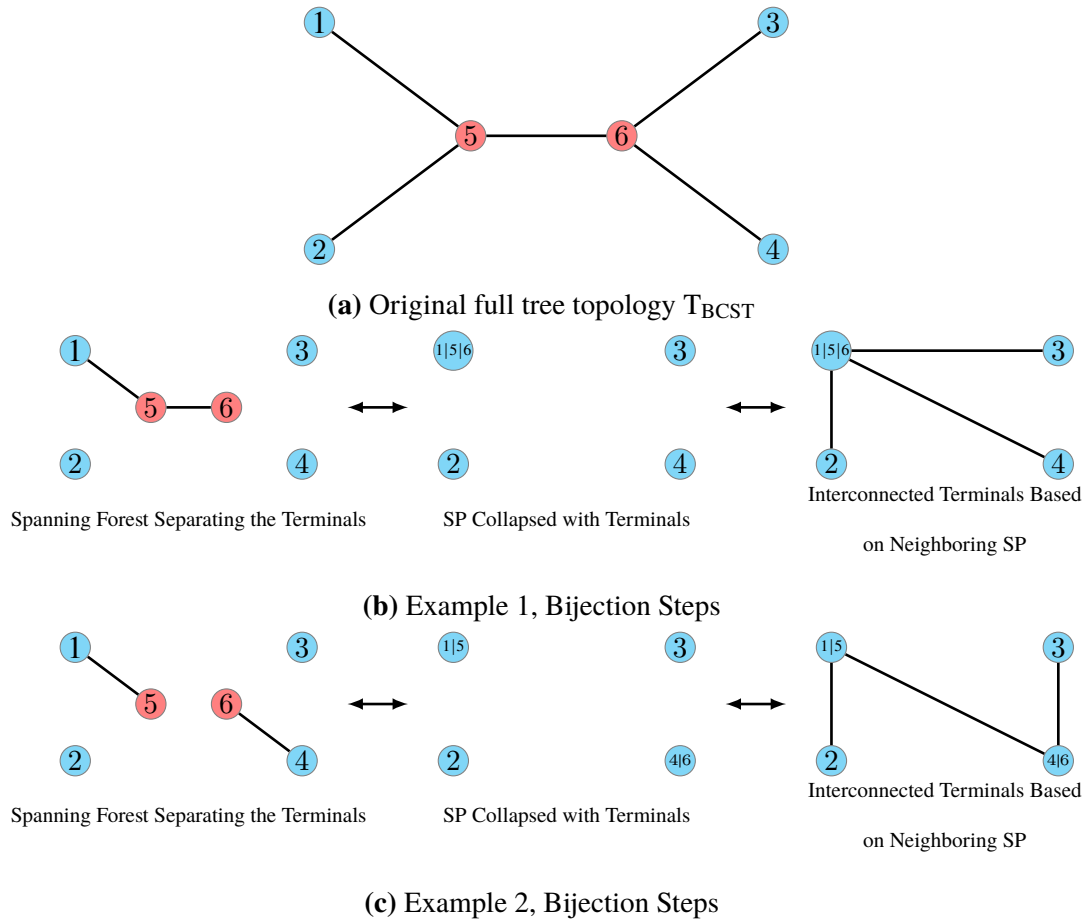


Figure D.5. Bijections Between Derivable CST Topologies from a Full Tree Topology and Their Terminal-Separating Forests. Figures D.5b) and D.5c) illustrate two examples of the relationship between a spanning forest and a derived CST topology of a full tree topology depicted in Figure D.5a.

D.6 Branching Angles at the Steiner Points in the BCST Problem

In this section, we formulate the branching angles in terms of the centralities of the edges for a given topology of the BCST problem. As stated in Section 5.5.1, it is sufficient to study the geometric optimization of 3 nodes connected by a single SP, with minimization objective given by

$$C(b) = \zeta_0 \|b - a_0\| + \zeta_1 \|b - a_1\| + \zeta_2 \|b - a_2\|. \tag{D.23}$$

Recall that node b represents the Steiner point whose coordinates need to be optimized, nodes $\{a_i\}$ are the terminals with fixed positions and $\zeta_i := m_{ba_i}(1 - m_{ba_i})$ are the centralities of the edges (b, a_i) (see Figure D.6).

We will reproduce the arguments exposed for the BOT problem in Bernot et al. [18] and Lippmann et al. [108] to determine the angles θ_1 and θ_2 . We will differentiate two cases:

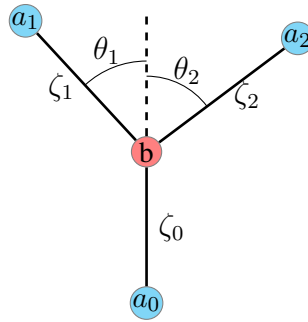


Figure D.6. Branching Angles at Steiner Point. The symbols ζ_i represent the normalized centralities of the edges, that is $\zeta_i := m_{ba_i}(1 - m_{ba_i})$.

when the SP does not coincide with any other terminal node; and when b collapses with one of the terminals.

D.6.1 Steiner Point b Does Not Collapse with a Terminal

In this case, the function is differentiable with respect to b , and therefore we just need to see where the gradient of equation (D.23) is equal to zero. The formula for the gradient is as follows

$$\nabla_b C(b) = \zeta_0^\alpha n_0 + \zeta_1^\alpha n_1 + \zeta_2^\alpha n_2, \quad (\text{D.24})$$

where $n_i = \frac{b-a_i}{\|b-a_i\|}$. By applying the dot product to $\nabla_b C(b)$ with each n_i and setting it equal to zero, we derive the following equalities:

$$\begin{aligned} \langle \nabla_b C(b), n_0 \rangle = 0 &\rightarrow \zeta_0^\alpha + \underbrace{\zeta_1^\alpha \langle n_1, n_0 \rangle}_{-\cos(\theta_1)} + \underbrace{\zeta_2^\alpha \langle n_2, n_0 \rangle}_{-\cos(\theta_2)} = 0 \\ \langle \nabla_b C(b), n_1 \rangle = 0 &\rightarrow \zeta_0^\alpha \underbrace{\langle n_0, n_1 \rangle}_{-\cos(\theta_1)} + \zeta_1^\alpha + \zeta_2^\alpha \langle n_2, n_1 \rangle = 0 \\ \langle \nabla_b C(b), n_2 \rangle = 0 &\rightarrow \zeta_0^\alpha \underbrace{\langle n_0, n_2 \rangle}_{-\cos(\theta_2)} + \zeta_1^\alpha \langle n_1, n_2 \rangle + \zeta_2^\alpha = 0 \end{aligned}$$

Solving the linear system we obtain that the angles satisfy

$$\begin{aligned} \cos(\theta_1) &= \frac{\zeta_0^{2\alpha} + \zeta_1^{2\alpha} - \zeta_2^{2\alpha}}{2\zeta_0^\alpha \cdot \zeta_1^\alpha}, \\ \cos(\theta_2) &= \frac{\zeta_0^{2\alpha} + \zeta_2^{2\alpha} - \zeta_1^{2\alpha}}{2\zeta_0^\alpha \cdot \zeta_2^\alpha}, \\ \cos(\theta_1 + \theta_2) &= \frac{\zeta_0^{2\alpha} - \zeta_1^{2\alpha} - \zeta_2^{2\alpha}}{2\zeta_1^\alpha \cdot \zeta_2^\alpha}. \end{aligned} \quad (\text{D.25})$$

D.6.2 Steiner Point b Collapses with a Terminal

In this case, in order to determine the optimality angles, we will use the subdifferential argument applied in Lippmann et al. [108]. W.l.o.g. we will assume that b collapses with

terminal a_0 .

The subdifferential of a convex function $h : \mathbb{R}^n \rightarrow \mathbb{R}$ at x is defined as the following set of vectors

$$\partial g(x) := \{v : h(z) \geq h(x) + \langle v, z - x \rangle, \forall z \in \mathbb{R}^n\}.$$

In other words, $\partial g(x)$ comprises all vectors v such that the line passing through $h(x)$ in the direction of v lies below the function h at all points. Each of these vectors is called a subgradient of h at x . When a function is differentiable at x , the subdifferential only contains the gradient of the function at x .

Fermat rule states that a convex function attains its minimum at x if and only if $0 \in \partial g(x)$. Furthermore, the subdifferential of two convex functions is equal to the union of the pairwise sums of their subgradients. In other words, for $g(x) = g_1(x) + g_2(x)$ then

$$\partial g(x) = \{v_1 + v_2 : v_1 \in \partial g_1(x), v_2 \in \partial g_2(x)\}. \quad (\text{D.26})$$

We can apply Fermat's rule to determine when the minimum is attained at $b = a_0$. For the function $g(x) = w \cdot \|x - a\|$, the subdifferential is given by

$$\partial g(x) = \begin{cases} \{v : \|v\| \leq w\}, & \text{if } x = a \\ \left\{ w \frac{x-a}{\|x-a\|} \right\}, & \text{otherwise} \end{cases}.$$

Thus, applying equation (D.26), the subdifferential of $C(b)$ at $b = a_0$ is given by

$$\partial C(a_0) = \left\{ v + \zeta_1^\alpha \frac{b - a_1}{\|b - a_1\|} + \zeta_2^\alpha \frac{b - a_2}{\|b - a_2\|} : \|v\| \leq \zeta_0^\alpha \right\}.$$

In order for b to be optimal at a_0 , zero has to belong to $\partial C(a_0)$, which is true if and only if

$$\begin{aligned} & \left\| \zeta_1^\alpha \frac{b - a_1}{\|b - a_1\|} + \zeta_2^\alpha \frac{b - a_2}{\|b - a_2\|} \right\| \leq \zeta_0^\alpha \\ \iff & \left\| \frac{b - a_1}{\|b - a_1\|} + \frac{b - a_2}{\|b - a_2\|} \right\|^2 = \zeta_1^{2\alpha} + \zeta_2^{2\alpha} + 2\zeta_1^\alpha \zeta_2^\alpha \cos(\gamma) \leq \zeta_0^{2\alpha} \end{aligned} \quad (\text{D.27})$$

where γ is the angle of the terminal triangle at a_0 , that is $\gamma := \angle a_1 a_0 a_2$. Isolating γ , we get

$$\gamma \geq \arccos \left(\frac{\zeta_0^{2\alpha} - \zeta_1^{2\alpha} - \zeta_2^{2\alpha}}{2\zeta_1^\alpha \cdot \zeta_2^\alpha} \right) = \theta_1 + \theta_2. \quad (\text{D.28})$$

Thus b will collapse to a_0 if the angle $\angle a_1 a_0 a_2$ is greater than the optimal angle given by (D.25). In such cases, the resulting branching is referred to as a V -branching.

Remark D.6.1. It is worth noting that the reasoning presented in this section remains independent of the weighting factors, which, in our case, were set equal to the normalized edge centralities powered to α . As a result, this finding holds true for any weights and can be used to determine an arbitrary weighted geometric median of three points. Furthermore, we emphasize that the position of the SP, b , depends exclusively on the angles and weighting factors and not on the distances between the terminal nodes.

D.7 Infeasibility of Degree-4 Steiner Points in the Plane for $\alpha = 1$

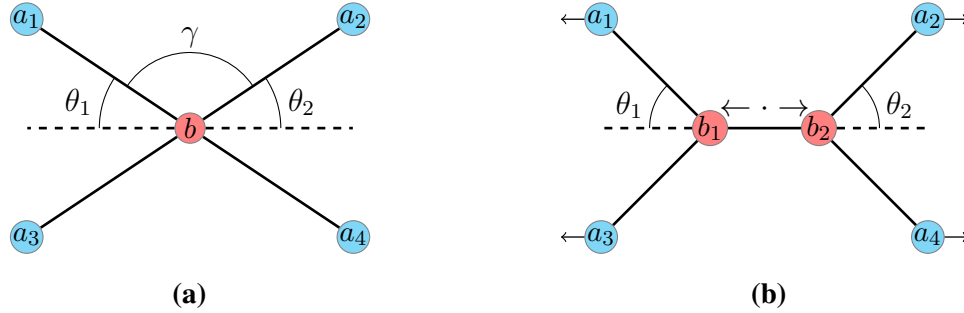


Figure D.7. Splitting Collapsed SP While Preserving Optimal Angles. D.7a) illustrates the collapsed solution of a 4-terminal configuration. D.7b) demonstrates that it is possible to move jointly the terminal points $\{a_1, a_3\}$ in a specific but opposite direction to the one of the terminals $\{a_2, a_4\}$, resulting in the splitting of the collapsed SP b into two distinct SPs, b_1 and b_2 . Remarkably, this split can be executed while preserving the angles θ_1 and θ_2 . Importantly, these angles must correspond to the optimal angles given by (D.25).

As stated in Section 5.5.2, the optimal position of the SPs is continuously dependent on the terminal positions and solely relies on the branching angles, as shown in Section 5.5.1. Consequently, assuming that there exists a configuration such that the SPs collapse, it is possible to find terminal positions that lead to an unstable collapse of the SPs. Here, instability refers to a configuration where an infinitesimal translation of the terminals results in the splitting of the SPs. This scenario is depicted in Figure D.7. In such cases, the angles realized by the terminals and the SPs will reach the upper bounds specified by (D.28). Therefore, the angles depicted in Figure D.7a fulfill the condition

$$\gamma = \pi - \theta_1 - \theta_2, \quad (\text{D.29})$$

where the angles satisfy

$$\cos(\gamma) = \frac{F(m_{a_1b} + m_{a_2b})^{2\alpha} - F(m_{a_1b})^{2\alpha} - F(m_{a_2b})^{2\alpha}}{2F(m_{a_1b})^\alpha F(m_{a_2b})^\alpha} \quad (\text{D.30})$$

$$\cos(\theta_1) = \frac{F(m_{a_3b} + m_{a_1b})^{2\alpha} + F(m_{a_1b})^{2\alpha} - F(m_{a_3b})^{2\alpha}}{2F(m_{a_3b} + m_{a_1b})^\alpha F(m_{a_1b})^\alpha} \quad (\text{D.31})$$

$$\cos(\theta_2) = \frac{F(m_{a_2b} + m_{a_4b})^{2\alpha} + F(m_{a_2b})^{2\alpha} - F(m_{a_4b})^{2\alpha}}{2F(m_{a_2b} + m_{a_4b})^\alpha F(m_{a_2b})^\alpha} \quad (\text{D.32})$$

We can manipulate (D.29) in the following way

$$\begin{aligned}
\gamma = \pi - \theta_1 - \theta_2 &\iff \cos(\gamma - \pi) = \cos(-\theta_1 - \theta_2) \\
&\iff -\cos(\gamma) = \cos(\theta_1 + \theta_2) \\
&\iff \underbrace{-\cos(\gamma)}_* = \cos(\theta_1) \cos(\theta_2) - \sqrt{(1 - \cos(\theta_1)^2)(1 - \cos(\theta_2)^2)}
\end{aligned} \tag{D.33}$$

where in (*) we have used the fact that

$$\cos(x+y) = \cos(x) \cos(y) - \sin(x) \sin(y) = \cos(x) \cos(y) - \sqrt{(1 - \cos(x)^2)(1 - \cos(y)^2)}.$$

If we square both sides of D.33 and equate to 0 we obtain.

$$(\cos(\gamma) + \cos(\theta_1) \cos(\theta_2))^2 - (1 - \cos(\theta_1)^2)(1 - \cos(\theta_2)^2) = 0. \tag{D.34}$$

Equation (D.34) depends on the variables m_{a_1b} , m_{a_2b} , m_{a_3b} , and α^1 . Equation (D.34) is generally too complex to be solved analytically. However, with the help of the Mathematica software [83], we have determined that for $\alpha = 1$, the equality does not hold within the constraints of the problem, namely

$$\sum_{i=1}^4 m_{a_i,b} = 1 \quad \text{and} \quad 0 < m_{a_i,b} < 1, \quad \forall i.$$

To simplify the notation, let's denote m_{a_ib} as m_i . For $\alpha = 1$, when we expand equation (D.34), we find that the numerator of the formula becomes a fourth-degree polynomial with respect to m_1 . The four roots, $\{m_1^{(j)}\}$ of the polynomial are

$$\begin{aligned}
m_1^{(1)} &= \frac{1}{2} \left(1 - m_2 - m_3 - \sqrt{(-1 + m_2 + m_3)^2 - \frac{4}{3} \left(-2m_2 - 2m_3 + 3m_2m_3 - \sqrt{m_2^2 - m_2m_3 + m_3^2} \right)} \right) \\
m_1^{(2)} &= \frac{1}{2} \left(1 - m_2 - m_3 + \sqrt{(-1 + m_2 + m_3)^2 - \frac{4}{3} \left(-2m_2 - 2m_3 + 3m_2m_3 - 2\sqrt{m_2^2 - m_2m_3 + m_3^2} \right)} \right) \\
m_1^{(3)} &= \frac{1}{2} \left(1 - m_2 - m_3 - \sqrt{(-1 + m_2 + m_3)^2 - \frac{4}{3} \left(-2m_2 - 2m_3 + 3m_2m_3 + 2\sqrt{m_2^2 - m_2m_3 + m_3^2} \right)} \right) \\
m_1^{(4)} &= \frac{1}{2} \left(1 - m_2 - m_3 + \sqrt{(-1 + m_2 + m_3)^2 - \frac{4}{3} \left(-2m_2 - 2m_3 + 3m_2m_3 + 2\sqrt{m_2^2 - m_2m_3 + m_3^2} \right)} \right)
\end{aligned} \tag{D.35}$$

where we have highlighted the difference between the roots. We will show that

$$1 < m_1^{(4)} + m_2 + m_3 < m_1^{(2)} + m_2 + m_3 \quad \text{and} \quad m_1^{(1)} < m_1^{(3)} < 0,$$

which implies that the problem constraints are not satisfied, and therefore SPs of degree 4 are not possible.

¹Since $\sum_{i=1}^4 m_{a_ib} = 1$, m_{a_4b} can be expressed as $1 - m_{a_1b} - m_{a_3b} - m_{a_2b}$.

Claim 1: $1 < m_1^{(4)} + m_2 + m_3 \leq m_1^{(2)} + m_2 + m_3$. From (D.35) it is clear that $m_1^{(4)} \leq m_1^{(2)}$. Thus, it is enough to prove the inequality for $m_1^{(4)}$:

$$\begin{aligned}
m_1^{(4)} + m_2 + m_3 &= \frac{1}{2}(1 + m_2 + m_3) + \frac{1}{2}\sqrt{(-1 + m_2 + m_3)^2 - \frac{4}{3}(-2m_2 - 2m_3 + 3m_2m_3 - \sqrt{m_2^2 - m_2m_3 + m_3^2})} \\
&= \frac{1}{2}(1 + m_2 + m_3) + \frac{1}{2}\sqrt{1 + \frac{2}{3}(m_2 + m_3) + (m_2 - m_3)^2 - \frac{8}{3}\sqrt{\frac{m_2^2 - m_2m_3 + m_3^2}{<(m_2+m_3)^2}}} \\
&> \frac{1}{2}(1 + m_2 + m_3) + \frac{1}{2}\sqrt{1 + \frac{2}{3}(m_2 + m_3) + (m_2 - m_3)^2 - \frac{8}{3}(m_2 + m_3)} \\
&= \frac{1}{2}(1 + m_2 + m_3) + \frac{1}{2}\sqrt{1 + (m_2 - m_3)^2 - \frac{1}{2}(m_2 + m_3)} \\
&> \frac{1}{2}\left(\underbrace{1 + m_2 + m_3}_{>2} + \sqrt{1 - \frac{1}{2}(m_2 + m_3)}\right) > 1
\end{aligned}$$

For the last inequality, we have used the fact $0 < m_2 + m_3 < 1$, that the function $g(x) = 1 + x + \sqrt{1 - x/2}$ is increasing in $[0, 1]$ and that $g(0) = 2$.

Claim 2: $m_1^{(1)} \leq m_1^{(3)} < 0$. From (D.35) it is clear that $m_1^{(1)} \leq m_1^{(3)}$. Thus, it is enough to prove the inequality for $m_1^{(3)}$.

$$\begin{aligned}
m_1^{(3)} &< 0 \\
&\iff \frac{1}{2}\left(1 - m_2 - m_3 - \sqrt{(-1 + m_2 + m_3)^2 - \frac{4}{3}(-2m_2 - 2m_3 + 3m_2m_3 + 2\sqrt{m_2^2 - m_2m_3 + m_3^2})}\right) < 0 \\
&\iff (-2m_2 - 2m_3 + 3m_2m_3 + 2\sqrt{m_2^2 - m_2m_3 + m_3^2}) < 0 \tag{D.36}
\end{aligned}$$

Thus, we need to focus on inequality D.36. We will differentiate various cases:

- **If $2/3 > m_3 \geq m_2$:**

$$\begin{aligned}
&\left(-2m_2 - 2m_3 + 3m_2m_3 + 2\sqrt{m_2^2 - m_2m_3 + m_3^2}\right) \\
&= m_2 \underbrace{(2 - 3m_3)}_{>0 \iff 2/3 > m_3} + 2 \underbrace{\left(m_3 - \sqrt{m_2^2 - m_2m_3 + m_3^2}\right)}_{\geq 0 \iff m_3 \geq m_2} > 0
\end{aligned}$$

For the second term, we have used the fact that

$$\begin{aligned}
m_3 \geq \sqrt{m_2^2 - m_2m_3 + m_3^2} &\iff m_3^2 \geq m_2^2 - m_2m_3 + m_3^2 \iff m_2m_3 \geq m_2^2 \\
&\iff m_3 \geq m_2
\end{aligned}$$

- **If $2/3 \geq m_3 \geq m_2$:** Analogous to previous case due to the symmetry of m_2 and m_3 in (D.36)
- **If $\max(m_2, m_3) \geq 2/3$:** W.l.o.g we can assume $m_2 \geq 2/3$ and $m_3 < 1/3$ due to the symmetry between m_2 and m_3 . For this case, we will find the roots with respect to m_2 of inequality (D.36) and see that the constraints on m_2 do not hold. Indeed,

$$\begin{aligned}
-2m_2 - 2m_3 + 3m_2m_3 + 2\sqrt{m_2^2 - m_2m_3 + m_3^2} &= 0 \implies \\
(-2m_2 - 2m_3 + 3m_2m_3)^2 &= 4(m_2^2 - m_2m_3 + m_3^2)
\end{aligned}$$

The roots $\{m_2^{(j)}\}$ of the polynomial are

$$\begin{aligned}
 m_2^{(1)} &= \frac{2m_3}{-5 + 6m_3 + \sqrt{3}\sqrt{7 - 12m_3 + 6m_3^2}} && \& \quad m_3 \neq \frac{2 - \sqrt{2}}{3}, \\
 m_2^{(2)} &= \frac{2m_3}{-5 + 6m_3 - \sqrt{3}\sqrt{7 - 12m_3 + 6m_3^2}} && \& \quad m_3 \neq \frac{2 - \sqrt{2}}{3}, \\
 m_2^{(3)} &= \frac{2(-3 + 2\sqrt{2})}{3(-2 + 3\sqrt{2})} && \& \quad m_3 = \frac{2 - \sqrt{2}}{3}.
 \end{aligned} \tag{D.37}$$

The denominator of the root $m_2^{(1)}$ has negative sign for $0 < m_3 < 1$, which leads to $m_2^{(1)} < 0$, contradicting the initial constraints. Trivially, $m_2^{(3)}$ is also negative.

The denominator of the root $m_2^{(2)}$ is negative for $0 < m_3 < \frac{2-\sqrt{2}}{3}$, resulting in $m_2^{(2)} < 0$. When $\frac{2-\sqrt{2}}{3} < m_3 < 1/3$, the denominator becomes positive but remains lower than $2m_3$, thus $m_2^{(2)} > 1$, which is also a contradiction.

We have ruled out all possible cases, thus we have proven the next theorem.

Theorem D.7.1. Let $\alpha = 1$. Given a set of terminals which lie in the plane, then the SPs of the optimal solution of the BCST problem will not contain SPs of degree 4 unless these collapse with a terminal.

D.8 Iteratively Reweighted Least Square for the Geometric Optimization of the Steiner Points

In this section we review briefly the iteratively reweighted least square (IRLS) algorithm proposed in [157]. This algorithm was initially developed for the geometric optimization of Steiner points (SPs) and later adapted in [108] for the branched optimal transport (BOT) problem. We will show that the same algorithm can be adapted for the BCST problem, since the algorithm is agnostic to the weighting factors multiplying the distances involved in the BOT and BCST objectives, as defined in equations (D.3) and (5.2), respectively.

Consider the following minimization problem for a fixed tree topology

$$\min_{X_B} C(X) = \min_{X_B} \sum_{(i,j) \in E} w_{ij} \|x_i - x_j\| \tag{D.38}$$

where w_{ij} are arbitrary weights, E is the set of edges of the tree, $X_B = \{x_{N+1}, \dots, x_{2N-2}\}$ are the coordinates of the SPs, which need to be optimized, and $X = \{x_1, \dots, x_{2N-2}\}$ is the set of all coordinates (terminals and SPs). Starting from arbitrary SPs coordinates, denoted

as $X^{(0)}$, the algorithm iteratively solves the following linear system of equations.

$$x_i^{(k+1)} = \frac{\sum_{j:(i,j) \in E} w_{ij} \frac{x_j^{(k+1)}}{\|x_i^{(k)} - x_j^{(k)}\|}}{\sum_{j:(i,j) \in E} \frac{w_{ij}}{\|x_i^{(k)} - x_j^{(k)}\|}}, \quad \forall N+1 \leq i \leq 2N-2. \quad (\text{D.39})$$

Note that only the coordinates corresponding to the SPs are updated. The coordinates of the terminals are kept fixed and set equal to their original coordinates.

We will show that in each iteration the cost of the objective function decreases, i.e. $C(X^{(k+1)}) < C(X^{(k)})$. As shown in [157], this implies that the $\lim_{k \rightarrow \infty} X^{(k)} = \arg \min C(X)$.

The algorithm can be considered an IRLS approach because it reinterprets the cost function as a quadratic function. Indeed, $C(X)$ can be rewritten as

$$C(X) = \sum_{(i,j) \in E} w_{ij} \|x_i - x_j\| = \sum_{(i,j) \in E} \frac{w_{ij}}{\underbrace{\|x_i - x_j\|}_{W_{ij}(X)}} \|x_i - x_j\|^2 = \sum_{(i,j) \in E} W_{ij}(X) \|x_i - x_j\|^2$$

In concrete, the solution of the linear system (D.39) minimizes the following quadratic function

$$Q^{(k)}(X) = \sum_{(i,j) \in E} W_{ij}(X^{(k)}) \|x_i - x_j\|^2.$$

That is $Q^{(k)}(X) \geq Q^{(k)}(X^{(k+1)}) \forall X$. Moreover, note that $C(X^{(k)}) = Q^{(k)}(X^{(k)})$. Now we can show that the cost C decreases at each iteration:

$$\begin{aligned} C(X^{(k)}) &= Q^{(k)}(X^{(k)}) \geq Q^{(k)}(X^{(k+1)}) \\ &= \sum_{(i,j) \in E} w_{ij} \frac{\left(\left| x_i^{(k)} - x_j^{(k)} \right| + \left| x_i^{(k+1)} - x_j^{(k+1)} \right| - \left| x_i^{(k)} - x_j^{(k)} \right| \right)^2}{\left| x_i^{(k)} - x_j^{(k)} \right|} \\ &= C(X^{(k)}) + 2(C(X^{(k+1)}) - C(X^{(k)})) \\ &\quad + \sum_{(i,j) \in E} w_{ij} \frac{\left(\left| x_i^{(k+1)} - x_j^{(k+1)} \right| - \left| x_i^{(k)} - x_j^{(k)} \right| \right)^2}{\left| x_i^{(k)} - x_j^{(k)} \right|} \end{aligned} \quad (\text{D.40})$$

$$\iff$$

$$\begin{aligned} C(X^{(k+1)}) &\leq C(X^{(k)}) - \underbrace{\sum_{(i,j) \in E} \frac{w_{ij}}{2} \frac{\left(\left| x_i^{(k+1)} - x_j^{(k+1)} \right| - \left| x_i^{(k)} - x_j^{(k)} \right| \right)^2}{\left| x_i^{(k)} - x_j^{(k)} \right|}}_{\geq 0} \\ &\leq C(X^{(k)}) \end{aligned}$$

D.9 Complexity mSTreg Heuristic

We will delve into the complexity of the two steps of our heuristic.

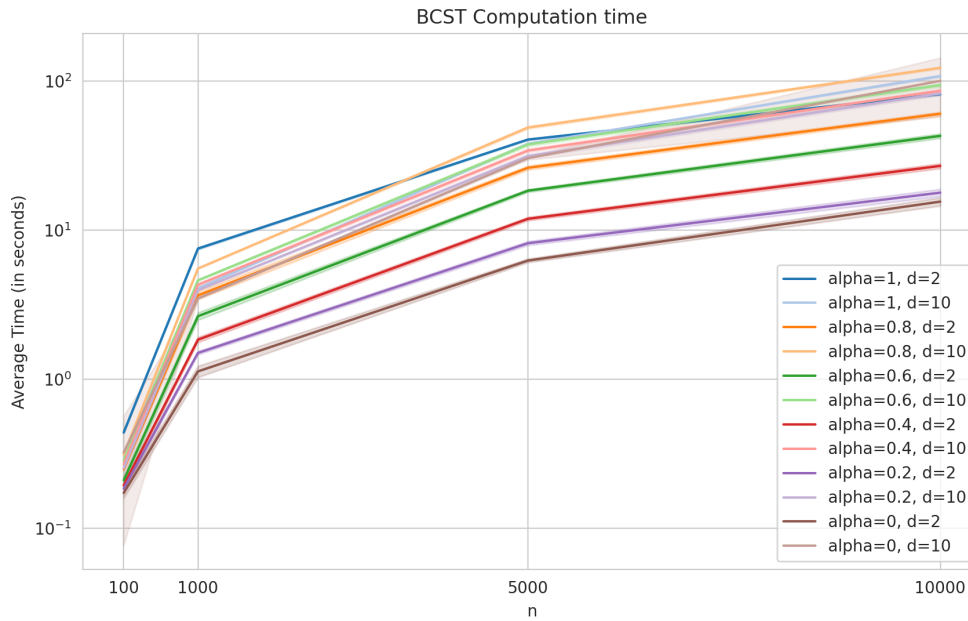


Figure D.8. BCST time complexity. Computation time of 20 iterations of the mSTreg heuristic averaged over 5 distinct instances with different numbers of terminals n , data dimensionality d , and α values.

- **Complexity geometric optimization:** We approximate the optimal SPs coordinates using the IRLS approach presented in Section D.8. Each iteration of the IRLS requires $\mathcal{O}(nd)$ operations, where n is the number of terminals and d is the data dimensionality. Within each iteration, d linear systems are solved. These can be solved in linear time and in parallel. The number of iterations needed for the IRLS to converge is not known a priori, however, Lippmann et al. [108] suggest that this number could scale on average like $\mathcal{O}(\log(n))$. Consequently, each geometric optimization step takes $\mathcal{O}(\log(n)nd)$.
- **Topology optimization step:** In the topology optimization step, we compute the minimum spanning tree (mST) over the terminals and SPs. Given a graph $G = (V, E)$, Kruskal's algorithm takes $\mathcal{O}(|E| \log |V|)$ operations to compute the mST. In a complete graph, this becomes $\mathcal{O}(n^2 \log(n))$. However, in some situations, we may expedite the mST computation by computing the mST over a k -nearest neighbor (kNN) graph. Approximating a kNN graph with k -d trees can have a complexity of $\mathcal{O}(dn \log(n)^2)$. In this case, the number of edges in the graph would be $|E| \approx kn$. Hence, the overall mST complexity would be $\mathcal{O}(dn \log(n)^2 + kn \log(n)) \approx \mathcal{O}(dn \log(n)^2)$.

Therefore, the heuristic's per-iteration complexity is approximately $\mathcal{O}(dn \log(n) + n^2 \log(n))$ or $\mathcal{O}(dn \log(n)^2)$ if the mST is computed over a kNN graph. Throughout our experiments, a limit of 20 iterations was set, though practical convergence often demands fewer. In addition, we gauged the computational time of the heuristic by averaging its performance over 20

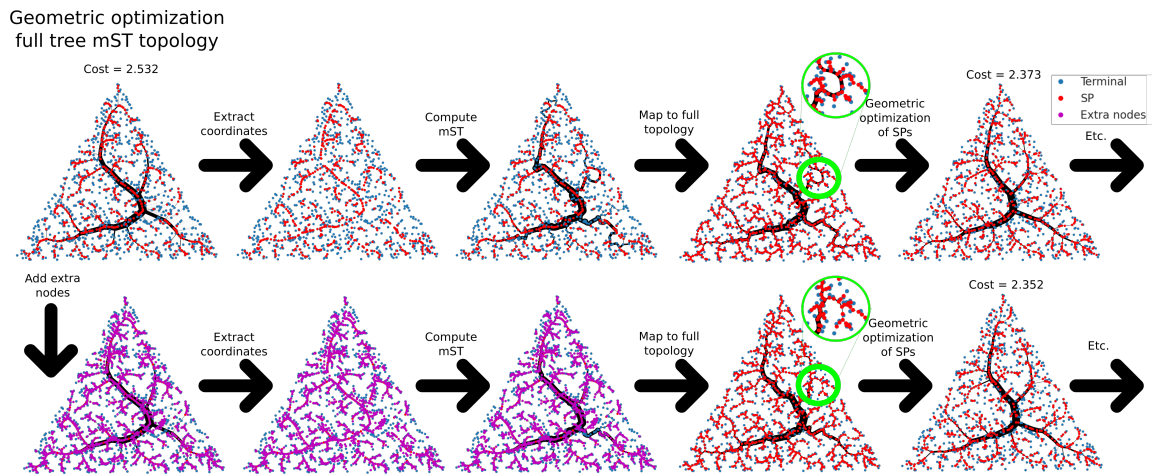


Figure D.9. mSTreg Heuristic with Additional Sampled Points. Effect of adding extra points per edge (visualized in violet) in the mST computation step of the mSTreg heuristic. Top left: BCST solution obtained once the mST has been mapped to a full tree topology and its Steiner point coordinates have been optimized. Top row: Next steps of the mSTreg heuristic without adding any extra point. Bottom row: Next steps of the mSTreg heuristic once an extra point has been added at the middle of each edge (shown in violet). The addition of extra points may allow the mST to more reliably follow the edges of the geometry-optimized tree from previous step. We zoom in to highlight an improvement in the topology resulting from the addition of these extra points. In this particular case, the cost obtained with the inclusion of the extra nodes is lower than the cost without them.

iterations across 5 distinct instances, varying n , d , and α . Data was generated by sampling n points from a d -dimensional unit cube. The performance times are presented in Figure D.8. The heuristic was executed on an Intel Xeon Gold 6254 CPU @ 3.10GHz.

D.10 Effect of Additional Intermediate Points in the mSTreg Heuristic

In Section 5.6.2 we have described the mSTreg heuristic as a solution approach for the BCST problem. The algorithm can be summarized in two steps: 1) Optimization of the SPs coordinates given a fixed topology; 2) topology update by computing the mST over the terminals and SPs. The motivation for the topology update is that the optimal positions of the SPs may suggest a more desirable topology, since they may be biased to move closer to other nodes than the ones to which they are connected. Thus, we hope that the new topology, defined by the mST over the SPs and the terminals, interconnects such nodes.

However, there are instances where the SPs may not be sufficiently close to each other,

causing the mST to fail in recovering the desired connections. the addition of intermediate nodes along the edges may address this problem, allowing the mST to more reliably follow the edges of the geometry-optimized tree from the previous step. An illustrative example highlighting the benefits of this approach can be seen in Figure D.9. In general, we have seen that adding between 1 and 3 nodes per edge can often yield improvements. However, the impact on the main backbone is minimal. In Algorithm 2, the number of intermediate points that are added along an edge is regulated by the `sampling_frequency` variable.

D.11 Strategies to Transform a Full Tree Topology into a CST Topology

When using the mSTreg heuristic described in Section 5.6.2 for solving the CST problem without branching points, we need to map from a full tree topology to a CST topology. As shown in Section D.5.2, this process is ambiguous and there may be an exponential number of derivable topologies with respect to the number of terminals. Hence to brute force the one which minimizes the CST cost is out of reach.

In this section, we describe some heuristic rules to transform a full tree topology into a CST topology. In order to transform a full tree topology into a CST topology, we collapse iteratively one SP at a time with one of its neighbors until there are no more SPs to collapse. The first factor to take into account is in which order the SPs are collapsed. We consider two strategies: 1) collapse the SP that is closest to a terminal (“Ordclosestterminal”) or 2) collapse the SP with the closest neighbor, i.e. the one that minimizes the distance to one of each neighbors independently of if it is a terminal or a SP (“Ordclosest”). In practice we did not see any big difference, though “Ordclosest” tends to be slightly better.

The second factor to take into account is to which neighbor should an SP collapse. We again compare two different heuristics: 1) collapsing the SP to the neighbor that minimally increases the CST cost (“greedy”); 2) collapsing the SP to the closest neighbor in terms of distance. We found empirically that the greedy approach yields significantly superior results.

Lastly, we conducted tests on updating the position of the collapsed SP. When a SP, denoted as b_1 , is collapsed with a neighbor b_2 , then the other neighbors of b_1 become neighbors of b_2 . We observed that updating the position of b_2 to the weighted geometric median of its neighbors (including those inherited from b_1) yielded improved results compared to not updating the coordinates of b_2 . The coordinates of b_2 were only updated when b_2 was an SP. If b_2 happened to be a terminal, its position was kept fixed. To denote if a strategy updated the position or not we will use the expression “update” and “no_update” respectively.

To evaluate the effectiveness of the strategies, we conducted a series of experiments by

sampling 200 problem instances for each N in the set $\{5, 6, 7, 8, 9\}$, where N represents the number of terminals. For each instance, we applied the mSTreg heuristic with different α values and utilized the aforementioned strategies to transform a full tree topology into a CST topology. Figure D.10 shows the mean ranking positions obtained by the different strategies, once all feasible solutions have been sorted. The results confirm the observations that we already pointed out. For all of our experiments we used the combination that produced the best results i.e. “update”+“greedy”+“Ordclosest”.

D.12 Further Details on the Brute Force Experiment

In this section, we analyze the behavior of the mSTreg heuristic with respect to α . To investigate this, we utilize the experiment described in Section 5.7, which compared the cost of the output tree generated by our heuristic with the optimal solution for different numbers of terminal nodes, denoted as N , while specifically examining the influence of α .

For each $N \in \{5, 6, 7, 8, 9\}$, we sample 200 problem instances. We computed the optimal CST and BCST topologies of all problems via brute-force and with the mSTreg heuristic² for all $\alpha \in \{0, 0.1, \dots, 0.9, 1\}$. Figures D.11 and D.12 show the relative error and how the heuristic solution ranks, when the costs of all topologies are sorted. When solving the BCST, though there is not a clear trend, we can observe that for higher N the heuristic tends to perform worse for higher α values, since on average the heuristic’s solution ranking is higher. When solving the CST problem this pattern can be more clearly seen.

D.13 Selection of α

In this section, we present our practical insights into determining the optimal value for α . It is crucial to emphasize that the choice of α is task-dependent and influenced by the desired level of structure preservation. Nevertheless, we share the observations derived from our empirical experiences.

For simpler examples, as the ones illustrated in Section D.1, we have consistently found that α values within the range of $[0.7, 1]$ yield high stability while preserving the primary data structure. In general, an increase in α correlates with a heightened inclination toward a star-shaped tree, in line with the limit case discussed in Appendix 5.2.1. This tendency becomes more pronounced in higher dimensions, where even relatively small α values ($\alpha \lesssim 1$) lead to an almost star-shaped tree, compromising data structure preservation. This arises from the

²As described in D.11, we can use different strategies to transform a full tree topology into a CST topology, when solving the CST problem with the mSTreg heuristic. We used the one that updates the position of SPs and collapses to the closest neighbor.

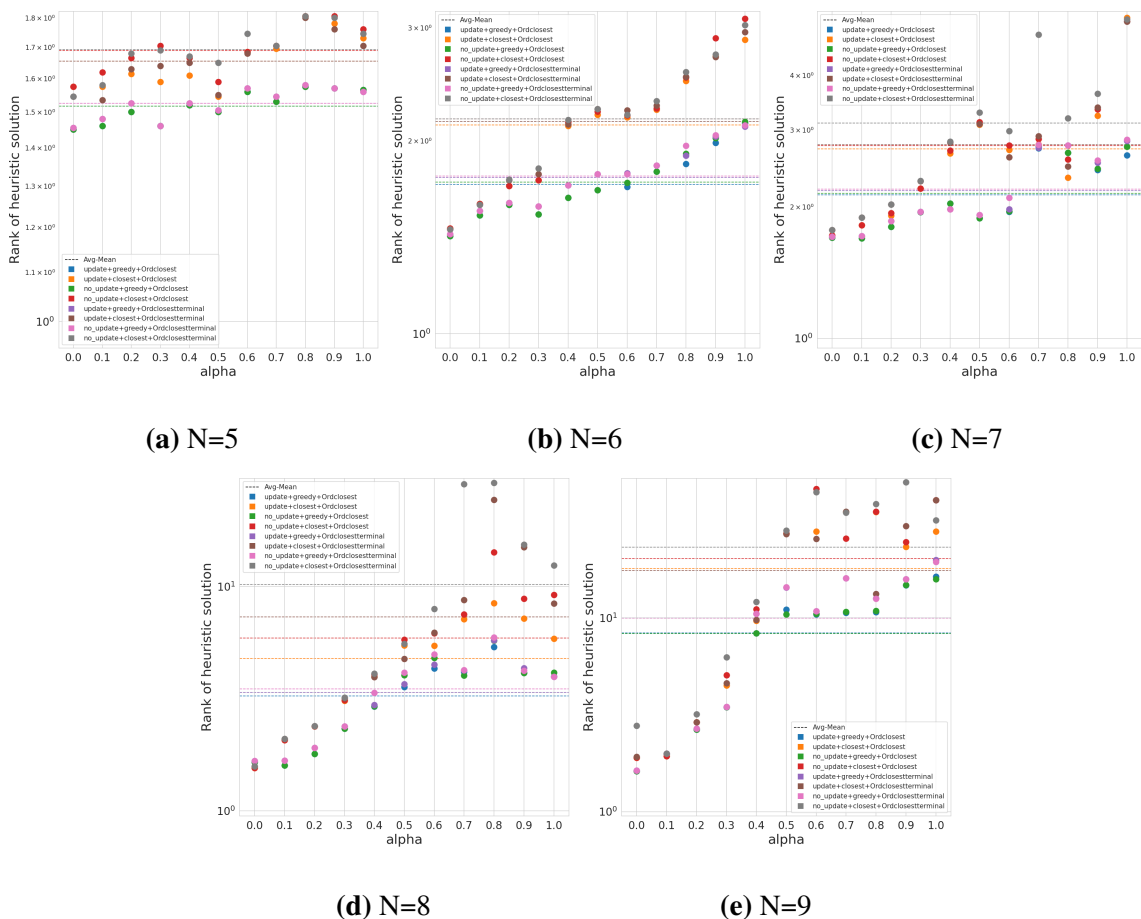


Figure D.10. Performance SPs Collapse Strategies. Comparison of different collapse strategies to transform a full tree topology into a CST topology. See Section D.11 for a short description of the strategies. We plot the average sorted position of the heuristic for different number of terminals, N and for different α values. We observe that the strategy combinations including the “greedy” collapse approach have significantly better results. The combinations which update the position of the collapsed SP (“update”) perform slightly better than the ones that do not (“no_update”). Analogously, the strategies that order the SPs based on the closeness to the neighbors (“Ordclosest”) is slightly better than “Ordclosestterminal”.

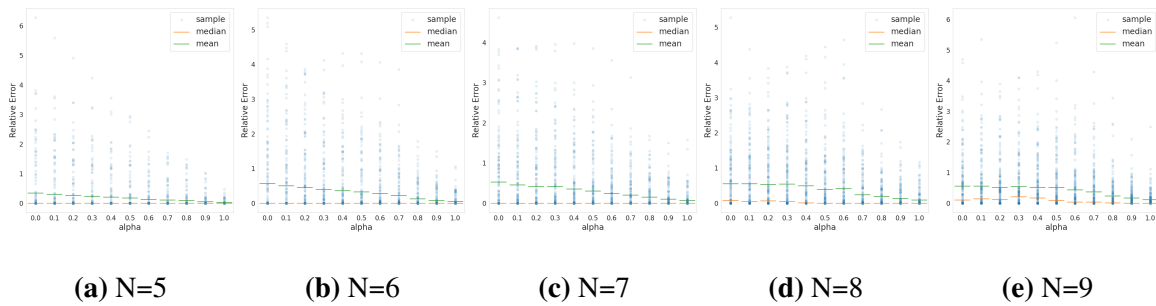
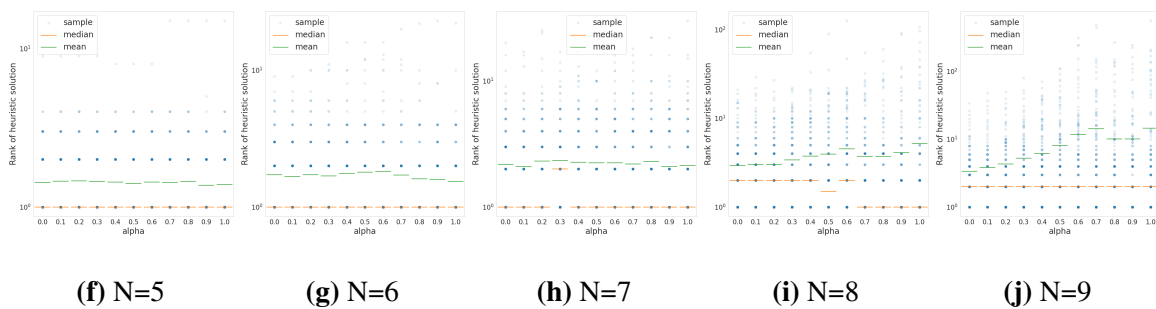
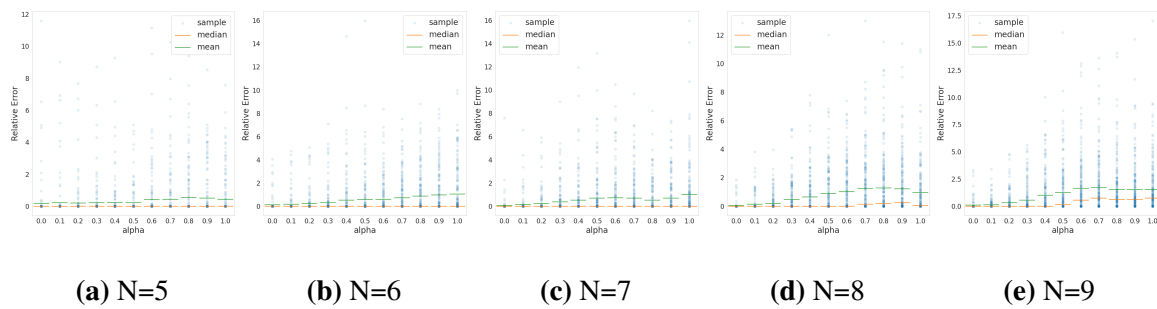
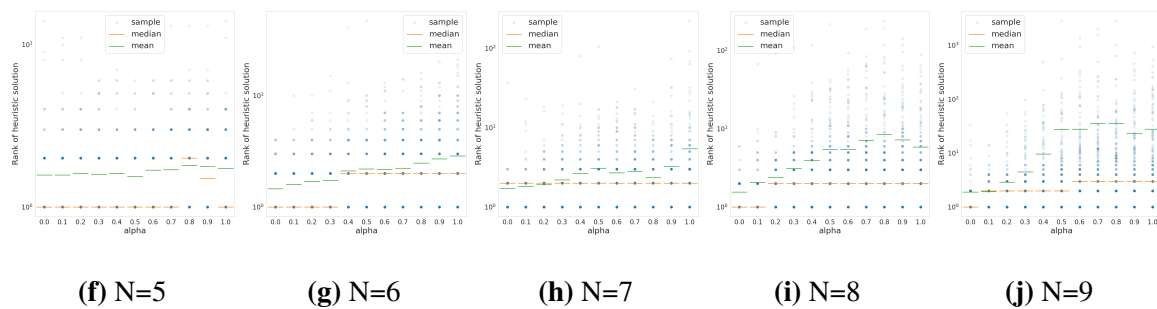
BCST relative error for different number of terminals N BCST rank of heuristic for different number of terminals N

Figure D.11. BCST Bruteforce Benchmark with Respect to α . Relative cost errors between the mSTreg heuristic and BCST optimal solutions; and sorted position of the heuristic tree for different number of terminals, N . For each N we uniformly sampled 200 different terminal configurations and we solved them for all $\alpha \in \{0.0, 0.1, \dots, 1.0\}$. Most runs ended up close to the global optimum. There is no clear pattern with respect to the performance of the heuristic with respect to the value of α , though for higher number of terminals, it seems that the rank of our solution gets to be worse on average.



CST relative error for different number of terminals N



CST rank of heuristic for different number of terminals N

Figure D.12. CST Bruteforce Benchmark with Respect to α . Relative cost errors between the mSTreg heuristic and optimal CST solutions; and sorted position of the heuristic tree for different number of terminals, N . For each N we uniformly sampled 200 different terminal configurations and we solved them for all $\alpha \in \{0.0, 0.1, \dots, 1.0\}$. Most runs ended up close to the global optimum. There is no clear pattern with respect to the performance of the heuristic with respect to the value of α , though for higher number of terminals, it seems that the rank of our solution gets to be worse on average.

curse of dimensionality, where the majority of point pairs in a high-dimensional Euclidean space become nearly equidistant. Consequently, the strong triangle inequality, derived in Theorem 5.2.2, will hold for most triplets of points due to the approximate equality of $c_{ku} \approx c_{kv}$ in equation (5.3). Figures 5.5i-5.5l) exemplify this effect on the Paul dataset.

We refrained from further exploring the case of $\alpha > 1$ due to both practical and theoretical observations pointing towards an excessively star-shaped tree. Indeed, as demonstrated in 5.2.1, when the number of terminals, N , is sufficiently large and $\alpha > 1$, the optimal solution results in a star-graph. The transition to a star-tree occurs quite early in this scenario. For instance, with moderate values of N and $\alpha \gtrsim 1$ (e.g., $N = 1000$, $\alpha = 1.13$), an optimal star-tree emerges. Refer to Figure 5.2 to observe how the value of α at which the optimal solution becomes a star-tree approaches 1 as N increases.

In summary, our empirical experience suggests that intermediate α values (around 0.5) effectively preserve the data structure while maintaining relative stability. This choice holds true for the applications highlighted in the thesis. We hope that by sharing our experiences, practitioners can better select an appropriate α for their respective applications.

D.14 Implementation Details

In this section, we explain some implementation details of the mSTreg heuristic and also the parameters used for the different experiments.

In each iteration of the mSTreg algorithm, it is necessary to compute the mST. Since we are working with a complete graph, the computational complexity of the mST computation is $O(N^2)$. To reduce this cost, we compute the mST over a k -nearest neighbor (kNN) graph, where we set the value of k to $\log(N)$. While the resulting mST over the kNN graph may not always match the optimal mST, in practice, they often yield similar results. It is worth noting that the introduction of additional nodes, as described in Section D.10, may provide more significant benefits when using the mST computed over a kNN graph.

In Section D.11 we have described different approaches to transform a full tree topology into a CST tree topology. The strategy used to collapse the SP nodes upon transforming a full tree topology into a CST tree was the one that updates the collapsed SP to the weighted geometric median, collapses greedily the SPs and determines the SP to be collapsed as the one with minimum distance to one of its neighbors (“update+greedy+Ordclosest”).

In all experiments, we set the `sampling_frequency` variable of Algorithm 2 equal to 3, and we set the maximum number of iterations of the mSTreg heuristic equal to 20.

Appendix E

BCST-Based 3D Plant Skeletonization

E.1 Additional Skeletonization Results

This section shows additional qualitative results of the experiments realized in Section 6.3. In Figure E.1, we showcase the skeleton obtained for the apple tree presented in the main part of the thesis across further density levels, specifically with 5000 and 10000 sample points, in addition to those that had already been presented. Additionally, E.2 and E.3 display the skeletons at various density levels for a different tree, namely a pine tree. Similarly, Figures E.4 and E.5 show the results obtained in the missing and noisy experiment settings described in Section 6.3 for the same pine tree.

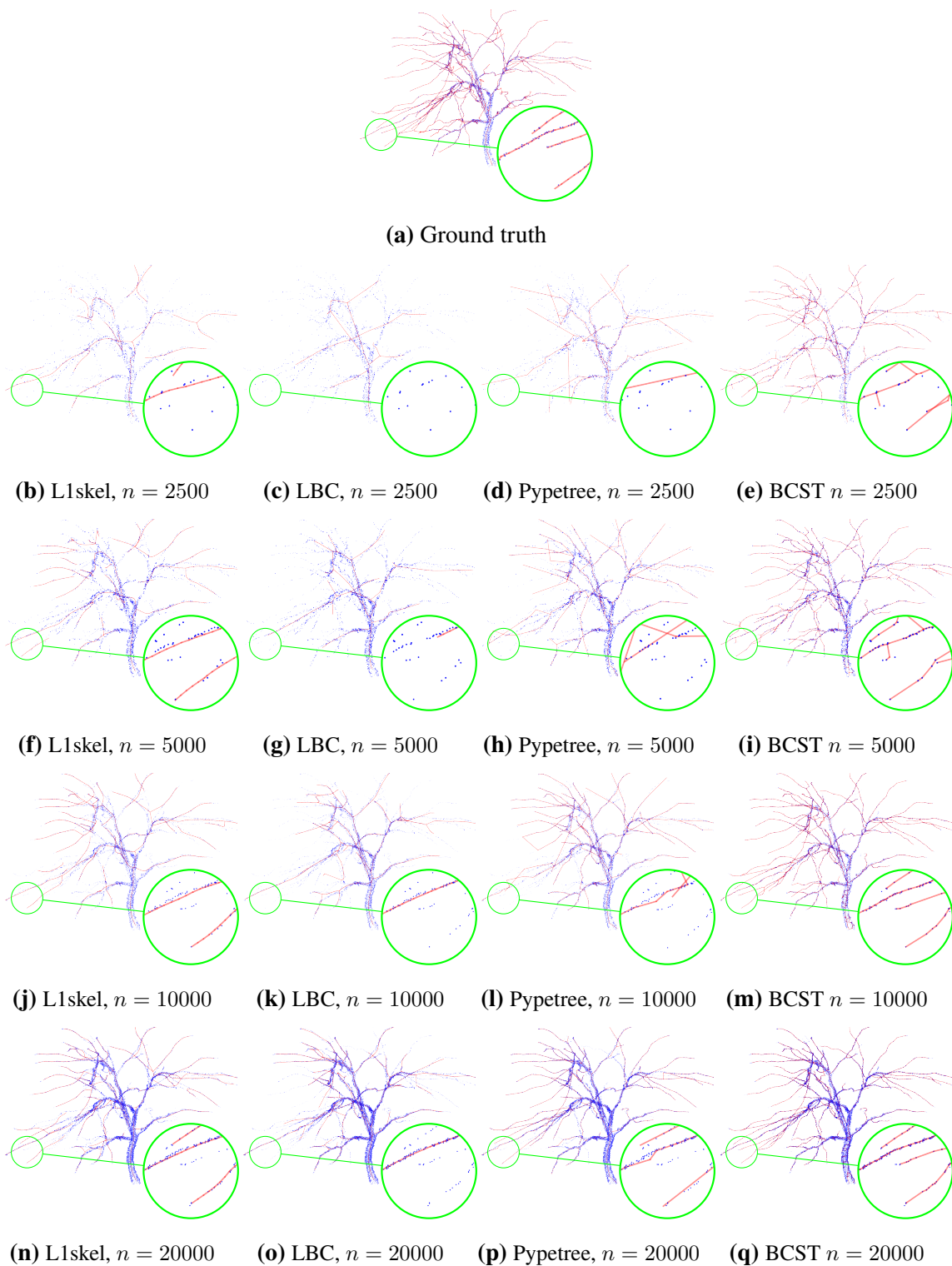


Figure E.1. Performance Comparison Skeletons at Different Density Levels (Apple Tree). Comparative analysis of skeletonization methods applied to different density levels of an apple tree instance. Each column corresponds to a distinct technique, while each row represents varying density levels. Notably, BCST outperforms other methods by accurately modeling a majority of branches, particularly evident at lower densities (E.1e) where the other methods miss most of the branches (E.1b-E.1d).

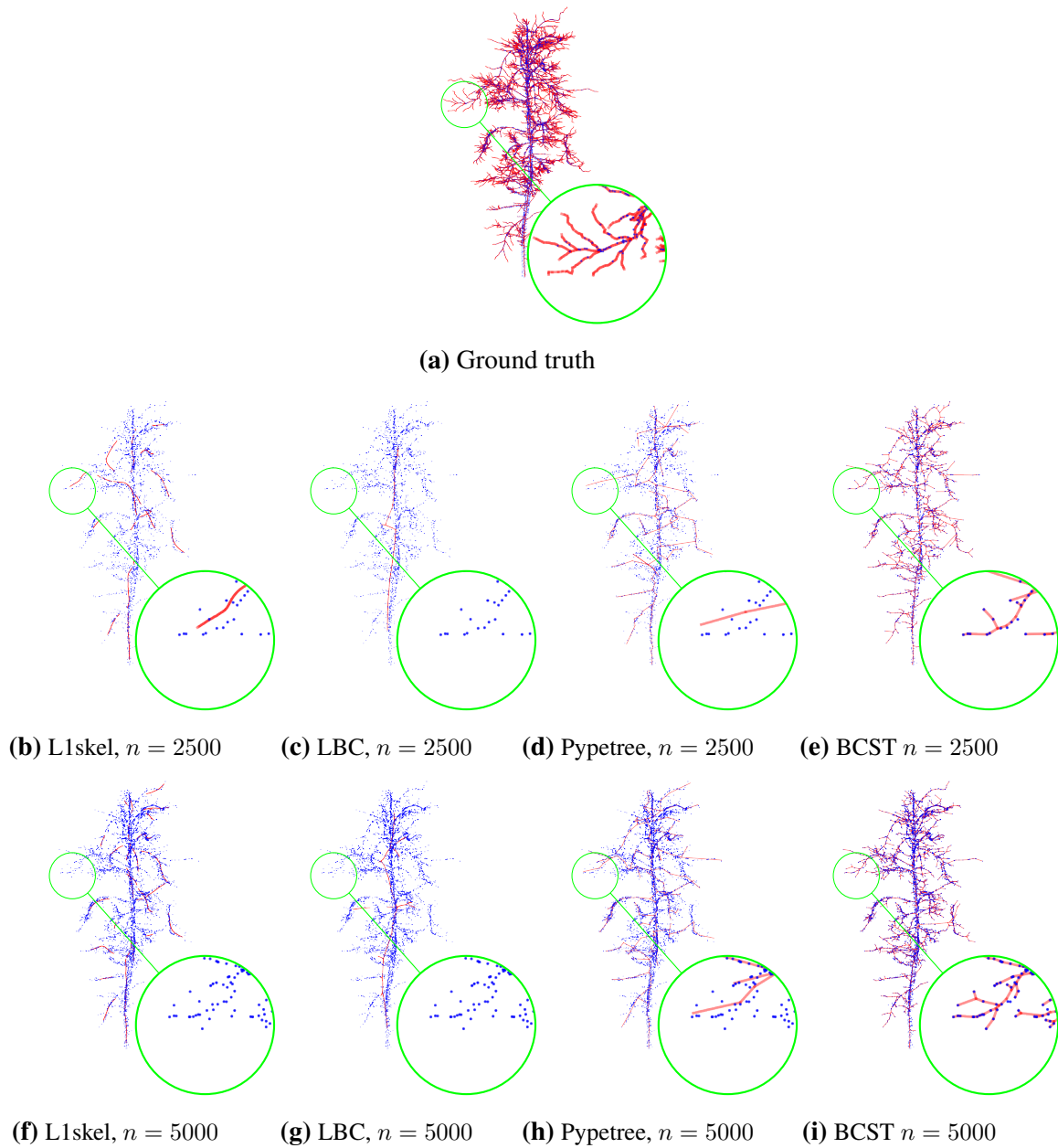


Figure E.2. See caption Figure E.3

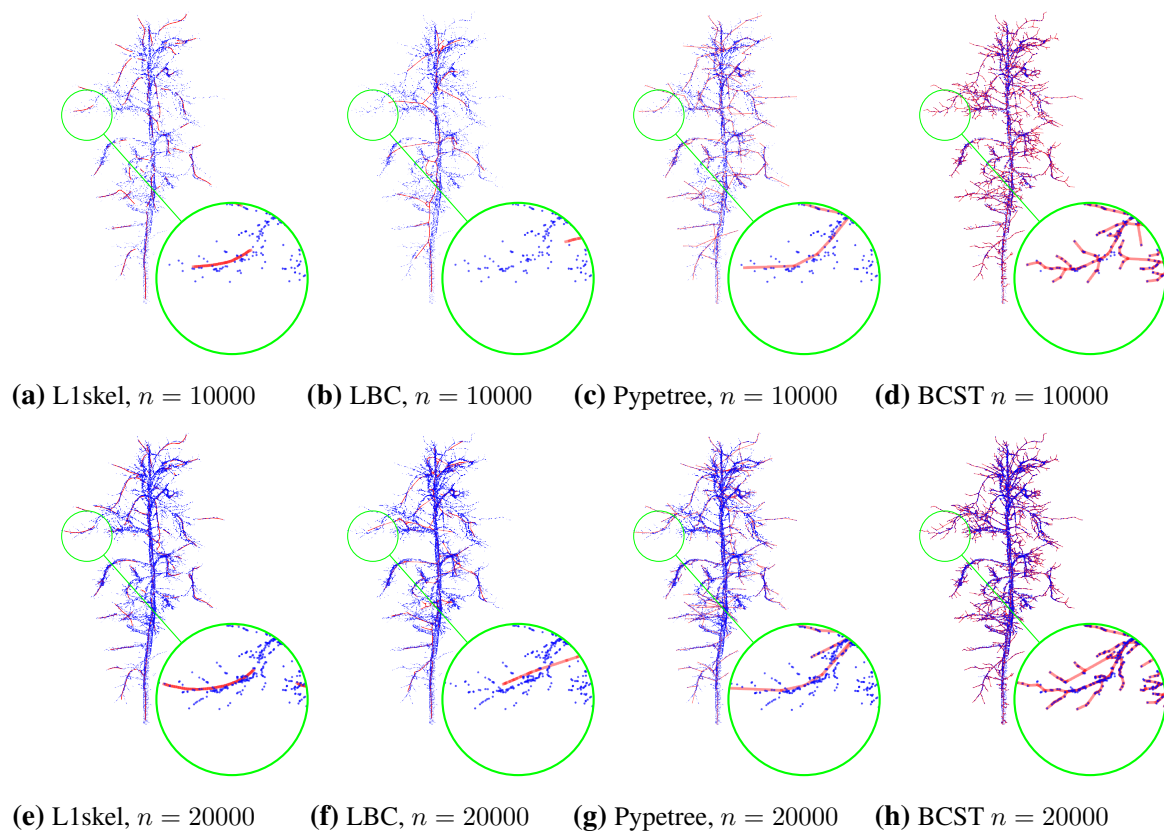


Figure E.3. Performance Comparison Skeletons at Different Density Levels (Pine Tree).

Comparative analysis of skeletonization methods applied to different density levels of pine tree instance. Each column corresponds to a distinct technique, while each row represents varying density levels. Notably, BCST outperforms other methods by accurately modeling a majority of branches, particularly evident at lower densities E.2f) where the other methods miss most of the branches (E.2c-E.2e).

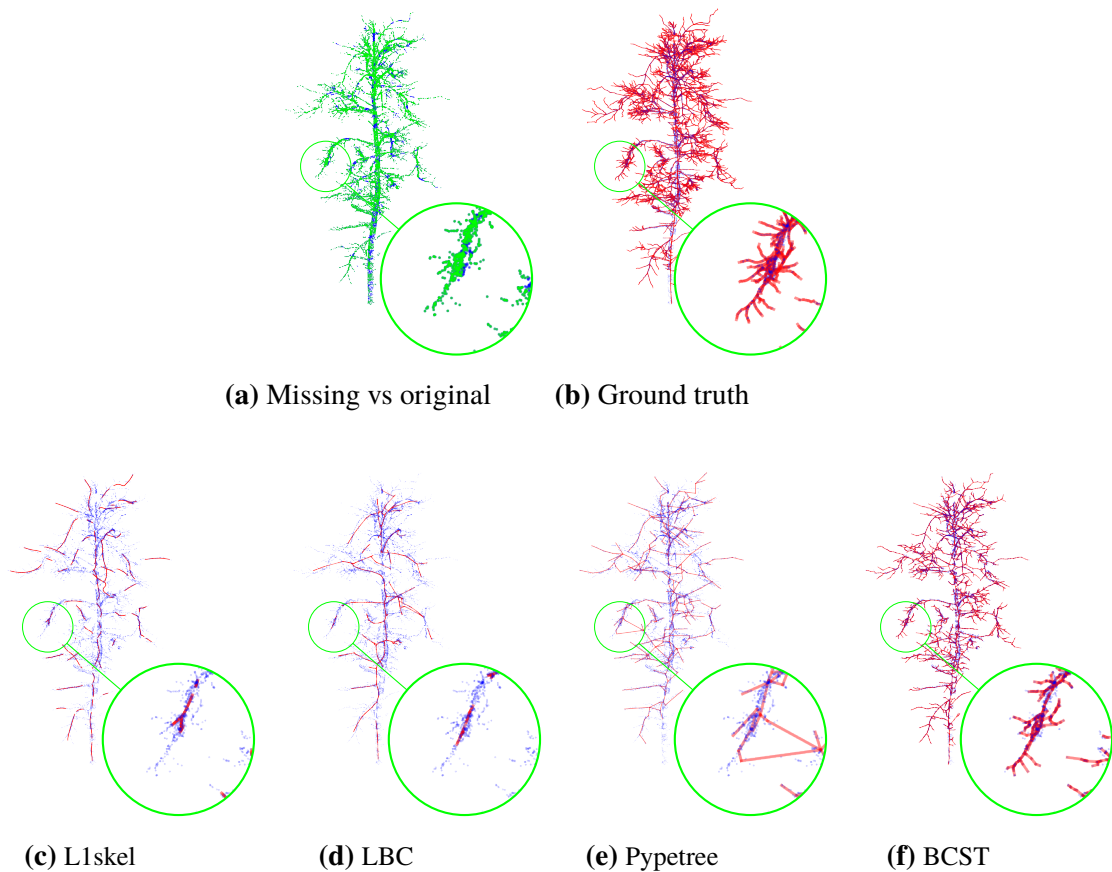


Figure E.4. Performance Comparison Skeletons with Missing Data (Pine Tree). Comparative analysis of skeletonization methods applied to an apple tree instance with 20% missing data. The green points in E.4a represent the input data points, while the blue points correspond to the 20% of data points intentionally removed from the original sample with $n = 20000$. BCST demonstrates superior performance compared to other methods, accurately modeling the majority of branches even in the presence of 20% missing data.

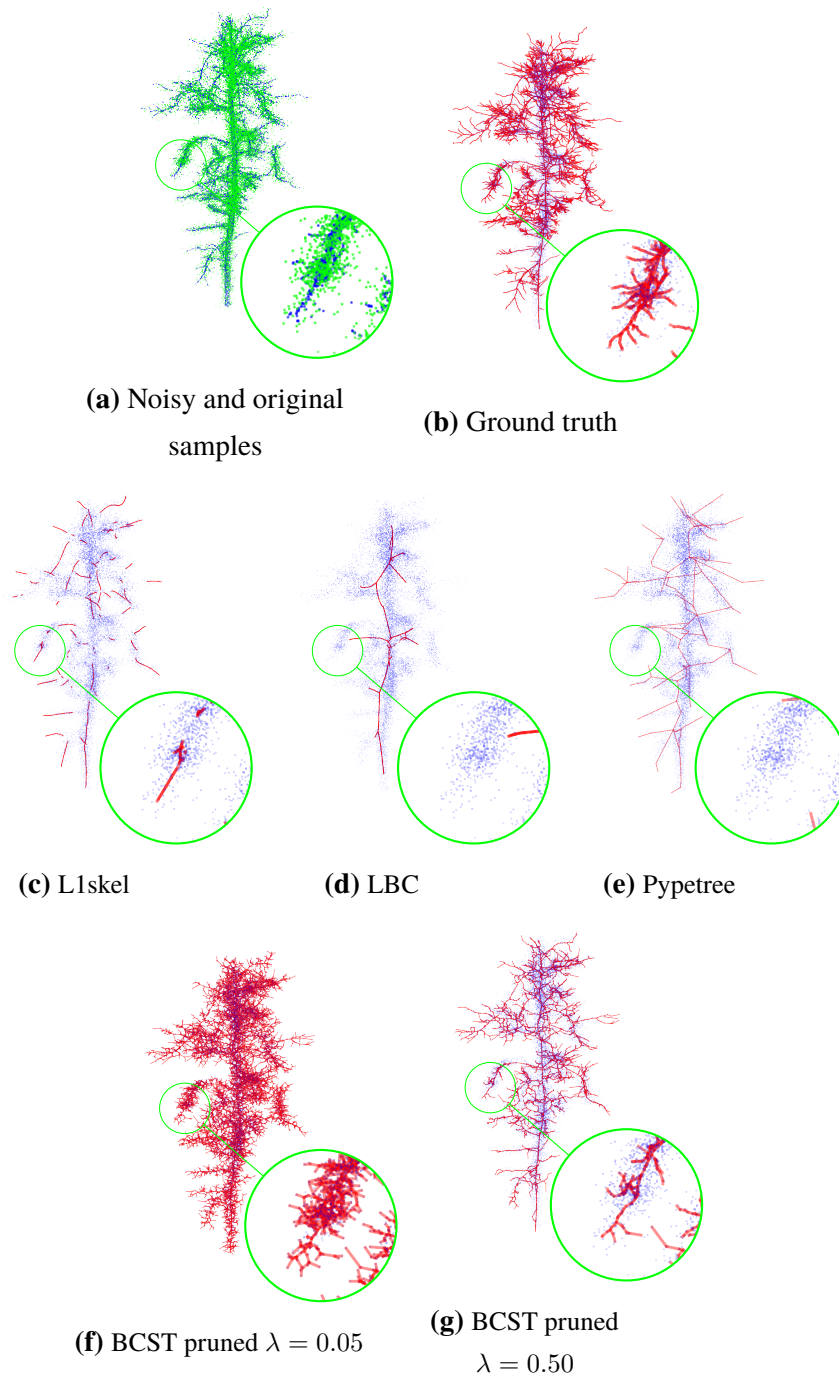


Figure E.5. Performance Comparison Skeletons with Noisy Data (Pine Tree). Comparative analysis of skeletonization methods on a pine tree instance affected by Gaussian noise. The green points in E.5a) represent the input data points post-application of Gaussian noise, while the blue points depict the original data points without any noise. Remarkably, BCST stands out by accurately preserving all branches, a feature not shared by other methods. However, the default weighting factor ($\lambda = 0.05$) used for pruning keeps several spurious branches. This issue can be addressed by increasing λ , as demonstrated in E.5g).

Bibliography

- [1] L. Addario-Berry, N. Broutin, and B. Reed. Critical random graphs and the structure of a minimum spanning tree. *Random Structures & Algorithms*, 2009. 35: 323–347.
- [2] S. Ahn, B. Chen, T. Wang, and L. Song. Spanning tree-based graph generation for molecules. *International Conference on Learning Representations*. 2022 .
- [3] M. Ai, Y. Yao, Q. Hu, Y. Wang, and W. Wang. An automatic tree skeleton extraction approach based on multi-view slicing using terrestrial lidar scans data. *Remote Sensing*, 2020. 12: 3824.
- [4] A. P. Akella. A brief survey on representation learning based graph dimensionality reduction techniques. *arXiv preprint arXiv:2211.05594*, 2022.
- [5] F. Alam, S. Joty, and M. Imran. Graph based semi-supervised learning with convolution neural networks to classify crisis related tweets. *Proceedings of the international AAAI conference on web and social media*, vol. 12. 2018 .
- [6] M. Alamgir. Shortest path distance in random k-nearest neighbor graphs. *Proceedings of the 29th International Conference on Machine Learning*, 2012. 8.
- [7] M. Aliannejadi, M. Kiaeeha, S. Khadivi, and S. S. Ghidary. Graph-based semi-supervised conditional random fields for spoken language understanding using unaligned data. *Proceedings of the Australasian Language Technology Association Workshop 2014*. 2014 98–103.
- [8] L. Anton-Sanchez, C. Bielza, and P. Larrañaga. Network design through forests with degree- and role-constrained minimum spanning trees. *Journal of Heuristics*, 2017. 23: 31–51.
- [9] H. K. Azad and A. Deepak. Query expansion techniques for information retrieval: a survey. *Information Processing & Management*, 2019. 56: 1698–1735.
- [10] X. Bai, L. Ye, Z. Liu, and B. Liu. Promask: Probability mask representation for skeleton detection. *Neural Networks*, 2023. 162: 11–20.

- [11] J. S. Baras and G. Theodorakopoulos. Path Problems in Networks. *Synthesis Lectures on Communication Networks*, 2010. 3: 1–77.
- [12] J. E. Beasley. A heuristic for Euclidean and rectilinear Steiner problems. *European Journal of Operational Research*, 1992. 58: 284–292. Publisher: Elsevier.
- [13] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 2001. 14.
- [14] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 2003. 15: 1373–1396.
- [15] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 2006. 7: 2399–2434.
- [16] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 1958. 16: 87–90.
- [17] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun. Discrete elastic rods. *ACM SIGGRAPH 2008 papers*, 1–12. 2008.
- [18] M. Bernot, V. Caselles, and J.-M. Morel. *Optimal transportation networks: models and theory*. Springer, 2008.
- [19] D. A. G. Betancur, A. Fabijańska, L. Florez-Valencia, A. M. Pinzón, E. E. D. Serrano, J.-C. Richard, M. Orkisz, and M. H. Hoyos. Airway segmentation, skeletonization, and tree matching to improve registration of 3d ct images with large opacities in the lungs. *International Conference on Computer Vision and Graphics*. 2016 .
- [20] S. Bezrukov, F. Kaderali, and W. Poguntke. On central spanning trees of a graph. *Combinatorics and Computer Science: 8th Franco-Japanese and 4th Franco-Chinese Conference Brest, France, July 3–5, 1995 Selected Papers*. Springer, 1996 53–57.
- [21] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008. 2008: P10008.
- [22] M. Borcan. Tf-idf explained and python sklearn implementation. <https://towardsdatascience.com/tf-idf-explained-and-python-sklearn-implementation-b020c5e83275>, 2020. [Online; accessed 04-May-2021].

- [23] U. Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 2008. 30: 136–145.
- [24] J. Calder, B. Cook, M. Thorpe, and D. Slepcev. Poisson learning: Graph based semi-supervised learning at very low label rates. *International Conference on Machine Learning*. PMLR, 2020 1306–1316.
- [25] R. Campos and M. Ricardo. A fast algorithm for computing minimum routing cost spanning trees. *Computer Networks*, 2008. 52: 3229–3247.
- [26] J. Cao, A. Tagliasacchi, M. Olson, H. Zhang, and Z. Su. Point cloud skeletons via laplacian based contraction. *2010 Shape Modeling International Conference*. IEEE, 2010 187–197.
- [27] J. L. Cárdenas-Donoso, C. J. Ogayar, F. R. Feito, and J. M. Jurado. Modeling of the 3d tree skeleton using real-world data: a survey. *IEEE Transactions on Visualization and Computer Graphics*, 2022.
- [28] B. A. Carre. An Algebra for Network Routing Problems. *IMA Journal of Applied Mathematics*, 1971. 7: 273–294.
- [29] A. Cayley. A theorem on trees. *Quart. J. Math.*, 1878. 23: 376–378.
- [30] S. Chaiken. A combinatorial proof of the all minors matrix tree theorem. *SIAM Journal on Algebraic Discrete Methods*, 1982. 3: 319–329.
- [31] A. Challa, S. Danda, B. S. D. Sagar, and L. Najman. Watersheds for semi-supervised classification. *IEEE Signal Processing Letters*, 2019. 26: 720–724.
- [32] A. K. Chandra, P. Raghavan, W. L. Ruzzo, R. Smolensky, and P. Tiwari. The electrical resistance of a graph captures its commute and cover times. *computational complexity*, 1996. 6: 312–340. Publisher: Springer.
- [33] A. Chaudhury and C. Godin. Skeletonization of plant point cloud data in stochastic imization framework. *bioRxiv*, 2020. 2020–02.
- [34] P. CHEBOTAREV. The matrix-forest theorem and measuring relations in small social groups. *Autom. Remote Control*, 1997. 58: 1505–1514.
- [35] P. Chebotarev. A class of graph-geodetic distances generalizing the shortest-path and the resistance distances. *Discrete Applied Mathematics*, 2011. 159: 295–302.
- [36] P. Chebotarev. The walk distances in graphs. *Discrete Applied Mathematics*, 2012. 160: 1484–1500.

- [37] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *International conference on machine learning*. PMLR, 2020 1597–1607.
- [38] L. Chizat, S. Zhang, M. Heitz, and G. Schiebinger. Trajectory inference via mean-field langevin in path space. S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds., *Advances in Neural Information Processing Systems*, vol. 35. Curran Associates, Inc., 2022 16731–16742.
- [39] Y. Chong, Y. Ding, Q. Yan, and S. Pan. Graph-based semi-supervised learning: A review. *Neurocomputing*, 2020. 408: 216–230.
- [40] C. Cortes, P. Haffner, and M. Mohri. Rational Kernels: Theory and Algorithms. *Journal of Machine Learning Research (JMLR)*, 2004. 5: 1035–1062.
- [41] C. Couprie, L. Grady, L. Najman, and H. Talbot. Power watershed: A unifying graph-based imization framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.
- [42] J. Cousty, G. Bertrand, L. Najman, and M. Couprie. Watershed cuts: Minimum spanning forests and the drop of water principle. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009. 31: 1362–74.
- [43] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *NIPS*. 2013 2292–2300.
- [44] M. L. Daggitt, A. J. T. Gurney, and T. G. Griffin. Asynchronous convergence of policy-rich distributed bellman-ford routing protocols. *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, Budapest Hungary, 2018 103–116.
- [45] J. De, X. Zhang, F. Lin, and L. Cheng. Transduction on directed graphs via absorbing random walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. 40: 1770–1784.
- [46] S. Delagrangé, C. Jauvin, and P. Rochon. PypeTree: A Tool for Reconstructing Tree Perennial Tissues from Point Clouds. *Sensors*, 2014. 14: 4271–4289.
- [47] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1959. 1: 269–271.

- [48] H. Dobbs, O. Batchelor, R. Green, and J. Atlas. Smart-tree: Neural medial axis approximation of point clouds for 3d tree skeletonization. *Iberian Conference on Pattern Recognition and Image Analysis*. Springer, 2023 351–362.
- [49] M. Droste, T. Stüber, and H. Vogler. Weighted finite automata over strong bimonoids. *Information Sciences*, 2010. 180: 156–166.
- [50] D. Dua and C. Graff. UCI machine learning repository. 2017.
- [51] S. R. Dubey. A decade survey of content based image retrieval using deep learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 2022. 32: 2687–2704.
- [52] J. Eisner. Parameter estimation for probabilistic finite-state transducers. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*. Association for Computational Linguistics, Philadelphia, Pennsylvania, 2001 1.
- [53] J. E. van Engelen and H. H. Hoos. A survey on semi-supervised learning. *Mach Learn*, 2020. 109: 373–440.
- [54] A. Erdem and M. Pelillo. Graph transduction as a non-cooperative game. X. Jiang, M. Ferrer, and A. Torsello, eds., *Graph-Based Representations in Pattern Recognition*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011 195–204.
- [55] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 1741. 128–140.
- [56] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern recognition*, 2008. 41: 176–190.
- [57] E. Fita Sanmartin, S. Damrich, and F. A. Hamprecht. Probabilistic watershed: Sampling all spanning forests for seeded segmentation and semi-supervised learning. *Advances in Neural Information Processing Systems 32*, 2780–2791. Curran Associates, Inc., 2019.
- [58] E. Fix and J. L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, 1989. 57: 238–247.
- [59] R. W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 1962. 5: 345.
- [60] F. Fouss, M. Saerens, and M. Shimbo. *Algorithms and models for network data and link analysis*. Cambridge University Press, 2016.

- [61] K. Françoisse, I. Kivimäki, A. Mantrach, F. Rossi, and M. Saeuens. A bag-of-paths framework for network data analysis. *Neural Networks*, 2017. 90: 90–111.
- [62] S. Freitas, D. Yang, S. Kumar, H. Tong, and D. H. Chau. Graph vulnerability and robustness: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2022. 35: 5915–5934.
- [63] E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM Journal on Computing*, 1968. 16.
- [64] M. Gondran and M. Minoux. *Graphs, Dioids and Semirings: New Models and Algorithms*. Operations Research/Computer Science Interfaces Series. Springer US, 2008.
- [65] J. C. Gower and G. J. Ross. Minimum spanning trees and single linkage cluster analysis. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 1969. 18: 54–64.
- [66] L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006.
- [67] T. G. Griffin and J. L. Sobrinho. Metarouting. *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '05*. ACM Press, Philadelphia, Pennsylvania, USA, 2005 1.
- [68] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016 855–864.
- [69] G. M. Guisewite and P. M. Pardalos. Minimum concave-cost network flow problems: Applications, complexity, and algorithms. *Annals of Operations Research*, 1990. 25: 75–99.
- [70] G. M. Guisewite and P. M. Pardalos. Algorithms for the single-source uncapacitated minimum concave-cost network flow problem. *Journal of Global Optimization*, 1991. 1: 245–265.
- [71] V. Gurvich. Metric and ultrametric spaces of resistances. *Discrete Applied Mathematics*, 2010. 10.
- [72] D. A. Harville. *Matrix Algebra From a Statistician's Perspective*. Springer-Verlag New York, 1997.

- [73] M. A. Hasan and M. J. Zaki. A survey of link prediction in social networks. *Social network data analytics*, 2011. 243–275.
- [74] G. He, X. Liu, F. Fan, and J. You. Classification-aware semi-supervised domain adaptation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2020 4147–4156.
- [75] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 1998. 13: 18–28.
- [76] C. Hoffman, M. Kahle, and E. Paquette. Spectral gaps of random graphs and applications. *International Mathematics Research Notices*, 2021. 2021: 8353–8404.
- [77] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.
- [78] T. C. Hu. Optimum Communication Spanning Trees. *SIAM Journal on Computing*, 1974.
- [79] H. Huang, S. Wu, D. Cohen-Or, M. Gong, H. Zhang, G. Li, and B. Chen. L_1 -medial skeleton of point cloud. *ACM Transactions on Graphics*, 2013. 32: 1–8.
- [80] L. Huang. Advanced Dynamic Programming in Semiring and Hypergraph Frameworks. *Coling 2008: Advanced Dynamic Programming in Computational Linguistics: Theory, Algorithms and Applications - Tutorial notes*. Coling 2008 Organizing Committee, Manchester, UK, 2008 1–18.
- [81] F. K. Hwang and D. S. Richards. Steiner tree problems. *Networks*, 1992. 22: 55–89. Publisher: Wiley Online Library.
- [82] G. Iacobelli and D. R. Figueiredo. Epicenter of random epidemic spanning trees on finite graphs. *Mathematical Modelling of Natural Phenomena*, 2022.
- [83] W. R. Inc. Mathematica, Version 13.2. 2022. Champaign, IL, 2022.
- [84] A. Jamakovic and S. Uhlig. On the relationship between the algebraic connectivity and graph’s robustness to node and link failures. *2007 Next Generation Internet Networks*. 2007 96–102.
- [85] N. Jean, S. M. Xie, and S. Ermon. Semi-supervised deep kernel learning: Regression with unlabeled data by minimizing predictive variance. *Advances in Neural Information Processing Systems*, 2018. 31.

- [86] E. Jenner, E. F. Sanmartín, and F. A. Hamprecht. Extensions of karger’s algorithm: Why they fail in theory and how they are useful in practice. *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021 4602–4611.
- [87] D. S. Johnson, M. Minkoff, and S. Phillips. The prize collecting steiner tree problem: theory and practice. *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*. 2000 760–769.
- [88] F. Kaiser and D. Witthaut. Topological theory of resilience and failure spreading in flow networks. *Phys. Rev. Res.*, 2021. 3: 023161.
- [89] D. R. Karger. Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. *Soda*, vol. 93. Citeseer, 1993 21–30.
- [90] A. Kasperski and P. Zieliński. On the approximability of robust spanning tree problems. *Theoretical Computer Science*, 2011. 412: 365–374.
- [91] J. G. Kemeny and J. L. Snell. Finite markov chains, undergraduate texts in mathematics. 1976.
- [92] K.-H. Kim and S. Choi. Walking on Minimax Paths for k-NN Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2013. 27: 8.
- [93] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*. 2016 .
- [94] G. Kirchhoff. Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird. *Annalen der Physik*, 1847. 148: 497–508.
- [95] I. Kivimäki, M. Shimbo, and M. Saerens. Developments in the theory of randomized shortest paths with a comparison of graph node distances. *Physica A: Statistical Mechanics and its Applications*, 2014. 393: 600–616.
- [96] G. W. Klau, I. Ljubić, P. Mutzel, U. Pferschy, and R. Weiskircher. The fractional prize-collecting steiner tree problem on trees. *Algorithms-ESA 2003: 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003. Proceedings 11*. Springer, 2003 691–702.
- [97] D. J. Klein and M. Randić. Resistance distance. *J Math Chem*, 1993. 12: 81–95.
- [98] T. K. Koo, A. Globerson, X. Carreras, and M. Collins. Structured prediction models via the matrix-tree theorem. *EMNLP-CoNLL*. 2007 .

- [99] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 1956. 7: 48–50. Publisher: American Mathematical Society.
- [100] K. Lang. Newsweeder: Learning to filter netnews. *Proceedings of the Twelfth International Conference on Machine Learning*. 1995 331–339.
- [101] R. Lang, R. Lu, C. Zhao, H. Qin, and G. Liu. Graph-based semi-supervised one class support vector machine for detecting abnormal lung sounds. *Applied Mathematics and Computation*, 2020. 364: 124487.
- [102] P. D. Leenheer. An elementary proof of a matrix tree theorem for directed graphs. *SIAM*, 2020. Publisher: Society for Industrial and Applied Mathematics.
- [103] D. J. Lehmann. Algebraic structures for transitive closure. *Theoretical Computer Science*, 1977. 4: 59–76.
- [104] T. Lengauer and D. Theune. Unstructured path problems and the making of semirings. F. Dehne, J.-R. Sack, and N. Santoro, eds., *Algorithms and Data Structures*, vol. 519, 189–200. Springer-Verlag, Berlin/Heidelberg, 1991. Series Title: Lecture Notes in Computer Science.
- [105] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 2007. 1: 2–es.
- [106] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data>, 2014.
- [107] H. Linmei, T. Yang, C. Shi, H. Ji, and X. Li. Heterogeneous graph attention networks for semi-supervised short text classification. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 2019 4821–4830.
- [108] P. Lippmann, E. Fita Sanmartín, and F. A. Hamprecht. Theory and Approximate Solvers for Branched Optimal Transport with Multiple Sources. *Advances in Neural Information Processing Systems*, 2022. 35: 267–279.
- [109] G. Litvinov. The maslov dequantization, idempotent and tropical mathematics: a brief introduction. *Zapiski Nauchnykh Seminarov POMI*, 2005. 321: 145–182.

- [110] X. Liu, X. Chen, L. Yang, Q. Chen, J. Guo, and S. Wu. Dynamic topology control in optical satellite networks based on algebraic connectivity. *Acta Astronautica*, 2019. 165: 287–297.
- [111] I. Ljubić. Solving steiner trees: Recent advances, challenges, and perspectives. *Networks*, 2021. 77: 177–204.
- [112] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 1982. 28: 129–137.
- [113] S. Loncaric. A survey of shape analysis techniques. *Pattern recognition*, 1998. 31: 983–1001.
- [114] P. Lotito, J.-P. Quadrat, and E. Mancinelli. Traffic assignment & Gibbs-Maslov semirings. G. L. Litvinov and V. P. Maslov, eds., *Contemporary Mathematics*, vol. 377, 209–219. American Mathematical Society, Providence, Rhode Island, 2005.
- [115] U. Luxburg, A. Radl, and M. Hein. Getting lost in space: Large sample analysis of the resistance distance. *Advances in Neural Information Processing Systems*, vol. 23. Curran Associates, Inc., 2010 .
- [116] R. Lyons and Y. Peres. *Probability on trees and networks*, vol. 42. Cambridge University Press, 2017.
- [117] H. Lyu, N. Sha, S. Qin, M. Yan, Y. Xie, and R. Wang. Manifold denoising by nonlinear robust principal component analysis. *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019 .
- [118] Z. Ma, R. Du, J. Xie, D. Sun, H. Fang, L. Jiang, and H. Cen. Phenotyping of silique morphology in oilseed rape using skeletonization with hierarchical segmentation. *Plant Phenomics*, 2023. 5: 0027.
- [119] L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 2008. 9.
- [120] S. A. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 2007. 8: 935–983.
- [121] A. Madkour, W. G. Aref, F. U. Rehman, M. A. Rahman, and S. Basalamah. A survey of shortest-path algorithms. *arXiv preprint arXiv:1705.02044*, 2017.
- [122] B. M. Maggs and S. A. Plotkin. Minimum-cost spanning tree as a path-finding problem. *Information Processing Letters*, 1988. 26: 291–293.

- [123] M. Marchand, K. Gallivan, W. Huang, and P. V. Dooren. Analysis of the neighborhood pattern similarity measure for the role extraction problem. *SIAM Journal on Mathematics of Data Science*, 2021. 3: 736–757.
- [124] M. Mareš. The saga of minimum spanning trees. *Computer Science Review*, 2008. 2: 165–221.
- [125] A. Masone, M. E. Nenni, A. Sforza, and C. Sterle. The minimum routing cost tree problem: State of the art and a core-node based heuristic algorithm. *Soft Computing*, 2019. 23: 2947–2957. Publisher: Springer.
- [126] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.
- [127] D. Mckenzie and S. Damelin. Power weighted shortest paths for clustering Euclidean data. *Foundations of Data Science*, 2019.
- [128] A. Mead. Review of the development of multidimensional scaling methods. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 1992. 41: 27–39.
- [129] Q. Mei, D. Zhang, and C. Zhai. A general optimization framework for smoothing language models on graph structures. *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 2008 611–618.
- [130] A. Mensch and M. Blondel. Differentiable dynamic programming for structured prediction and attention. *ICML*. 2018 .
- [131] W. Meulemans, B. Speckmann, K. Verbeek, and J. Wulms. A framework for algorithm stability and its application to kinetic euclidean MSTs. *LATIN 2018: Theoretical Informatics: 13th Latin American Symposium, Buenos Aires, Argentina, April 16-19, 2018, Proceedings 13*. Springer, 2018 805–819.
- [132] L. Meyer, A. Gilson, O. Scholz, and M. Stamminger. Cherrypicker: Semantic skeletonization and topological reconstruction of cherry trees. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023 6243–6252.
- [133] M. Mohri. Semiring Frameworks and Algorithms for Shortest-Distance Problems. *Journal of Automata, Languages and Combinatorics*, 2002. 7: 321–350.
- [134] C. Monma and S. Suri. Transitions in geometric minimum spanning trees. *Proceedings of the seventh annual symposium on Computational geometry*. 1991 239–249.

- [135] L. Mourot, L. Hoyet, F. Le Clerc, F. Schnitzler, and P. Hellier. A survey on deep learning for skeleton-based human animation. *Computer Graphics Forum*, vol. 41. Wiley Online Library, 2022 122–157.
- [136] D. Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.
- [137] B. Nadler, N. Srebro, and X. Zhou. Semi-supervised learning with the graph Laplacian: the limit of infinite unlabelled data. *Proceedings of the 22nd International Conference on Neural Information Processing Systems, NIPS'09*. Curran Associates Inc., Red Hook, NY, USA, 2009 1330–1338.
- [138] M. E. Newman. Random graphs with clustering. *Physical review letters*, 2009. 103: 058701.
- [139] M. Niendorf and A. R. Girard. Robustness of communication links for teams of unmanned aircraft by sensitivity analysis of minimum spanning trees. *2016 American Control Conference (ACC)*. IEEE, 2016 4623–4629.
- [140] F. Paul, Y. Arkin, A. Giladi, D. A. Jaitin, E. Kenigsberg, H. Keren-Shaul, D. Winter, D. Lara-Astiaso, M. Gury, and A. Weiner. Transcriptional heterogeneity and lineage commitment in myeloid progenitors. *Cell*, 2015. 163: 1663–1677. Publisher: Elsevier.
- [141] M. D. Penrose. On k-connectivity for a geometric random graph. *Random Structures & Algorithms*, 1999. 15: 145–164.
- [142] K. B. Petersen, M. S. Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 2008. 7: 510.
- [143] J.-E. Pin. Tropical Semirings. J. Gunawardena, ed., *Idempotency (Bristol, 1994)*, Publ. Newton Inst. 11, 50–69. Cambridge Univ. Press, Cambridge, 1998.
- [144] C. Pozrikidis. Node degree distribution in spanning trees. *Journal of Physics A: Mathematical and Theoretical*, 2016. 49: 125101.
- [145] Y. Reddy, P. Viswanath, and B. E. Reddy. Semi-supervised learning: A brief review. *Int. J. Eng. Technol*, 2018. 7: 81.
- [146] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. *AAAI*. 2015 .

- [147] W. Saelens, R. Cannoodt, H. Todorov, and Y. Saeys. A comparison of single-cell trajectory inference methods. *Nature Biotechnology*, 2019. 37: 547–554.
- [148] M. Saerens, Y. Achbany, F. Fouss, and L. Yen. Randomized shortest-path problems: Two related models. *Neural computation*, 2009. 21: 2363–2404.
- [149] P. K. Saha, G. Borgefors, and G. S. di Baja. A survey on skeletonization algorithms and their applications. *Pattern recognition letters*, 2016. 76: 3–12.
- [150] S. Sattari and F. Didehvar. A metaheuristic algorithm for the minimum routing cost spanning tree problem. *Iranian Journal of Operations Research*, 2015. 6: 65–78. Publisher: Iranian Journal of Operations Research.
- [151] E. Schröder. Vier combinatorische probleme. *Z. Math. Phys*, 1870. 15: 361–376.
- [152] R. Schwartz, S. Thomson, and N. A. Smith. Bridging CNNs, RNNs, and Weighted Finite-State Machines. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 2018 295–305.
- [153] M. Senelle, S. García-Díez, A. Mantrach, M. Shimbo, M. Saerens, and F. Fouss. The sum-over-forests density index: Identifying dense regions in a graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014. 36: 1268–1274.
- [154] P. Sharma, S. Singh, D. Ghosh, and R. Chandrasekaran. Robust discrete spanning tree problem: local search algorithms. *OPSEARCH*, 2022. 59: 632–644.
- [155] I. Simon. Limited subsets of a free monoid. *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*. 1978 143–150. ISSN: 0272-5428.
- [156] D. Singaraju, L. Grady, and R. Vidal. Interactive image segmentation via minimization of quadratic energies on directed graphs. *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008 1–8.
- [157] W. D. Smith. How to find Steiner minimal trees in Euclidean d-space. *Algorithmica*, 1992. 7: 137–177. Publisher: Springer.
- [158] Z. Song, X. Yang, Z. Xu, and I. King. Graph-based semi-supervised learning: A comprehensive review. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [159] K. Street, D. Risso, R. B. Fletcher, D. Das, J. Ngai, N. Yosef, E. Purdom, and S. Dudoit. Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. *BMC genomics*, 2018. 19: 1–16. Publisher: Springer.

- [160] J. Stuhmer, P. Schroder, and D. Cremers. Tree shape priors with connectivity constraints using convex relaxation on general graphs. *Proceedings of the IEEE International conference on Computer Vision*. 2013 2336–2343.
- [161] Z. Su, Z. Huang, J. Ai, X. Zhang, L. Shang, and F. Zhao. Enhancing the scalability of distance-based link prediction algorithms in recommender systems through similarity selection. *PloS one*, 2022. 17: e0271891.
- [162] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. *2003 Shape Modeling International*. IEEE, 2003 130–139.
- [163] A. Tagliasacchi, T. Delame, M. Spagnuolo, N. Amenta, and A. Telea. 3d skeletons: A state-of-the-art report. *Computer Graphics Forum*, vol. 35. Wiley Online Library, 2016 573–597.
- [164] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1972. 1: 146–160.
- [165] A. Teixeira, P. Monteiro, J. Carriço, M. Ramirez, and A. Francisco. Spanning edge betweenness. *Workshop on Mining and Learning with Graphs*, vol. 24. 2013 27–31.
- [166] A. S. Teixeira, P. T. Monteiro, J. A. Carriço, M. Ramirez, and A. P. Francisco. Not seeing the forest for the trees: size of the minimum spanning trees (MSTs) forest and branch significance in MST-based phylogenetic analysis. *Plos one*, 2015. 10: e0119315.
- [167] F.-S. Tsen, T.-Y. Sung, M.-Y. Lin, L.-H. Hsu, and W. Myrvold. Finding the most vital edge with respect to the number of spanning trees. *IEEE Transactions on Reliability*, 1994. 43: 600–602.
- [168] W. T. Tutte. *Graph theory*. Encyclopedia of mathematics and its applications. Addison-Wesley, 1984.
- [169] H. Wang, C. Ding, and H. Huang. Directed graph learning via high-order co-linkage analysis. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2010 451–466.
- [170] Y. Wang, Y. Xu, S. Tsogkas, X. Bai, S. Dickinson, and K. Siddiqi. Deepflux for skeletons in the wild. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019 5287–5296.
- [171] D. M. Warme, P. Winter, and M. Zachariasen. *Exact algorithms for plane Steiner tree problems: A computational study*. Springer, 2000.

- [172] T. R. Willemain and M. V. Bennett. The distribution of node degree in maximum spanning trees. *Journal of Statistical Computation and Simulation*, 2002. 72: 101–106.
- [173] R. J. Wilson. *Introduction to graph theory*. Pearson Education India, 1979.
- [174] G. Winkler. *Image analysis, random fields and Markov chain Monte Carlo methods: a mathematical introduction*, vol. 27. Springer Science & Business Media, 2012.
- [175] F. A. Wolf, F. K. Hamey, M. Plass, J. Solana, J. S. Dahlin, B. Göttgens, N. Rajewsky, L. Simon, and F. J. Theis. PAGA: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells. *Genome Biology*, 2019. 20: 59.
- [176] S. Wolf, C. Pape, A. Bailoni, N. Rahaman, A. Kreshuk, U. Kothe, and F. Hamprecht. The mutex watershed: efficient, parameter-free image partitioning. *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018 546–562.
- [177] R. T. Wong. Worst-case analysis of network design problem heuristics. *SIAM Journal on Algebraic Discrete Methods*, 1980. 1: 51–63. Publisher: SIAM.
- [178] W. K. Wong, Z. Lai, J. Wen, X. Fang, and Y. Lu. Low-rank embedding for robust image feature extraction. *IEEE Transactions on Image Processing*, 2017. 26: 2905–2917.
- [179] B. Y. Wu and K.-M. Chao. *Spanning trees and optimization problems*. CRC Press, 2004.
- [180] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020. 32: 4–24.
- [181] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2021. 2: 109–127.
- [182] H. Xu, N. Gossett, and B. Chen. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Transactions on Graphics (TOG)*, 2007. 26: 19–es.
- [183] L. Xu, A. Krzyżak, and C. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Trans. Syst. Man Cybern.*, 1992. 22: 418–435.

- [184] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han. Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17. Association for Computing Machinery, New York, NY, USA, 2017 1245–1254.
- [185] Z. Yang, W. Cohen, and R. Salakhudinov. Revisiting semi-supervised learning with graph embeddings. *International conference on machine learning*. PMLR, 2016 40–48.
- [186] L. Yen, M. Saerens, A. Mantrach, and M. Shimbo. A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08. Association for Computing Machinery, New York, NY, USA, 2008 785–793.
- [187] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich. Local higher-order graph clustering. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17. Association for Computing Machinery, New York, NY, USA, 2017 555–564.
- [188] W. I. Zangwill. Minimum concave cost flows in certain networks. *Management Science*, 1968. 14: 429–450.
- [189] Q. Zhang, M. Li, and Y. Deng. Measure the structure similarity of nodes in complex networks based on relative entropy. *Physica A: Statistical Mechanics and its Applications*, 2018. 491: 749–763.
- [190] Y.-M. Zhang, K. Huang, and C.-L. Liu. Fast and robust graph-based transductive learning via minimum tree cut. *2011 IEEE 11th International Conference on Data Mining*. 2011 952–961.
- [191] G. X. Zheng, J. M. Terry, P. Belgrader, P. Ryvkin, Z. W. Bent, R. Wilson, S. B. Ziraldo, T. D. Wheeler, G. P. McDermott, and J. Zhu. Massively parallel digital transcriptional profiling of single cells. *Nature communications*, 2017. 8: 14049. Publisher: Nature Publishing Group UK London.
- [192] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *NIPS*. 2004 321–328.

-
- [193] D. Zhou, T. Hofmann, and B. Schölkopf. Semi-supervised learning on directed graphs. *Advances in Neural Information Processing Systems 17*, 1633–1640. MIT Press, 2005.
- [194] D. Zhou, J. Huang, and B. Schölkopf. Learning from labeled and unlabeled data on a directed graph. *Proceedings of the 22nd International Conference on Machine Learning, ICML '05*. Association for Computing Machinery, New York, NY, USA, 2005 1036–1043.
- [195] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI open*, 2020. 1: 57–81.
- [196] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. *ICML*. 2003 912–919.
- [197] I. Ziamtsov, K. Faizi, and S. Navlakha. Branch-pipe: Improving graph skeletonization around branch points in 3d point clouds. *Remote Sensing*, 2021. 13: 3802.

List of Figures

1.1	Data Represented as Graphs	2
1.2	Comparison Between Transductive and Inductive Semi-Supervised Learning	4
1.3	Probability Distribution over Spanning Trees of a Complete Graph with 4 Nodes	5
1.4	Impact of Graph Node Distance in Semi-Supervised Classification	7
2.1	Overview DProbWS	12
2.2	Procedure to Transform an In-Forest in $\mathcal{F}_{s_1, q}^{\vec{s}_2}$ to a Spanning In-Tree in \bar{G}_1 and \bar{G}_2 as Defined in Lemma 2.3.3	19
2.3	DProbWS Performance Comparison	27
3.1	Expectation and Standard Deviation of Unweighted Degree in a Spanning Tree of a Grid Graph	44
3.2	Expectation and Standard Deviation of Weighted Degree in a Spanning Tree of a Weighted Graph	45
4.1	Schematic Path Relevance in the Log-Norm Distance	57
4.2	Necessity of the Subadditivity of g for the Triangle Inequality to Hold	61
5.1	Euclidean Central Spanning Tree Family of Problems with and without Steiner Points	66
5.2	Threshold Function $\alpha^*(N)$ Guaranteeing Optimal (B)CST Star-Tree	72
5.3	(B)CST Star-Tree Optimality with Respect to $\alpha \gtrsim 1$ in Samples Uniformly Drawn from a Rectangle	73
5.4	(B)CST Robustness Analysis	75
5.5	mST, (B)CST and BMRCT of the Paul Dataset	77
5.6	Realizable BCST Solutions Topologies from a Full Tree Topology with 4 Terminals	79
5.7	CST and BCST Topology Correspondence	79
5.8	Optimal CST and BCST Topologies May Not Be Derived from Each Other Given the Same Terminal Configuration	81

5.9	Branching Angles at Steiner Point	82
5.10	Optimal Angles in Degree-4 <i>SP</i> Require $\psi \leq \gamma_i$	83
5.11	Splitting Collapsed <i>SP</i> while Preserving Optimal Angles	84
5.12	<i>SPs</i> of Degree 4 in the Plane are Likely not Feasible for $\alpha \in]0.5, 1[$	86
5.13	mSTreg Heuristic	88
5.14	Bruteforce mSTreg Benchmark	91
5.15	Steiner and MRCT OR Library Dataset Benchmark	92
5.16	Comparing mSTreg and GRASP_PR	94
6.1	BCST-Based Skeletonization Steps	99
6.2	BCST Skeletonization Pruning Algorithm	104
6.3	Impact of λ in the Skeletonization Pruning Algorithm	105
6.4	Performance Comparison Skeletons at Different Density Levels (Apple Tree)	110
6.5	Performance Comparison Skeletons with Missing Data (Apple Tree)	111
6.6	Performance Comparison Skeletons with Noisy Data (Apple Tree)	112
C.1	Computation Ranked Absorption Probability	149
D.1	(B)CST Examples from Rectangle Uniform Distribution	152
D.2	(B)CST Examples from Triangle Uniform Distribution	153
D.3	(B)CST Examples from Non-Convex Uniform Distribution	154
D.4	Necessity of the Triangle Inequality for Path Graph to be (B)CST Optimum as $\alpha \rightarrow -\infty$	164
D.5	Bijection Between Derivable CST Topologies from a Full Tree Topology and Their Terminal-Separating Forests	168
D.6	Branching Angles at Steiner Point	169
D.7	Splitting Collapsed <i>SP</i> While Preserving Optimal Angles	171
D.8	BCST Time Complexity	176
D.9	mSTreg Heuristic with Additional Sampled Points	177
D.10	Performance <i>SPs</i> Collapse Strategies	180
D.11	BCST Bruteforce Benchmark with Respect to α	181
D.12	CST Bruteforce Benchmark with Respect to α	182
E.1	Performance Comparison Skeletons at Different Density Levels (Apple Tree)	186
E.2	Performance Comparison Skeletons at Different Density Levels (Pine Tree)	187
E.3	Performance Comparison Skeletons at Different Density Levels (Pine Tree)	188
E.4	Performance Comparison Skeletons with Missing Data (Pine Tree)	189
E.5	Performance Comparison Skeletons with Noisy Data (Pine Tree)	190

List of Tables

4.1	Family of Log-Norm Distances	49
4.2	Semiring Examples	51
6.1	Comparison of Skeletonization Methods Accross Varying Density Levels	109
6.2	Comparison of Skeletonization Methods with Missing Data	111
6.3	Comparison of Skeletonization Methods in Noisy Data	113

List of Algorithms

1	Directed Probabilistic Watershed / Directed Random Walker	23
2	mSTreg Heuristic	90
3	Skeletonization Pruning Algorithm	106