Journal of Machine Learning Research 21 (2020) 1-68 Submitted 12/19; Revised 9/20; Published 10/20

# Mining Topological Structure in Graphs through Forest Representations

**Robin Vandaele**[1,2] ROBIN.VANDAELE@UGENT.BE
**Yvan Saeys**[2] YVAN.SAEYS@IRC.VIB-UGENT.BE
**Tijl De Bie**[1] TIJL.DEBIE@UGENT.BE
[1] *IDLab, Department of Electronics and Information Systems*
*Ghent University*
*Technologiepark-Zwijnaarde 19, 9052 Gent, Belgium*
[2] *Data mining and Modelling for Biomedicine (DaMBi),*
*VIB Inflammation Research Center*
*Technologiepark-Zwijnaarde 927, 9052 Gent, Belgium*

**Editor:** Karsten Borgwardt

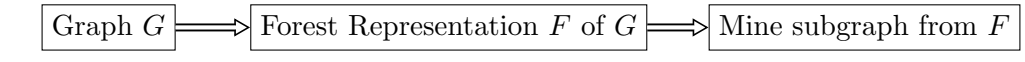
## Abstract

We consider the problem of inferring simplified topological substructures—which we term *backbones*—in metric and non-metric graphs. Intuitively, these are subgraphs with 'few' nodes, multifurcations, and cycles, that model the topology of the original graph well. We present a multistep procedure for inferring these backbones. First, we encode local (geometric) information of each vertex in the original graph by means of the *boundary coefficient* (BC) to identify 'core' nodes in the graph. Next, we construct a *forest representation* of the graph, termed an *f-pine*, that connects every node of the graph to a local 'core' node. The final backbone is then inferred from the *f*-pine through CLOF (***C****onstrained* ***L****eaves* ***O****ptimal sub****F****orest*), a novel graph optimization problem we introduce in this paper. On a theoretical level, we show that CLOF is NP-hard for general graphs. However, we prove that CLOF can be efficiently solved for forest graphs, a surprising fact given that CLOF induces a nontrivial monotone submodular set function maximization problem on tree graphs. This result is the basis of our method for mining backbones in graphs through forest representation. We qualitatively and quantitatively confirm the applicability, effectiveness, and scalability of our method for discovering backbones in a variety of graph-structured data, such as social networks, earthquake locations scattered across the Earth, and high-dimensional cell trajectory data.

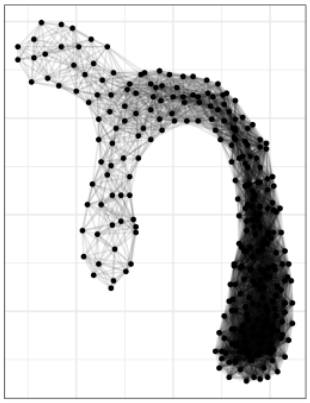
**Keywords:** topological data analysis, graph mining, metric spaces, visualization, topological skeletonization, cluster coefficient, cell trajectory inference

## 1. Introduction

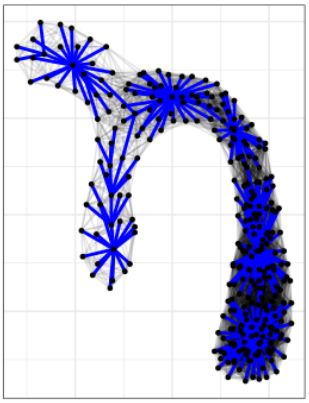
*Motivation.* Many real-world graphs, whether given (social networks, road networks, image webs, ...) or derived from point cloud data (gene expression data of differentiating cells, GPS traces, earthquake locations, galaxy coordinates in space, ...), exhibit topologies of which the underlying structure can be naturally represented using a much 'simpler' subgraph, as shown in Figure 1d. I.e., although the topology of the original graph might



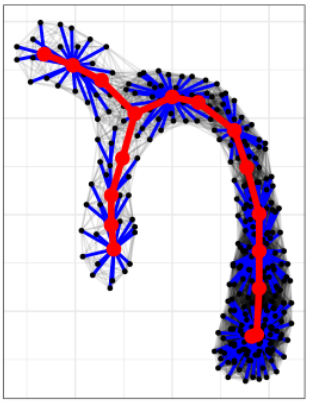

(a) High level overview of our introduced method for mining substructures in graphs.



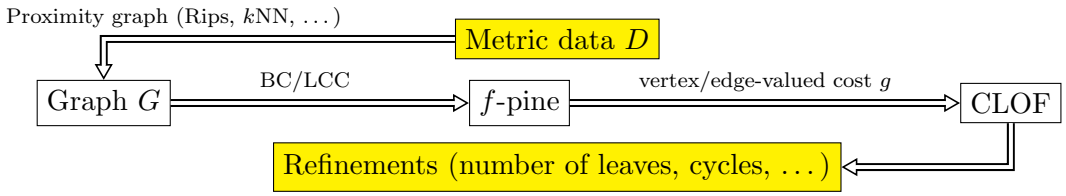
(b) The original graph $G$.



(c) A forest representation $F$ of $G$.



(d) A backbone (red) of $G$ mined through $F$.




(e) Detailed overview of our method for mining topological subtructures in graph-structured data. Yellow blocks denote pre- and post-processing steps.

Figure 1: Overview of the method proposed in this paper.

be complex (e.g., in terms of degree sequences, multifurcations, cycles, ...), many vertices lie often close to some core subgraph, having a much 'simpler' topology, from which they emerge. We call this core topological substructure the *backbone* of the graph. Figure 1 shows a toy example of such particular type of graph, as well as an overview of the method we will introduce for inferring these backbones in graphs.

Identifying and visualizing the topological structure of backbones in graphs, and hence, of the original graphs, is an active topic of research, applicable to many fields of science (Aanjaneya et al., 2012; Cannoodt et al., 2016; Choi et al., 2010; De Baets et al., 2015; Nicolau et al., 2011; Rizvi et al., 2017; Vandaele et al., 2019a). E.g., in biology, inferring backbones in high-dimensional cell trajectory data allows one to model the dynamic changes immune cells undergo to protect our body against environmental and internal threats (Saelens et al., 2019). In geoinformatics, inferring backbones from GPS coordinates allows one to obtain up-to-date road maps, which are critical for many applications, such as GPS-based navigation services and autonomous transportation (He et al., 2018). In social sciences, backbones allow one to model how different communities are connected, and identify which figures play a key role in these connections (Bedi and Sharma, 2016). Nevertheless, inferring such backbones is generally a difficult task, as it involves dealing with issues such as topological bias, noise, outliers, in addition to computational problems such as intractability.

In this paper, we build upon and extend our earlier work (Vandaele et al., 2019b), where we introduced the *boundary coefficient* (Section 2.1) and *$f$-pine* (Section 2.2) of a graph. We will *mine backbones* in given graphs *through forest representations* (Figure 1). A simple, but yet a crucial and powerful intermediate step for many practical purposes, which we demonstrate throughout this entire paper.

*Example.* Figure 1 illustrates how our method results in the identification and location of the backbone in a synthetic point cloud data set $D$. The points in $D$ correspond to the vertices of the proximity graph $G$ (constructed from $D$) shown in Figure 1b. The forest representation $F$ of $G$ (Figure 1c) connects every vertex of $G$ to a local core point. The problem is now to infer the backbone from $F$, and CLOF (***C**onstrained **L**eaves **O**ptimal sub**F**orest*, Section 2.3) is the answer we provide to that (Figure 1d).

We emphasize that a straightforward optimization in $G$ for identifying and locating core topological structures, would introduce many difficulties in terms of accuracy, robustness, and scalability, as we will discuss in Section 1.2. Hence, apart from introducing a new method that overcomes these issues, a main purpose of our paper is to illustrate the effectiveness of intermediate forest representations for this task (Sections 1.2, 2.3 & 3).

### 1.1 Contributions

- We introduce a method for inferring backbones in a wide variety of graphs $G$ (Section 2.4), consisting of two major steps (Figure 1e).
  1. A 'core'-measure $f$ (Section 2.1) is used to construct an *$f$-pine* (Section 2.2), which gives the representation of the graph as illustrated in Figure 1c. More formally, an $f$-pine is a forest subgraph connecting nodes to local minima of $f$ (core nodes) that may be efficiently computed through the minimum spanning tree (MST) algorithm (Proposition 13). For unweighted graphs, we show the ordinary local cluster coefficient (LCC) to be sufficient as a core measure (Section 3). For weighted graphs, we use the *boundary coefficient* (BC) for this purpose (Section 2.1). This coefficient is accompanied by extensive theoretical analysis, comparisons, as well as an efficient formula for computation (Theorem 8).
  2. Our newly introduced graph-optimization problem in Section 2.3, termed the '**C**onstrained **L**eaves **O**ptimal sub**F**orest'-problem (CLOF), is used to effectively infer backbones through these pines, as illustrated in Figure 1d.
- We prove that CLOF is NP-hard for general graphs (Section 2.3), but induces a nontrivial monotone submodular set function maximization problem subject to a cardinality constraint on tree graphs, for which a greedy approach provides an exact solution in polynomial time (Section 2.3.1). Furthermore, we show how this allows an efficient solution in practice for the case of forest graphs as well (Section 2.3.2).
- We qualitatively and quantitatively show that our method leads to effective topological models, i.e., backbones, in multiple real-world graph-structured data sets, arising from social networks, geosciences, and biology (Section 3).
- We summarize how our method improves on state-of-the-art approaches (Section 4), and opens up new possibilities for further improvements on both the theoretical and experimental level (Section 5).

### 1.2 Related Work

An earlier version of the current work was presented at a non-archival workshop (Vandaele et al., 2019b), which introduced the *boundary coefficient* and *$f$-pine*. In this paper, we show how these concepts lead to a novel backbone inference method in graphs through CLOF.

In the rest of this section, we summarize the, to our knowledge, current methods that may deal or help with locating and/or visualizing backbones. We start with *Facility Location in Networks*, discussing their issues that lead us to introducing intermediate forest representations to effectively mine backbones in graphs. Next, as we introduced the *boundary coefficient* to quantify the 'coreness' of nodes for constructing this representation, we discuss how current existing vertex measures are insufficient for the purpose of backbone inference. Other methods that identify graph-structured models, but are unable to deal with graph-structured data as input, will also be discussed shortly. Finally, we discuss methods from the field of *Topological Data Analysis* (TDA). This section also includes a limited background on *persistent homology*, which we will use to identify cycles missing from our forest-structured backbone in Section 2.4.3.

#### 1.2.1 Facility Location in Networks

The general setting of *Facility Location Problems in Networks* (Mesa and Boffey, 1996) is:

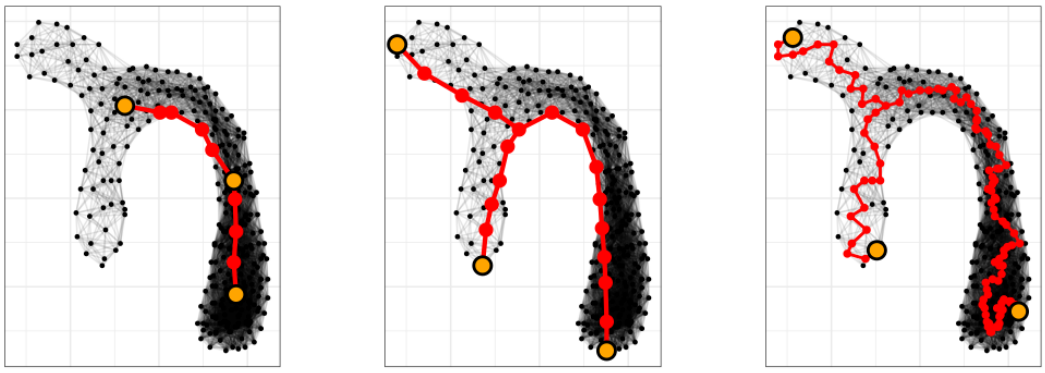
"given a graph $G$, a collection $\mathcal{F}$ of subgraphs of $G$, and a cost function $f : \mathcal{F} \to \mathbb{R}$
*optimize* $f(F)$ *subject to* $F \in \mathcal{F}$."

Note that the term 'facility' in our context refers to 'backbone'. Both the inference of $f$-pines (Section 2.2) and solving CLOF (Section 2.3), will be facility location problems in networks. A more commonly known example of subgraph inferred through a facility location problem is the *minimum spanning tree* (MST). Here, $\mathcal{F}$ is the set of spanning trees of $G$ (forests if $G$ is disconnected), and $f$ maps a tree onto the sum of the weight of its included edges, which is the cost to be minimized. *Steiner trees* generalize this concept. They minimize the same cost function as minimum spanning trees, but are only required to cover a given set of nodes, called *terminals*. Finding a minimum spanning tree can be done in linear time (Chazelle, 2000), whereas finding a Steiner tree is an NP-hard problem (Garey and Johnson, 1990). In Section 3, we show that neither facility is effective for inferring backbones that model the underlying topology of a graph well.

Existing facility location problems come with a variety of issues that prevent them to effectively identify and locate core topological structure in graphs, summarized below.

*Computational complexity.* Many facility location problems in networks are NP-hard for general graphs (Mesa and Boffey, 1996). Certain formulations even lead to NP-hard problems when the original graph is a tree graph (Crainic and Laporte, 1998). In contrast to this, we present effective and efficient algorithms that provide an exact solution to our introduced facility locations problems, which are identifying an $f$-pine and solving CLOF in forest representations (Section 2 and Appendix B).

*Sensitive to outliers.* Outliers are harmful when either the constraint $\mathcal{F}$ (Kim et al., 1989) or the cost $f$ (Aneja and Nair, 1992) specifies that all nodes in the original graph should lie close to the facility. Furthermore, facilities may 'pass through' outliers to reach one



(a) An approximated Steiner tree (Sadeghi and Fröhlich, 2013) in red, which we constructed through three terminal nodes/medoids selected by a partitioning around medoids (PAM) algorithm (orange). The selection of these medoids is regarded as a facility location problem in metric spaces (Mitra et al., 2019), and is highly biased towards dense regions.

(b) A subgraph (red) obtained by iteratively chosing farthest points, and connecting them by the shortest path between them and the current tree structure. These paths always take 'shortcuts' when available, shifting them away from the true core in the presence of curvature. Furthermore, outliers are especially harmful when connecting to farthest points (Section 3).

(c) The output (red) of our method presented in Figure 1e, where we replaced the BC-pine as a forest representation by the ordinary minimum spanning tree (MST). The selected leaves during CLOF are shown in orange. Subgraphs minimizing the maximum edge weight, such as the MST, are biased towards including low-weight edges, leading to 'wiggled' results.

Figure 2: Subgraphs mined from an original graph $G$ (black), (c) with and (a-b) without intermediate forest representation. Comparing these results with Figure 1d, (a-b) illustrate the usefulness of an intermediate forest representation for mining topological substructures in graphs, whereas (c) illustrates the importance of designing an effective representation for this purpose, both the subject of this paper. Note that all methods may be regarded as a combination of selecting important nodes and constructing a subgraph through these. We will discuss these methods in detail in Section 3.

region from another, shifting the facility from the true backbone of the graph. Our method overcomes these issues by marking outliers as leaves in our forest representations.

*Sensitive to density.* To overcome the sensitivity to outliers, the constraint or cost may be postulated in terms of the average/mean distance of the facility to all other nodes (Richey, 1990). However, such approach tends to fail revealing important structure in case of a non-uniform density across the underlying topology (Figure 2a). In contrast to this, our method effectively infers backbones that extend across the entire original graph, while remaining near the core of the graph (Figure 1d).

*Not or too topologically constrained.* Facility location problems are mainly considered in topics such as routing, logistics, and dispatching (Hu et al., 2018). In these scenarios, rather than representing the topological model underlying the graph, the objective of the facility is

to reach all nodes of the original graph as close as possible, while maintaining a low cost of the facility. These facility location problems are insufficient for inferring topological models underlying graphs. There may not be any constraints on the topological complexity of the facility (e.g., in terms of the number of leaves or multifurcations), allowing for an arbitrary complex backbone that fails to provide insight into the underlying structure. E.g., the only topological restriction on the facility may simply be that the facility is a tree (Richey, 1990). In other cases, the facility is topologically too constrained. In particular, facility location problems may also search for a specific path (Avella et al., 2005), which is not suited to capture the underlying topology in many practical examples.

Steiner trees allow some control over the topological complexity of the final facility through the number of terminals that are specified (Akoglu et al., 2013). However, the presence of outliers or non-uniform density may be harmful when selecting these terminals in an unsupervised manner (Figure 2a). Furthermore, the topological complexity, such as the number of leaves, of the resulting (approximated) Steiner tree is often not consistent with the number of terminals (Figure 2a and Section 3).

In contrast to these methods, our objective is to reveal the underlying topology of a graph—the cost of which does not matter to us—in a robust and effective way. Hence, we will be able to provide a method for tuning the topological complexity of our backbone in a data- and scale-independent way (Section 2.3).

*Topological bias.* Many facilities may just not be meaningful representations for the underlying topology. E.g., a trivial example is the longest path through a graph (if existing), which may 'wiggle' through the entire graph without reflecting the true underlying topology, even if this is linear. Furthermore, other facilities may only be meaningful in the absence of outliers (as also discussed above), in the absence of curvature (Figure 2b), or may be biased to include mostly low-weight edges due to the minimization of a sum or maximum of the edge weights of the facility. Extreme examples of this are the MST and its subgraphs, which 'wiggle' through the entire graph (Figure 2c).

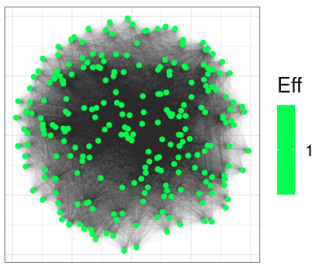
The problems listed above are the main reasons why we introduce the forest representation as an intermediate step for mining topological substructures, as we overcome all of these by designing such representation of our graph (compare Figures 1 & 2).

### 1.2.2 Existing Core Measures in Graphs

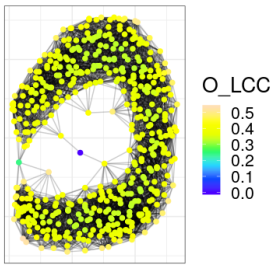
A crucial part of our method will be using a vertex measure to quantify the coreness of a node $v$ of a graph $G = (V, E)$. Certain vertex measures that might be used to identify such nodes already exist. A well known example is the *local cluster coefficient* (Watts and Strogatz, 1998). For every node $v \in V$ with degree $\delta(v) > 1$, it is defined as

$$\mathrm{LCC}(v) \coloneqq \frac{1}{\delta(v)(\delta(v)-1)} \sum_{u \neq w \in \mathcal{N}(v)} \mathbf{1}_{\{u,w\} \in E}\,,$$

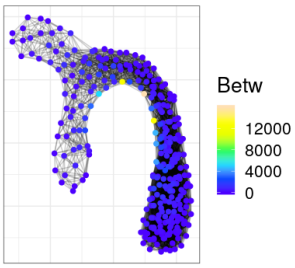
where $\mathcal{N}(v)$ denotes the set of neighbors of $v$ in $V$, and $\mathbf{1}_{\{u,w\}\in E} = 1$ if $\{u, w\} \in E$ and $\mathbf{1}_{\{u,w\}\in E} = 0$ otherwise. Hence, $\mathrm{LCC}(v)$ is the number of closed wedges adjacent to $v$, divided by the number of (all) wedges adjacent to $v$. For nodes $v$ with $\delta(v) = 1$, $\mathrm{LCC}(v)$ is either undefined, or (commonly) defined as 0.




(a) Apart from lacking scalability (taking more than 24 minutes to compute on a complete graph on 250 vertices using the brainwaver library in R), the *local efficiency* is not applicable to fully weighted networks. By mapping every node to the same value, it is unable to detect the true core nodes of this network.




(b) The presence of only a small amount of outliers makes it difficult for *Onella's generalized local cluster coefficient*—one of the many vertex measures generalizing the local cluster coefficient to weighted networks—to identify the core nodes near the underlying C-structured topology of this network.




(c) *Betweenness centrality*, measuring how many shortest paths go through a particular node, does not perform well when the true underlying topology is curved. Shortest paths will always take shortcuts when available, shifting them from the true core nodes of our underlying Y-structured topology.

Figure 3: Various possible existing 'core' measures in Rips graphs $\mathcal{R}_{10}(D)$ built from 2D point cloud data sets $D$. None of them capture the true core nodes of the graph well.

We will show that the LCC is particularly useful to our method for investigating topological structure in a variety of unweighted graphs (Section 3). However, we found its generalizations to weighted graphs—an extensive summary of these is given by Wang et al. (2017))—as well as other existing measures trying to quantify the coreness of a node, such as graph centrality measures (Klein, 2010; Hage and Harary, 1995; Barthélemy, 2004), to be insufficient for many of our practical examples (Figure 3). These were not designed for the purpose of identifying or visualizing a global core structure within a wide variety of graph-structured data sets. As such, they lack important properties of the boundary coefficient, such as scalability, applicability to fully weighted networks (compare Figure 3a to 6a), robustness to outliers (compare Figure 3b to 6b), and the ability to deal with nonlinear substructures (compare Figure 3c to 6c).

Vandaele et al. (2019b) recently demonstrated the superior effectiveness of the boundary coefficient over existing measures for the purpose of topological data analysis of graphs, on both a qualitative and quantitative level. This is part of the contribution of this paper. We provide a formal discussion why the boundary coefficient outperforms these measures for this purpose in Section 2.1.3 (Remark 5).

### 1.2.3 Topological skeletonization, thinning, or fitting in structured data

Various methods have been developed for extracting underlying graph-structured topologies when the input data satisfies specific structural criteria. These criteria are generally not satisfied in any given graph. E.g., many *topological skeletonization* algorithms deal

with thinning structured input data, such as 2D or 3D images, towards a graph-structured skeleton of the object represented by the image (Wang et al., 2018; Abu-Ain et al., 2013; Jin et al., 2016). Other methods based on *principal graphs* (Gorban and Zinovyev, 2010), or many of the *cell trajectory inference methods* (Saelens et al., 2019), such as *Slingshot* (Street et al., 2018), assume the input data to be a finite representation of the underlying topology in a vector space, as they rely on local averaging techniques. Since these methods require the presence of structure that is generally not present in graphs, we do not consider them to be applicable to our problem. However, in Section 3, we will show that our method is comparable to Slingshot—the currently top ranked method in terms of accuracy (Saelens et al., 2019)—for the specific purpose of cell trajectory inference.

#### 1.2.4 Methods from Topological Data Analysis (TDA)

The emergent area of Topological Data Analysis (TDA) (Carlsson, 2009), aims to understand the *shape of data* (Wasserman, 2018). Nevertheless, *persistent homology* (Ghrist, 2008), the most profoundly used and studied tool within TDA, is unable to be straightforwardly applied to our problem. Note that persistent homology only quantifies topological information, and does lead to an actual model. Furthermore—based on this topological information—persistent homology cannot even distinguish between an underlying linear or bifurcating topology through the customary *Vietoris-Rips filtration*. However, we will use this method for identifying *cycles* missing from our forest-structured backbone (Section 2.4.3). Discussing its foundations, however, would require us to introduce concepts from *algebraic topology* (Hatcher, 2002) that are (far) beyond the scope of this paper, and hence, we will instead provide a visual introduction to persistent homology.

*Topological persistence* (Ghrist, 2008) tracks the (dis)appearance of distinct shape features (more spefically, 'holes'), across a *filtration* (Figure 4a), i.e., a sequence of simplicial complexes (Hatcher, 2002)

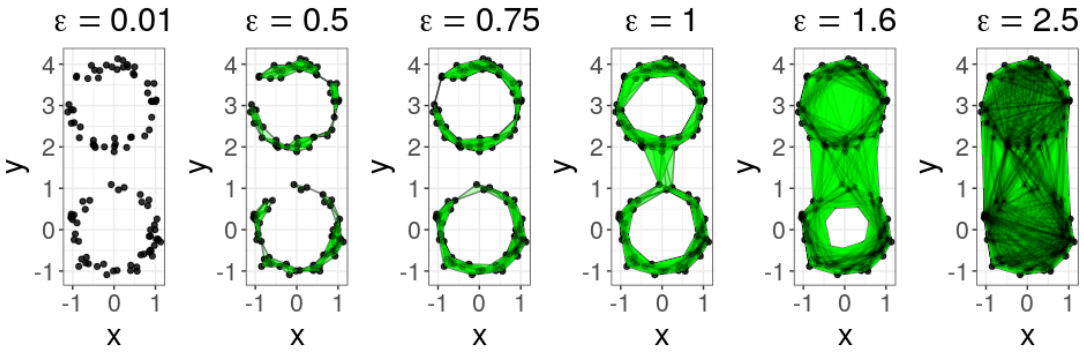
$$\sigma_{\epsilon_1} \subseteq \sigma_{\epsilon_2} \subseteq \ldots \subseteq \sigma_{\epsilon_n} \ ,$$

for an index sequence $\epsilon_1, \ldots, \epsilon_n$. Though manually defined filtrations on a given simplicial complex (such as a graph) are possible (Rieck and Leitte, 2015), the custom illustrative case is that we have a point cloud data set $D$ embedded in a metric space $(M, d)$, and we parameterize the filtration by means of a distance parameter $\epsilon$, corresponding to the *Vietoris-Rips filtration* (Figure 4a):

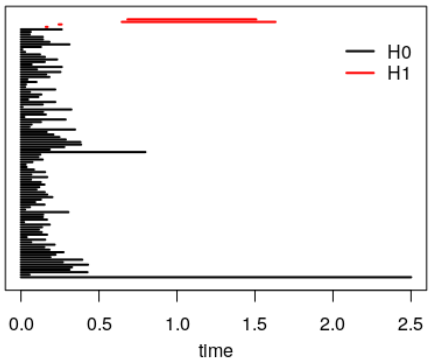
$$\left(\sigma_\epsilon \coloneqq \left\{S \subseteq 2^D : |S| \leq k+2 \wedge \forall x, y \in S : d(x,y) \leq \epsilon\right\}\right)_\epsilon \ ,$$

where $k \in \mathbb{N}$ is a parameter constraining the dimension of topological features (holes) we are in interested in. Any complex in this filtration is called a *Vietoris-Rips complex*. An element $s$ of a particular complex is a $(|s|-1)$-*simplex*, where $|s|$ denotes the cardinality of $s$. If $k = 0$, we will also simply refer to the complex as the *(Vietoris-)Rips* graph.

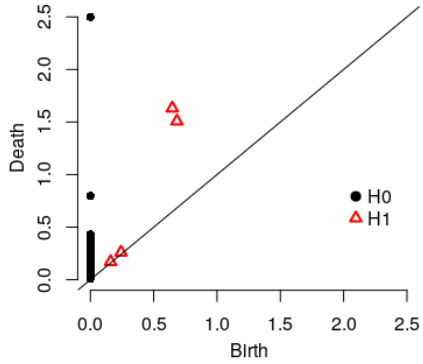
By evaluating how long certain features exist, we are able to deduce *topological invariants*, i.e., *topological features* that are preserved under homeomorphism. In this case, we infer *holes* in the underlying data structure (Medina and Doerge, 2015). The evolution of these (dis)appearing features may be visualized by means of *persistence barcodes*, where the number of bars occurring at a fixed value of $\epsilon$ denotes the $k$-th *Betti number* $\beta_k$, expressing the number of distinct $k$-dimensional holes at index $\epsilon$ in the filtration (Figure 4b). In




(a) Simplicial complexes in the Vietoris-Rips filtration for different distance values $\epsilon$. Nodes represent 0-simplices, edges represent 1-simplices, and green triangles represent 2-simplices in a particular complex of the filtration. At $\epsilon = 0.01$, the corresponding complex consists only of all isolated points. Starting around $\epsilon = 0.5$, two connected components represent the underlying true components, which have a relatively long persistence, i.e., they persist for a 'large' interval of values $\epsilon$. At $\epsilon = 0.75$, two cycles formed by the boundaries of the two 'rings' are present in the complex. At $\epsilon = 1$ the two components have merged, and the complex will stay connected for all further distance values $\epsilon$. At $\epsilon = 1.6$, one cycle is completely 'filled in', whereas the other is still present. Finally, the second cycle will be filled in as well, as seen at $\epsilon = 2.5$. The simplicial complex will continue to grow until each pair of nodes is connected by an edge.




(b) Persistence barcodes obtained from applying persistent homology to $D$ seen as a finite metric space. Bars for connected components (H0) are shown in black, and for cycles (H1) in red. Two bars show to persist for each of these $k$-dimensional holes, $k \in \{0, 1\}$. Short bars are often considered to represent 'topological noise' (Oudot, 2015). The time denotes the distance value $\epsilon$ at which the topological features are present in the filtration. Note that the $y$-axis of the persistence barcodes has no significant meaning in this example.




(c) The results of persistent homology may also be represented by means of persistence diagrams, where a bar persisting from $b$ to $d$ is replaced by a point $(b, d)$ above the diagonal in the first quadrant of the Euclidean plane. The elevation $d-b$ of a point according to this diagonal now corresponds to the persistence of the feature it represents. Higher elevated points correspond to features with a longer persistence.

Figure 4: Persistent homology of point cloud data $D$ representing two disconnected cycles.

this sense, a 0-dimensional hole represents a gap between components, and $\beta_0$ equals the number of connected components in our complex. A 1-dimensional hole represents a cycle (e.g., the hole in a solid ring), a 2-dimensional hole represents a void (e.g., the inside of a balloon), and higher-dimensional holes represent higher-dimensional analogues. Long bars resemble topological features that 'persist' for many consecutive values $\epsilon_i, \epsilon_{i+1}, \ldots, \epsilon_j$, and indicate features of the underlying topology of the point cloud data set (Figure 4). Hence, the naming 'persistent' homology. Persistence barcodes may also be represented by means of *persistence diagrams* (Figure 4c), which are mathematically more convenient to work with (Oudot, 2015). In this representation, a bar persisting from $b$ to $d$ is replaced by a point $(b, d)$ above the diagonal in the first quadrant of the Euclidean plane.

Though persistent homology is an increasingly useful tool to machine learning problems (Hofer et al., 2017; Rieck et al., 2019; Oudot, 2015; Singh et al., 2014; Moor et al., 2019; Garside et al., 2019), one remaining disadvantage is its computational cost, which is cubic in the number of simplices (Otter et al., 2017). Furthermore, when we are interested in cycles, i.e., 1-dimensional holes, the number of simplices itself is cubic in the number of data points, as one needs to store up to triangular relations. Existing approximating algorithms for persistent homology (Silva and Carlsson, 2004; Cavanna et al., 2015) usually construct the filtration on a farthest point sample—using properties of the entire data set to define the simplices—and come with theoretical guarantees that the resulting persistence diagrams are 'close' to the diagrams of the original metric space (Cavanna et al., 2015). However, these guarantees are accompanied by outliers being prone to be selected during the sampling, and topological noise remaining in the resulting barcodes (Section 2.4.3).

As stated above, persistent homology is not straightforwardly applicable to our problem. A linear and a bifurcating topology would both consist of one connected component and no higher-dimensional holes. Hence, persistent homology is currently unable to distinguish between these spaces, and more generally between any tree-shaped (underlying) topologies. However, some possible refinements of persistent homology, as well as the Mapper algorithm in TDA, do allow us to investigate graph-structured topologies, as discussed below.

*Metric graph reconstruction.* Aanjaneya et al. (2012) make use of local detection techniques based on connected components to classify edges and non-edges in metric graphs constructed from point cloud data. Vandaele et al. (2019a) extend this by also classifying the type of non-edge (a leaf, a bifurcation, trifurcation, ...), as well as identifying locations through which cycles pass. Global reconstruction techniques are used to retrieve the underlying topology from this information. Unlike persistent homology, they locally infer connected components in a punctuated neighborhood at a *fixed* scale $\epsilon$. They require one single Rips graph $\mathcal{R}_\epsilon$ to be (locally) built from the data. Hence, they do not track the the evolution across various scales. This induces a high parameter sensitivity, and the methods quickly fail in more complex or noisy examples. For this reason, they are generally not applicable to $k$-nearest neighbor ($k$NN) graphs—which turn out to favorable in many practical cases where the data is characterized by varying scales or a non-uniform density across its underlying topology (Von Luxburg and Alamgir, 2013)—or given non-metric graphs.

*Local topological persistence.* Fasy and Wang (2016) and Wang et al. (2011) provide a more general tool for investigating local structure in—not necessarily graph-structured—data, by refining topological persistence to qualitatively investigate the underlying topology

in a (punctuated) local neighborhood of a data point. Unlike the methods for metric graph reconstruction, they do track the evolution of topological features locally at various scales. However, the sensitivity to outliers remains, and this approach lacks a method to effectively use this type of local information for automating the inference or reconstruction of local and global topologies. Furthermore, computing topological persistence for many data points is often computational inefficient for dense and large data sets.

*Mapper.* Nicolau et al. (2011) present the *Mapper* algorithm, providing a general tool for visualizing point cloud data. First, the data is mapped to a low-dimensional space, usually by means of a dimensionality reduction (such as PCA) to $\mathbb{R}$ or $\mathbb{R}^2$. A grid of overlapping cells is built in the low-dimensional space, and for each cell, the points mapped to this cell are clustered in the original space. Overlapping clusters are then connected, leading to a graph visualization of the underlying topology of the data. Unfortunately, the Mapper algorithm is quite sensitive to the used parameters, such as the type of filter, the amount of overlap of cells, and the clustering method in the original space. Furthermore, Vandaele et al. (2019a) showed that Mapper may fail to retrieve simple underlying (such as Y-structured) topologies in cell trajectory data sets characterized by noise and non-uniform densities.

## 2. Methods

A schematic overview of our method for topological data analysis of graph-structured data is shown in Figure 1e. The organization of our Methods section is based on this overview, and is as follows. In Section 2.1, we first discuss the *boundary coefficient* (BC), a local vertex measure designed to identify core nodes in weighted graphs, introduced by Vandaele et al. (2019b). In Section 2.2, we discuss *$f$-pines*, also introduced by Vandaele et al. (2019b). We illustrate how these may be used to obtain a forest representation of a graph. More specifically, Letting $f = \text{BC}$ will lead to effective representations for topological data analysis of graphs. In Section 2.3, we introduce the novel *CLOF*-problem, as well as an algorithm to efficiently solve it in tree and forest graphs. Finally, in Section 2.4, we discuss how all of the above fits together and forms our newly introduced method for topological data analysis of graph-structured data (Figure 1e).

### 2.1 The Boundary Coefficient: a Powerful Core Measure in Graphs

The first step of our method requires us to locate 'core nodes' in our graph. Intuitively, these are the nodes that lie close to the backbone of our graph, i.e., its underlying simplified graph-structured topology (Figure 1). As we made clear in Section 1.2, many existing measures that might be used to determine the core nodes of a graph lack important properties required for identifying such nodes in many practical weighted graphs (see also Figure 3). Hence, in Section 2.1.2 we present our recently introduced *boundary coefficient* (BC), defined as the negative average *transmissivity* (Section 2.1.1) of a node (Vandaele et al., 2019b). We discuss important properties of the BC, as well as its relationship to the ordinary LCC (Section 2.1.3). Finally, we present a way to efficiently compute the BC through (sparse) matrix multiplication in Section 2.1.4.

### 2.1.1 The Transmissivity of a Node

Given two vectors $\mathbf{x}, \mathbf{y}$ in the Euclidean space $\mathbb{R}^n, n \in \mathbb{N}^*$, we know that the angle $\alpha$ between them satisfies

$$\cos\alpha = \frac{\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - \|\mathbf{x}-\mathbf{y}\|^2}{2\|\mathbf{x}\|\|\mathbf{y}\|} \ .$$

As all of the terms in the fraction are expressed as (Euclidean) distances (between pairs of the triple of vectors $(\mathbf{x}, \mathbf{y}, \mathbf{0})$), we can straightforwardly generalize the concept of angle to arbitrary metric spaces $(M, d)$. Furthermore, a positively weighted graph $G = (V, E)$ can be converted to a metric space $(V, d)$, where for $u, v \in V$, $d(u, v)$ denotes the length of the shortest (weighted) path from $u$ to $v$ in $G$. This extends the definition of angle in Euclidean spaces to graphs as well (Vandaele et al., 2019b).

**Definition 1** *Let $G = (V, E)$ be an undirected, positively weighted graph. Suppose that $u, v, w \in V, u \neq v \neq w$, belong to the same connected component of $V$. We define the (cosine of the)* angle $\widehat{uvw}$ *as*

$$\cos\widehat{uvw} \coloneqq \left(\frac{d(u,v)^2 + d(v,w)^2 - d(u,w)^2}{2d(u,v)d(v,w)}\right) \ ,$$

*where $d$ denotes the pairwise shortest distance metric on $G$. The* transmissivity $\mathcal{T}(u, v, w)$ *of $v$ for $u$ and $w$ is defined as*

$$\mathcal{T}(u, v, w) \coloneqq -\cos\widehat{uvw} \ .$$

The transmissivity $\mathcal{T}(u, v, w)$ of $v$ for $u$ and $w$ has a meaningful interpretation even when the graph is not embedded in a Euclidean space. $\mathcal{T}(u, v, w)$ will be high if the cost of going first straight from $u$ to $v$, and then straight from $v$ to $w$, does not differ a lot from the cost of going straight from $u$ to $w$. Here, by going straight we mean taking the shortest path, and hence, by the cost the weighted length of this path, i.e., the sum of the weights of its included edges. Moreover, if going through $v$ is the only possibility to go from $u$ to $w$, then $\mathcal{T}(u, v, w) = 1$ (note that the reverse implication does not necessarily hold). Vice versa, $\mathcal{T}(u, v, w)$ will be low if it is much more costly to travel from $u$ to $w$ through $v$, than to go straight from $u$ to $w$, and exactly $-1$ if $u = w$.

Furthermore, it is important to note that the graph $G = (V, E)$ must not be metric, i.e., the weights $\omega$ do not have to satisfy the *triangle inequality* in $G$. This means we may have $\omega(\{u, v\}) + \omega(\{v, w\}) < \omega(\{u, w\})$ for $\{u, v\}, \{v, w\}, \{u, w\} \in E$. The shortest path metric $d$ will always naturally satisfy the triangle inequality, which is needed to generalize the Euclidean angle to graphs.

### 2.1.2 The Boundary Coefficient as the Average Transmissivity

The *boundary coefficient* (BC) of a node $v$ is defined as its negative transmissivity averaged over the pairs of neighbors of $v$ (Vandaele et al., 2019b). As illustrated by Fig. 5 and Fig. 6, this is a measure for how close vertices are near the 'boundary' of the graph (hence the name), and by this, whether the nodes are close or far from the graph's core.

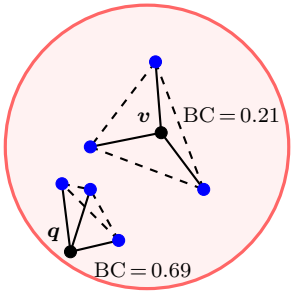



Figure 5: Geometric interpretation of the boundary coefficient: a point $v$ lying further from the boundary has many more pairs of neighbors defining a large angle, than a point $q$ lying close to the boundary. The dashed line represents the shortest path — not necessarily an edge — between two nodes. The boundary coefficients are computed using only the drawn connections and their Euclidean lengths.

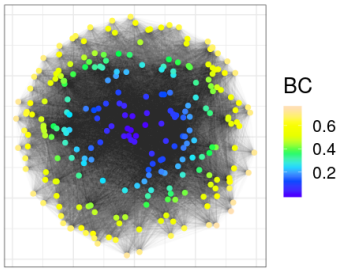
**Definition 2** *Let $G = (V, E)$ be an undirected, positively weighted graph, without selfloops. For every $v \in V$ we define $\mathcal{N}(v) \subseteq V$ to be the set of neighbors of $v$ in $G$. For every $v \in V$ with degree $\delta(v) = |\mathcal{N}(v)| > 0$, we define its* boundary coefficient *(BC) as*

$$BC(v) := \frac{-1}{\delta(v)^2} \sum_{u,w \in \mathcal{N}(v)} \mathcal{T}(u, v, w) \; .$$

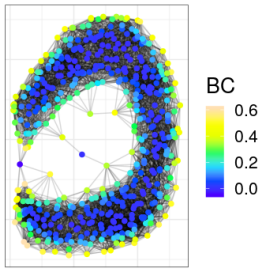
### 2.1.3 Properties of the Boundary Coefficient

As is the case with the ordinary LCC, for a graph $G = (V, E)$, the BC of a vertex $v \in V$ is an averaged value over triples adjacent to $v$. In the case of the LCC, the assignment to each triple $(u, v, w)$ is a 'hard' 0-1 assignment. In the unweighted case, i.e., where each edge has weight 1, the assigned value to the triple $(u, v, w)$ in the averaged sum of BC$(v)$ equals

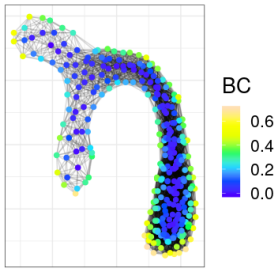
$$-\mathcal{T}(u, v, w) = \cos \widehat{uvw} = \begin{cases} \frac{1}{2} & \text{if } \{u, w\} \in E \; , \\ -1 & \text{if } \{u, w\} \notin E \wedge u \neq w \; , \\ 1 & \text{if } u = w \; . \end{cases} \tag{1}$$




(a) The boundary coefficients for a graph with an underlying disk-shaped topology.




(b) The boundary coefficients for a graph with an underlying C-shaped topology.




(c) The boundary coefficients for a graph with an underlying Y-shaped topology.

Figure 6: The boundary coefficients for each one of the graphs in Figure 3. The BC can handle fully weighted networks, curvature, as well and outliers, which are separated from the major core through boundary nodes.

The intuition behind this is as follows. Suppose for $\{u,v\},\{v,w\} \in E$, that $(u,v,w)$ forms a closed triangle adjacent to $v$, i.e., $\{u,w\} \in E$. Since the graph is unweighted, each edge of this triangle gets assigned the same distance. Hence, the triple $(u,v,w)$ is regarded as an equilateral triangle, which has all angles equal to $60°$. This coincides that with the fact that $\mathcal{T}(u,v,w) = -\frac{1}{2} = -\cos 60°$. If $\{u,w\} \notin E$, we regard the triplet $(u,v,w)$ as a straight line segment, defining a $180°$ angle in $v$. Again, this coincides with the fact $\mathcal{T}(u,v,w) = 1 = -\cos 180°$. In this case, there may be other shortest paths from $u$ to $w$ in $G$, but $u \to v \to w$ is definitely one of them. If $u = w$, we regard the triple as two coinciding line segments defining a $0°$ angle in $v$. In this case, we find that $\mathcal{T}(u,v,w) = -1 = -\cos 0°$.

The explicit relationship between the BC and LCC is as follows (Vandaele et al., 2019b).

**Proposition 3** *Suppose $G = (V,E)$ is an unweighted graph, i.e., a graph in which every edge gets a weight equal to 1, without selfloops. Then for every $v \in V$ with $\delta(v) > 1$*

$$BC(v) = \frac{\delta(v)-1}{\delta(v)}\left(\frac{3}{2}LCC(v) - 1\right) + \frac{1}{\delta(v)}\,.$$

**Proof** See Appendix D. ■

**Corollary 4** *Suppose $G = (V,E)$ is an unweighted graph without selfloops. Then for every $v \in V$, $\lim_{\delta(v)\to\infty} BC(v) = \frac{3}{2}LCC(v) - 1$.*

**Proof** This is an immediate consequence of Proposition 3. ■

Proposition 3 implies that the BC does not fulfill the *general versatility* requirement, i.e., it does not coincide with the ordinary LCC on unweighted graphs, as other generalizations of the LCC to weighted graphs do (Wang et al., 2017). However, the BC does appear to be closely related to the LCC: it is nearly an affine transformation of the LCC (as given in Corollary 4). In Section 2.2, we show that such transformations result in the same forest representation through $f$-pines (Proposition 12).

The fact that the relationship between the BC and the LCC is not an exact affine transformation, is due to us allowing BC to be well-defined for nodes $v$ with $\delta(v) = 1$, i.e., allowing $u = w$ in the summation over triples $(u,v,w)$ adjacent to $v$. $\mathrm{BC}(v) = \cos\widehat{wvw} = 1$ for these nodes, which coincides with our idea of nodes with a high boundary coefficient lying at the boundary of the graph. Any path that enters a node $v$ with $\delta(v) = 1$ from a node $w$ and wishes to continue, has no choice than to take a $180°$ turn back to $w$. Intuitively, the path has reached a 'dead end' in $v$, and hence, reached the boundary of the graph. Furthermore, if $F$ is a spanning forest of $G$ (Definition 9), then a *leaf* of $G$, i.e., a node $v \in V$ for which $\delta(v) = 1$, will always be a leaf of $F$ as well. In Section 2.2, we will use the BC to obtain a particular spanning forest, in which leaves are exactly meant to represent boundary nodes of $G$, i.e., for which the coefficient is high. Hence, for our method, it makes sense that the BC is both well-defined at leaves, and obtains its maximal value there.

As is the case for the ordinary LCC, $\mathrm{BC}(v)$ is undefined for nodes $v$ with $\delta(v) = 0$.

**Remark 5** *The crucial differences between the BC, the LCC and many of its generalizations (Wang et al., 2017), and standard global centrality measures such as eccentricity (Hage and Harary, 1995) or betweenness (Barthélemy, 2004), as well as the main reasons why the BC outperforms these measures for a wide variety of applications, are that for a node $v \in V$:*

- *The assignment $-\mathcal{T}(u, v, w)$ to a triple $(u, v, w)$ in the sum of BC(v) may attain different values over triples where $\{u, w\} \notin E$ (i.e., it is not always 0, such as with LCC, Onella's generalized LCC, ...).*
- *The assignment $-\mathcal{T}(u, v, w)$ to a triple $(u, v, w)$ in the sum of BC(v) may be low even if $\{u, w\} \in E$ and—if weighted—the three corresponding weights are relatively high (this is not the case with LCC, Onella's generalized LCC, ...).*
- *The scope of the BC is local: it does not take into account the shortest paths to all other nodes (as is often the case with standard centrality measures, such as betweenness). Hence, the BC allows us to locate boundary nodes even in the presence of complex, long, or curving underlying topologies, and is less affected by outliers.*

Though the BC does not coincide with the ordinary LCC on unweighted graphs, it does satisfy four other essential properties of generalizations of the LCC to weighted graphs, as discussed in Wang et al. (2017). One of these, i.e, its *applicability to fully weighted networks*, has been illustrated in Figure 6. We consider this property of the boundary coefficient, as well as its *weight-scale invariance*, *continuity*, and *robustness to noise* (Wang et al., 2017), far more important for our purpose of identifying core structure in a wide variety of weighted graphs, than coinciding with the LCC on unweighted graphs. However, they will be less important for explaining our method, and were (partially) discussed by Vandaele et al. (2019b). Hence, we state and prove these properties formally in Appendix D.

#### 2.1.4 Expressing the Boundary Coefficient through Matrix Operations

In this section, we give an important result for computing the BC. First, we introduce some new definitions.

**Definition 6** *Let $G = (V, E)$ be an undirected, positively weighted graph, and $D$ the matrix of pairwise shortest path distances between the nodes of $G$. Let $|E(P)|$ denote the unweighted length of a path $P$. For $k \in \mathbb{N}$, we define the* hop-$k$-approximation *of $D$ as the matrix $\mathcal{H}_k(D) = (\mathcal{H}_k(D)_{u,v})_{u,v \in V}$, where*

$$\mathcal{H}_k(D)_{u,v} \coloneqq \begin{cases} D_{u,v} & \text{if there exists a path } P \text{ from } u \text{ to } v \text{ with } |E(P)| \leq k\,, \\ 0 & \text{otherwise}\,, \end{cases}$$

It is easy to see that increasing $k$ leads to a better approximation of $D$ (hence the name 'hop-$k$-approximation'). A formal proof of this is given in Appendix D (Proposition D.11).

For a disconnected graph $G$, $\mathcal{H}_k(D)$ will never be equal to $D$ for any $k \in \mathbb{N}$. $D_{u,v}$ is either undefined or defined to be $+\infty$ if $u$ and $v$ lie in different connected components of $G$. However, $\mathcal{H}_k(D)_{u,v}$ is always well-defined, and will be equal to 0 if there is no path between $u$ and $v$. This will show to be convenient for computing the BC (Theorem 8).

**Notation 7** *For any $z \in \mathbb{Z}$, we define the mapping*

$$\cdot^{\odot^z} : \bigcup_{n,m\in\mathbb{N}} \mathbb{R}^{n\times m} \to \bigcup_{n,m\in\mathbb{N}} \mathbb{R}^{n\times m} : A \mapsto A^{\odot^z}, \text{ with } A^{\odot^z}_{u,v} \coloneqq \begin{cases} A^z_{u,v} & \text{if } z \geq 0 \vee A_{u,v} \neq 0\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

*Hence, $\cdot^{\odot^z}$ denotes the pointwise application of the $\cdot^z$-operation on the elements of a given matrix for which this is well-defined. If $A^z_{u,v}$ would not be well-defined (i.e., if $z < 0$ and $A^z_{u,v} = 0$), then this entry gets mapped to 0. For $G = (V, E)$ an undirected, positively weighted graph with pairwise distance matrix $D$, and $k \in \mathbb{N}$, we define $\mathcal{H}_k^{\odot^z}(D) \coloneqq \mathcal{H}_k(D)^{\odot^z}$.*

**Theorem 8** *(Vandaele et al., 2019b). Let $G = (V = \{v_1, \ldots, v_n\}, E)$ be an undirected, positively weighted graph, without selfloops. Let $D$ denote the matrix of pairwise shortest path distances between the nodes of $G$. If $\delta(v) > 0$ for all $v \in V$, then*

$$\begin{pmatrix} BC(v_1) \\ \vdots \\ BC(v_n) \end{pmatrix} = \begin{pmatrix} \frac{1}{\delta(v_1)^2} \\ \vdots \\ \frac{1}{\delta(v_n)^2} \end{pmatrix} \odot \left[ \left( \sum_{u\in V} \mathcal{H}_1(D)_u \right) \odot \left( \sum_{u\in V} \mathcal{H}_1^{\odot^{-1}}(D)_u \right) - \frac{1}{2} diag\left( \mathcal{H}_1^{\odot^{-1}}(D) \mathcal{H}_2^{\odot^2}(D) \mathcal{H}_1^{\odot^{-1}}(D) \right) \right], \tag{2}$$

*where $A \odot B$ denotes the pointwise multiplication between matrices $A$ and $B$ of the same dimensions.*

**Proof** See Appendix D. ■

It follows that the BC may be computed using pairwise Dijkstra's algorithm with early termination, and (sparse) matrix multiplications. A computational analysis of the algorithm that follows from Theorem 8 is provided in Appendix B. Note that both the left hand side and right hand side in (2) are undefined for nodes $v \in V$ with $\delta(v) = 0$. We regard such nodes simultaneously as boundary nodes, as well as core nodes within their own component.

We conclude that the BC is a powerful measure for locating nodes near the backbone of a graph, while admitting an efficient way for computation. In the next section, we show how the BC leads to effective forest presentations for topological data analysis of graphs.

### 2.2 Forest Representations of Graphs through f-Pines

In order to overcome the wide variety of issues accompanied with ordinary facility location problems in graphs for our purpose of locating simplified topological subtructures (Section 1.2), we propose an intermediate step that represents the given graph $G$ by means of a spanning forest of $G$. This step will use our recently introduced concept of the *f-pine* of a graph (Vandaele et al., 2019b), which we present in Section 2.2.1. We will discuss a variety of its properties in Section 2.2.2. Finally, in Section 2.2.3 we illustrate how the boundary coefficient can be used to find a forest representation, from which we may efficiently mine simplified topological structures.

### 2.2.1 The f-Pine of a Graph

Given a graph $G = (V, E)$ and a real-valued function $f : V \to \mathbb{R}$, we want to find a spanning forest with leaves marking higher values of $f$ (Vandaele et al., 2019b).

**Definition 9** *Let $G = (V, E)$ be a graph. A spanning forest $F$ of $G$ is a subgraph of $G$, such that each connected component of $G$ is also a connected component of $F$ in terms of its contained vertices, and $F$ contains no cycles.*

**Definition 10** *Let $G = (V, E)$ be a graph, and $f : V \to \mathbb{R}$. A spanning forest $F$ of $G$ is called an $f$-pine*[1] *in $G$, if*

$$F \in \arg\min \left\{ \sum_{v \in V} \delta_{F'}(v) f(v) : F' \textit{is a spanning forest of } G \right\} , \tag{3}$$

*where $\delta_{F'}(v)$ denotes the degree of $v$ in the subgraph $F'$ of $G$.*

The intuition behind the naming is that an $f$-pine corresponds to trees having many 'needles' that 'stick out and point' towards (locally) high values of $f$ (Figure 1c).

### 2.2.2 Properties of the $f$-Pine

Looking at Definition 10, an $f$-pine of a graph $G$ is a spanning forest of $G$ that prefers high-degree nodes where $f$ attains a low value. More specifically, an $f$-pine attaches every node $u$ to a node $v$ where $f$ reaches a local minimum (Vandaele et al., 2019b).

**Proposition 11** *Let $G = (V, E)$ be a graph, $f : V \to \mathbb{R}$, and $F$ an $f$-pine in $G$. For every $u \in V$ with $\delta_G(u) > 0$, there exists $v \in \arg\min\{f(w) : w \in \mathcal{N}_G(u)\}$ such that $\{u, v\} \in E(F)$.*

**Proof** See Appendix D. ■

The intuition behind the proposition above is that the building blocks of an $f$-pine are several large star graphs that result from pulling every node towards a node where $f$ attains a local minimum. Furthermore, Definition 10 implies that the centers of these star graphs will be connected through nodes where $f$ attains a low value on average as well.

It turns out that an $f$-pine is invariant to affine transformations of $f$ with a positive scaling factor. Hence, we may apply such transformation to $f$—retaining its robustness properties—without effecting the resulting $f$-pine. Furthermore, we may also easily compare $f$-pines for different functions $f$ (e.g., graph centrality, core, or transitivity measures), even if the corresponding function values take on a different scale (Vandaele et al., 2019b).

**Proposition 12** *(Vandaele et al., 2019b). Let $G = (V, E)$ be a graph, $f : V \to \mathbb{R}$, and $F$ an $f$-pine in $G$. If $g = af + b$ for some $a \in \mathbb{R}^+, b \in \mathbb{R}$, $F$ is also a $g$-pine in $G$.*

1. The term 'pine' may not be ideal if $G$ is disconnected, as there will be multiple 'pines'. In this case, the term 'vine' may be more appropriate. However, the main emphasize of the term 'pine' is on 'many leaves' (needles) and 'few branches', and not on the number of components.

**Proof** See Appendix D. ■

We can efficiently find an $f$-pine by finding a minimum spanning tree after reweighing the edges in $G$ with the summed value $f$ attains at their endpoints (Appendix B).

**Proposition 13** *(Vandaele et al., 2019b). Let $G = (V, E)$ be a graph, and $f : V \to \mathbb{R}$. Finding an $f$-pine in $G$ is equivalent to finding a minimum spanning tree for each connected component in $G$, where each edge $\{u, v\}$ is assigned to have weight $f(u) + f(v)$.*

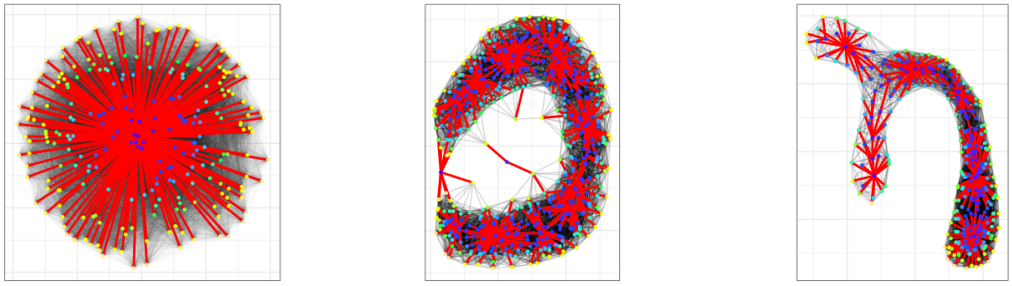
**Proof** See Appendix D. ■

We are now prepared to show how the BC (Section 2.1) and $f$-pines work together to provide effective forest representations for topological data analysis of graphs.

### 2.2.3 The BC-pine of a Graph

Proposition 11 states that an $f$-pine connects nodes to its local minima. Hence, if $f$ is a 'core' measure identifying nodes close to the underlying core structure of the graph, then an $f$-pine is the result of interconnecting star graphs through the core of the given graph. This is the exact purpose for which we designed the boundary coefficient, taking on low values near the core structure of a graph, and high values near the boundary nodes of the graph. Three example BC-pines are illustrated in Figure 7.

An important property of the BC-pine is displayed on Figure 7b. As the BC is able to separate outliers from the main core structure through boundary nodes where the BC attains locally higher values by design, the BC-pine avoids passing through outliers to reach one (true) core region from another. This would increase the cost of the pine according to (3), as it would include too many boundary nodes on either side of the outlier node.



(a) The BC-pine for a graph with an underlying disk-shaped topology.

(b) The BC-pine for a graph with an underlying C-shaped topology.

(c) The BC-pine for a graph with an underlying Y-shaped topology.

Figure 7: The BC-pines (with edges in red) for the three example graphs shown in Figure 6. Note that by Proposition 11, a BC-pine will result in a star graph that connects all nodes to a global minimum in a complete graph, such as for the graph on the left.

Vandaele et al. (2019b) showed that by iteratively 'pruning' the BC-pine, i.e., discarding its leaves, we retract our pine towards the topological core underlying the graph. However, if the original graph has a very complex or even no ground-truth underlying topology, revealing a low complexity topology by means of this 'top-down' method may be difficult, discarding (too) many nodes. Furthermore, even in graphs with a simple underlying topology, the presence of outliers may require us to prune many times to remove these connections (Figure 7b), resulting in our backbone retracting from the true underlying leaves as well. Hence, we are clearly in need of an approach that allows us to identify interesting substructures in forest graphs, spreading out across the entire graph, while the resulting complexity remains easy to tune and control. This leads us to the following section, introducing a 'bottom-up' method for identifying our backbone.

### 2.3 Finding Optimal Subforests with a Constrained Number of Leaves

In this section, we will introduce a novel theoretically founded and well-posed problem for the purpose of locating interesting substructures in forest graphs. We will show that solving this problem in the BC-pine leads to an effective method for topological inference in graphs.

We will consider two variants of this problem, one where the cost of the structure will be determined by an edge-valued function, and one where its cost will be determined by a vertex-valued function.

**Definition 14** *(CLOF). Let $G = (V, E)$ be a graph, and suppose $f$ is a real-valued function, associating a positive cost to either each vertex or each edge of $G$. For a subgraph $H = (V(H), E(H))$ of $G$, we define its cost $f(H) \coloneqq \sum_{\alpha \in Dom(f) \cap V(H) \cap E(H)} f(\alpha)$. The* **C**onstrained **L**eaves **O**ptimal sub**F**orest *problem (CLOF) is stated as follows.*

$$\textit{Given } k \in \mathbb{N}_{\geq 2}, \textit{ find a subforest } F \textit{ in } G \textit{ with at most } k \textit{ leaves, maximizing } f(F)\,. \quad (4)$$

**Proposition 15** *CLOF is NP-hard.*

**Proof** For an edge-valued cost function and $k = 2$ leaves, the stated problem is equivalent to the NP-hard longest path problem (Uehara and Uno, 2005). ■

Though CLOF is NP-hard in general, we are actually not interested in its solution for arbitrary graphs. As discussed in Section 1.2, such solutions may just not be topological meaningful. This is one of the main reasons we choose for a forest representation as an intermediate step. In this way, we can design our forest such that these solutions are indeed meaningful as well as robust. Furthermore, as we will show in this section, CLOF is efficiently solvable for forest graphs in practice.

In Section 2.3.1 we will derive an efficient solution to CLOF for tree graphs, which will lead us to an efficient solution for forest graphs in Section 2.3.2.

#### 2.3.1 Solving CLOF in Tree Graphs

We start by showing that for tree graphs, CLOF is equivalent to a monotone submodular set function maximization problem subject to a cardinality constraint.

**Theorem 16** *Let $T = (V, E)$ be a tree graph and $f$ a real-valued function associating a positive cost to either each vertex or each edge of $T$. Then (4) is equivalent to a monotone submodular set function maximization problem subject to a cardinality constraint (Krause and Golovin, 2011).*

**Proof** See Appendix D. ■

The general problem of maximizing a monotone submodular function subject to a cardinality constraint is NP-hard, but admits a $1 - 1/e$ approximation algorithm (Krause and Golovin, 2011). However, the interestingness of Theorem 16 lies in the fact that (4) is equivalent to a nontrivial monotone submodular set function maximization problem, for which we are able to actually provide an exact solution in polynomial time.

**Theorem 17** *(A greedy solution for CLOF). Let $T = (V, E)$ be a tree graph and $f$ a real-valued function associating a positive cost to either each vertex or each edge of $T$. Given $k \in \mathbb{N}_{\geq 2}$, the following algorithm finds a subtree $T'$ in $T$ that maximizes $f(T')$ over all subtrees $T'$ in $T$ with at most $k$ leaves.*

1. *Let $T'$ be the longest path (according to $f$) between two leaves in $T$.*
2. *While $T' \neq T$ and $T'$ has less than $k$ leaves, add the longest path (according to $f$) from the remaining leaves of $T$ to $T'$.*

**Proof** See Appendix D. ■

**Remark 18** *Due to a greedy algorithm resulting in the optimal subtree according to (4) for tree graphs, we only need to conduct the algorithm once to get all solutions up to the given value $k \in \mathbb{N}_{\geq 2}$. By storing the included vertices or edges for each iteration, we can quickly obtain the corresponding subgraph and analyze the results for different number of leaves (see also Algorithm 3 in Appendix B for the pseudocode of the corresponding algorithm). Storing the cost up to a certain number of leaves turns out to be useful in practice as well. It may serve as a tool for tuning the number of leaves, as we will discuss in Section 2.4.*

**Remark 19** *Theorem 17 implies that a greedy approach leads to the optimal solution in any tree (including spanning trees). Hence, a heuristic search such as a* beam search *(Ow and Morton, 1988) will not be necessary to optimize (4) for topological data analysis of graphs (Section 2.4).*

The following result shows to be very useful for practical applications, as we will discuss in Section 2.4. First, we need another definition.

**Definition 20** *Let $G = (V, E)$ be a tree graph, and suppose $f$ is a real-valued function, associating a cost to either each vertex or each edge of $G$. We say that $f$ is* constant on leaves *of $G$, if either $f$ is constant on $\{\{u, v\} \in E : \delta(u) = 1 \vee \delta(v) = 1\}$ if $f$ is edge-valued, or $f$ is constant on $\{v \in V : \delta(v) = 1\}$ if $f$ is vertex-valued.*

**Theorem 21** *Let $T = (V, E)$ be a tree graph with $|E| > 1$, and suppose $f$ is a real-valued function, associating a positive cost to either each vertex or each edge of $T$. If $f$ is constant on leaves of $T$, then a solution to (4) for the subgraph $T'$ of $T$ that results from discarding all leaves of $T$, i.e., by* pruning *$T$, can be converted to a solution to (4) for $T$ in linear time.*

**Proof** See Appendix D. ■

#### 2.3.2 Solving CLOF in Forest Graphs

The main problem for solving (4) for forest graphs is that the greedy approach described in Theorem 17, will not work for forest graphs. E.g., consider the union of a linear graph $L$ and a bifurcating tree $T$ (Figure 8). Suppose $f$ is an edge-valued cost function such that $f(L) = 10$, and $f(B) = 4$ for each of the three branches $B$ connecting the bifurcation point to a leaf in $T$. The longest path (according to $f$) in the union of these graphs is $L$. However the maximal subforest with at most 3 leaves is $T$, which does not contain $L$.

Nevertheless, we are able to straightforwardly apply the algorithm solving (4) for tree graphs, to each separate connected component of the forest. From this, a solution for forest graphs is easily derived. We discuss this more formally in Appendix B.

We conclude that CLOF is a graph optimization problem of high theoretical interest. It induces a nontrivial monotone submodular set function maximization problem that can be efficiently solved for forest graphs. Nevertheless, its practical value remains unclear until this point. It turns out that CLOF provides an effective way for inferring backbones in graphs through forest presentations. Hence, in the next section, we finally bring together the boundary coefficient, $f$-pines, and CLOF, for topological data analysis of graphs.

### 2.4 f-Pines for Topological Data Analysis of Graph-Structured Data

We are now fully prepared to present our new method for topological data analysis (TDA) of graph-structured data, the schematic overview of which is given in Figure 1e.

1. In case of point cloud data, we need to represent the underlying topology through a graph. This is usually done by means of a well-known proximity graph, such as the Rips graph or $k$-nearest neighbor ($k$NN) graph.

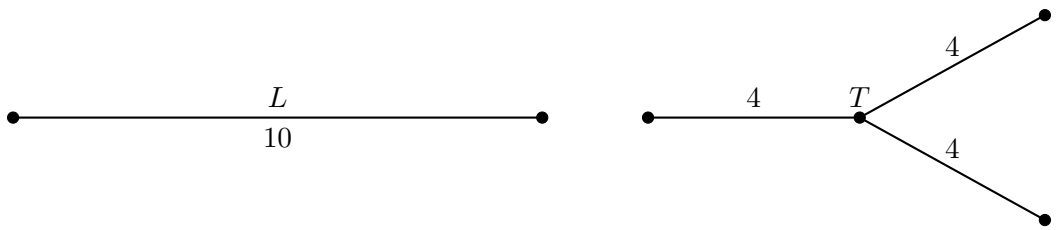



Figure 8: A greedy approach will always start from the path $L$, which cannot be extended to a forest containing three leaves. However, the optimal subforest with three leaves is $T$.

2. Based on the core measure $f$, we build an $f$-pine $F$ in $G$ (Definition 10) using the minimum spanning tree algorithm (Proposition 13). For weighted graphs where each weight represents a notion of distance between vertices, i.e., higher weights denote more distant nodes, we use the boundary coefficient (BC) as core measure (Definition 2). Its superior effectiveness over existing core measure for locating core structure near the underlying backbone of a graph, has been discussed in Sections 1.2 & 2.1, and qualitatively and quantitatively demonstrated by Vandaele et al. (2019b). For unweighted graphs, we use the ordinary local cluster coefficient (LCC), due to its close relation to the boundary coefficient (Proposition 3, Corollary 4 & Proposition 12), and the extensive amount of research that has already been performed on both its theoretical and computational aspects (Watts and Strogatz, 1998; Zhu et al., 2017).

3. We solve CLOF for a well-chosen cost function $g$ on either the vertices or edges in $F$ (Section 2.4.1). We solve (4) for either a given number of leaves $k \in \mathbb{N}_{\geq 2}$ of interest, or a (possibly infinite) upper bound on the expected number of leaves.

4. Further analysis may be required to result in the final backbone topology. E.g., in the case of an unknown number of leaves where an upper bound was provided, visual inspection or an elbow locating method may be used to infer the number of leaves, which we discuss in Section 2.4.2 (see also Remark 18). A further step may be required to identify cycles that are missing a representation in the forest-structured backbone, which we discuss in Section 2.4.3.

### 2.4.1 Introducing Cost Functions for CLOF

To effectively mine topological substructures through CLOF in forest representations, we need a function $g$ defined on either the vertices or edges of the representations to optimize in terms of (4). Some interesting choices for the $g$ are discussed in Appendix C, which also demonstrates the usefulness of Theorem 21 on a practical example. However, we will focus on *vertex betweenness*, which equals how many shortest paths go through a particular node.

The intuition for using this measure is as follows. By Proposition 11, many (boundary) nodes are not located on the backbone, but attached to it by means of a local minimum. To reach one (boundary) node from another, we must first travel to the backbone, then from one location on the backbone to another location, and thereafter leave the backbone to connect to the other node. Hence, nodes on the backbone represent important nodes through which

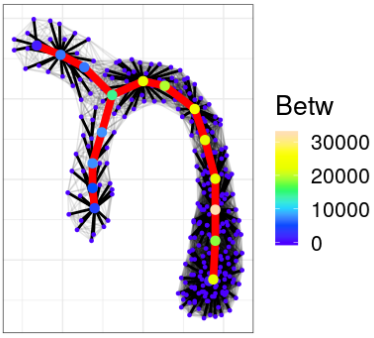



Figure 9: Optimal subgraph with 3 leaves in a pruned BC-pine according to the vertex betweenness (edges in red). Nodes with high betweenness represent important nodes for accessing many others, due to the uniqueness of paths between nodes in a forest.

a lot of 'traffic' flows in the pine. The only possibility to reach distant locations from each other is to pass through these, resulting in nodes with a high betweenness.

The effectiveness of using vertex betweenness for TDA of graph-structured topologies through forest representations and CLOF rests on the following properties.

- Though the betweenness is tedious to compute in large graphs, it becomes a lot easier in forest graphs. The reason for this is that there is one unique path between each pair of nodes lying within the same connected component. This also implies that the forest can be treated as an unweighted graph for computing the vertex betweenness.

- Not only is the vertex betweenness constant on leaves (Definition 20), making it a lot more efficient to solve CLOF by prepruning the forest representation (Theorems 21 & B.4), it also equals 0 on leaves. This implies that a solution to (4) in the pruned forest representation equals a solution to (4) in the original representation.

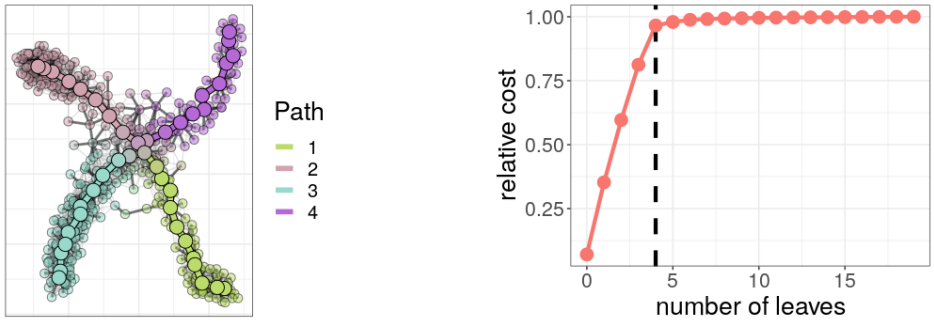
- Contrary to other interesting vertex/edge-valued functions (Appendix C), the vertex betweenness also accounts for the global topological structure of the forest. Hence, given an effective forest representation, the relation between the vertex betweenness and the backbone substructure goes in both directions. This means that nodes with a high vertex betweenness correspond to nodes inducing the backbone substructure in the forest representation, and vice versa (Figure 9).

We emphasize that the same measure, in this case the betweenness centrality, might only be come topologically meaningful for identifying backbone structures in graphs through a forest representation. This can be seen by comparing Figures 3c &. 9 (see also Remark 5).

### 2.4.2 Estimating the Number of Leaves

Solving CLOF requires one parameter as input, namely the number $k$ of leaves to be included in the backbone. For the purpose of topological data analysis of graphs, this parameter is ideally inferred from the data itself. As discussed in Remark 18, the fact that CLOF can be solved through a greedy algorithm admits a convenient way to perform this estimate. We will consider the case where our original graph $G$ is connected, i.e., the resulting pine is a tree graph $T$. In case of disconnected graphs, our current approach is to estimate the number of leaves for each component separately.

As the cost of the subtree increases with each iteration of the algorithm described in Theorem 17, we can track the increase in cost according to the added number of leaves. Initially, the increase in cost is high when we add true branches to our current subtree. When all true branches are added, we start connecting our subtree to surrounding noise or outliers, and the increase in cost drops. This is illustrated in Figure 10, which also shows that an 'elbow' inference method may be used to tune the number of leaves. This may be done either visually, or by a using an automated procedure, such as minimizing the second-order finite differences of the function shown in Figure 10b. Note that we can easily extract any solution with fewer leaves from the current solution (Remark 18)




(a) Optimal subgraph with 4 leaves in a (pruned) BC-pine using vertex betweenness. Edges and nodes are colored according to their closeness to 1 of the 4 branches.

(b) Tracking the relative cost of the subtree, i.e., the cost of the subtree divided by the cost of the pine, displays an 'elbow' at $k = 4$ leaves.

Figure 10: Extracting the optimal result to (4) for a tuned number of leaves $k = 4$ in a BC-pine using vertex betweenness as cost. The pine was obtained for a Rips graph ($\epsilon = 8$) on a 2D point cloud data set with an underlying X-shaped topology.

### 2.4.3 Identifying Missing Cycles

Though until now we have thoroughly illustrated the advantages of working with intermediate forest representations of graphs for mining topological substructures, there is also a disadvantage of using such representations at first sight. As by Definition 9 a spanning forest may never include a cycle, we are unable to use any of its subgraphs for representing the underlying topology of a graph if the true underlying model contains cycles.

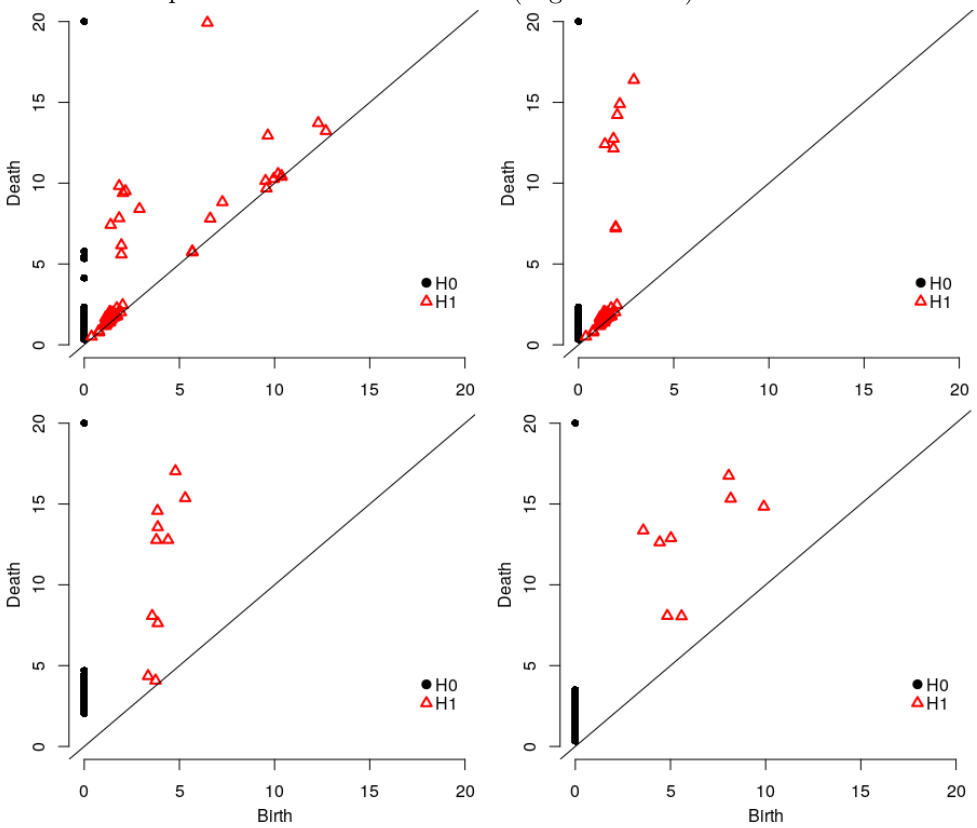
However, as we will illustrate in this section, identifying a simplified underlying forest-structured topology can be highly beneficial for identifying cycles missing in the backbone representation of the underlying topology. We emphasize that though the approach discussed in this section is still experimental, it leads to effective results in practice.

Consider the forest-structured backbone $B$ we constructed throughout a point cloud data set $D$ of 807 observations representing Pikachu in the Euclidean plane, by means of our method for TDA of graph-structured data sets (Figure 11a). We used a Rips graph $G$ with $\epsilon = 3.5$ as a graph modeling the underlying topology of Pikachu.

Figure 11b shows four different persistence diagrams resulting from $D$ (Section 1.2). One for the metric space $(D, d_{\text{euclidean}})$, one for $(D, d_G)$ where $d_G$ denotes the shortest path metric on the Rips graph $G$, one that results from approximating the diagram for $(D, d_G)$ through the method described by Cavanna et al. (2015), and finally one for $(V(B), d_G)$.

The diagram for the original point cloud data (Figure 11b, Top Left) displays many long persisting cycles, as well as some cycles corresponding to topological noise (holes with a low persistence). The diagram for $(D, d_G)$ (Figure 11b, Top Right) does not include the original large cycles with a large birth value anymore. These cycles are never born due to infinite distances between nodes in different connected components of $G$. However, the topological noise with a low birth value remains. A small amount of topological noise also remains in the approximated diagram for $(D, d_G)$ (Figure 11b, Bottom Left). However, the

(a) A forest-structured backbone graph $B$ (edges and nodes in red) for a point cloud data set resembling Pikachu. Persistent homology (Figure 11b) allows one to identify cycles that should be represented in the backbone $B$ (edges in black).




(b) Persistence diagrams for various metric spaces. (Top Left) A diagram for $(D, d_{\text{euclidean}})$. (Top Right) A diagram for $(D, d_G)$. (Bottom Left) An approximated diagram for $(D, d_G)$ using $|V(B)|$ points. (Bottom Right) A diagram for $(V(B), d_G)$.

Figure 11: Compared to the standard approaches to persistent homology, our forest structured backbone inferred through solving CLOF in the BC-pine allows for a more effective and efficient approach to identify cycles in Pikachu.

diagram for $(V(B), d_G)$ (Figure 11b, Bottom Right) also disposes of the topological noise, and what remains is exactly one point (H1) for each one of the eight 'gaps' that are still present in $B$. One for each of Pikachu's two cheeks, ears, and eyes, one for its lip, and one for its tongue. These gaps are characterized by two nodes of $B$ lying close to each other in the original graph $G$, i.e., according to $d_G$, but not in $B$, i.e., according to $d_B$.

Hence, the application of forest-structured backbones to topological data analysis goes in both directions. On the one hand, the forest-structured backbone $B$ makes the computation of persistent homology more efficient and reduces topological noise. On the other hand, persistent homology itself provides a tool for identifying the missing cycles in the backbone.

Furthermore, depending on the used implementation, the computation of persistent homology also allows one to locate a cycle that represents the hole corresponding to each one of the points (H1) in the diagram (Fasy et al., 2014). This allows one to locate the cycles that are missing a representation within the backbone topology. These cycles are shown in Figure 11a. Note that however, the cycle is computed to be underlying the metric space $(V(B), d_G)$, and might not correspond to an actual subgraph of $G$.

## 3. Experiments

In this section, we show how our method is applicable to a wide variety of graphs arising from different fields of science. For our applications, we will focus on topological data analysis, visualization, and graph simplifications. Note that graph simplifications recently showed to increase the performance of existing graph embedding methods (Chen et al., 2018). Our method will show to be applicable to a wide variety of data sets, from social networks, to high-dimensional point cloud data.

We will present the ten different data sets on which we will conduct our experiments in Section 3.1. In Section 3.2, we will discuss the baseline methods we will use to verify the effectiveness of our newly introduced method on these data sets. In Section 3.3, we qualitatively discuss our obtained results. Section 3.4 considers the introduction and results of our quantitative metrics used to measure the performance of our method. We will also conduct a separate large scale experiment on 333 cell trajectory data sets in Section 3.4.3, using a domain specific baseline and set of quantitative measures.

### 3.1 Summary of the used Data Sets

We will consider various types graphs to analyze the performance of our method.

*Swiss Roll (SR).* The Swiss Roll is a commonly used manifold for analyzing the performance of nonlinear dimensionality reductions (Tenenbaum et al., 2000). We generated a synthetic data set of 1000 points lying on such manifold. A Rips graph with 47 013 edges ($\epsilon = 0.75$) was constructed from this data for analysis through our method.

*Karate Network (K).* Zachary's karate club is a well-known social network of a karate club, studied by Wayne W. Zachary for a period of three years from 1970 to 1972 (Zachary, 1977). The network consists of 34 members of a karate club. Each one of the 78 edges between pairs of members denotes an interaction outside the club. During the study a conflict arose between the—under pseudonyms known—administrator "John A" and instructor "Mr. Hi", which led to the split of the club into two. Half of the members formed a new club around

Mr. Hi, whereas members from the other part either found a new instructor or gave up karate. This splits the network into two ground-truth communities. Furthermore, each edge is weighted with the the number of common activities the club members took part of. We mapped each edge weight to its inverse, as higher weights corresponded to more distant nodes when we introduced the BC (Vandaele et al., 2019b).

*Harry Potter Network (HP).* We consider an unweighted network $G$ displaying 221 relationships between 63 characters from the Harry Potter novels. The original graph can be found at `github.com/hzjken/character-network`, and has edges representing sentiment relationships between characters. The original edges represented either a hostile or friendly relationship. However, we only considered edges denoting friendly relationships between characters, as to detect a natural flow in the network.

*Game of Thrones Network (GoT).* We consider an unweighted network $G$ defined on 208 characters of the Game of Thrones saga[2], obtainable from `https://shiring.github.io/networks/2017/05/15/got_final`. Each one of the 326 edges between two nodes denotes either a (undirected) 'mother', 'father', or 'spouse' relationship. Due to the similarity of its concept to the previous network, we present its qualitative analysis in Appendix A.

*Co-authorship Networks: KDD & NeurIPS.* We consider two more challenging non-metric graphs, displaying co-authorship relations for two different machine learning and data mining conferences: KDD (*K*nowledge *D*iscovery and *D*ata Mining, 5749 nodes & 19 715 edges), and NeurIPS (*Neur*al *I*nformation *P*rocessing Systems, 6525 nodes & 15 770 edges). The data used to construct this graph is publicly available on `aminer.org/citation`. We only considered the largest connected components of these graphs for analysis. Each edge is weighted by the inverse of the number of papers co-authored by the corresponding two authors. Hence, low weights imply more closely connected authors. Due to their similarity, the qualitative analysis of the NeurIPS network is presented in Appendix A

*Earthquake Locations (EQ).* We obtained a data set containing information on 80 549 earthquakes ranging between the years 1950 and 2017. This data is freely accessible from USGS Earthquake Search. The topology underlying such a data set was already analyzed by Aanjaneya et al. (2012) and Vandaele et al. (2019a). However, contrary to their followed procedure, we do not restrict ourselves to a particular small rectangular domain with a low amount of noise, do not apply any noise filtering in advance, and are able to obtain a topological simplification through a $k$NN graph. A random sample of 5000 earthquake locations was taken, from which we constructed an undirected 10NN graph with 31 129 edges using the Great circle (geographic) distances between location coordinates for analysis. These distances where also used as weights of resulting edges.

*Cell Trajectories.* We will first demonstrate the effectiveness of our method for cell trajectory inference or visualization by means of a synthetic cell trajectory data set of 556 cells in a 3475-dimensional gene expression space (**SC**). This data represents a snapshot of these cells

2. Both the HP and GoT network depict relationships among individuals within 'societies'. These societies are 'good' and 'bad' in the HP network, whereas they are the houses in the GoT network. Hence, these graphs fall under the category of graphs that are studied in social sciences. Although one might argue whether these graphs can be considered 'real-world' graphs, we believe these are examples to which many readers can relate, as to confirm the effectiveness of our method for these types of graphs.

at a specific point during a cell differentiation process. Different stages in the differentiation process correspond to differentially expressed genes. Hence, the ground-truth underlying topology of the point cloud gene expression data set is the (embedding of the) differentiation network. A $k$NN graph ($k = 10$, 4646 edges) will be used to analyze the underlying topology of this data. We will conduct a similar experiment on a $k$NN graph ($k = 5$, 1543 edges) constructed from a real cell trajectory data of 355 cells embedded into a 3397-dimensional gene expression space (**RC1**), as well as on a $k$NN graph ($k = 10$, 974 edges) constructed from a second real cell trajectory data set of 154 cells in a 1770-dimensional gene expression space (**RC2**). We will use the latter data set to show how a dimensionality reduction can significantly improve our obtained results. We will use Euclidean distances to construct these graphs, and for the corresponding edge weights.

Our observations on these data sets will lead us to a new method for cell trajectory inference (Saelens et al., 2019), discussed in Section 3.4.3. We will evaluate this new method on a combination of 227 synthetic and 106 real gene expression data sets, all of which (including those above) may be obtained from `https://zenodo.org/record/1443566#.Xab5deYza02`. The number of cells (observations) ranged from 59 to 13 281, while the number of genes (features) ranged from 373 to 23 658. The underlying ground-truth topologies consisted of a mix of linear structures, bifurcating, tree graphs, forest graphs, as well as general graphs, i.e., with cycles. We will also evaluate our method for various $k$NN graphs $k \in \{5, 10, 15, 20, 25\}$ constructed from each of these considered cell trajectory data sets, to evaluate the sensitivity of our method to the choice of this parameter.

### 3.2 Summary of the Baseline Methods

We will evaluate the performance of our method by comparing our results to those obtained through three different baselines, chosen to address the following questions (Figure 2).

1. *Why do we need intermediate forest representations?*
2. *Why are our introduced pines effective forest representations?*

Similar to how we can specify the number of leaves to be selected through CLOF, each of these baseline methods will require a number of 'important' points to be selected. For comparison, we will specify each baseline method to select the same number of nodes (componentwise) as the number of leaves selected through our own method. The different baselines we will consider are summarized below.

*Facility Location + Steiner Tree (FacilitySteiner).* We will use this baseline will to investigate the applicability of Steiner trees to our problem. First, we use an intermediate step based on facility location *in metric spaces*. We start by selecting $k$ *medoids* through a partitioning around medoids (PAM) algorithm (Mitra et al., 2019). PAM is a clustering algorithm reminiscent to the ordinary $k$-means algorithm (Hartigan, 1975), but chooses data points as centers (medoids) and can be used with arbitrary distances. Once these medoids have been selected, we pass them to an algorithm for approximating a Steiner tree through these nodes, as described by Sadeghi and Fröhlich (2013). The result of this method is illustrated in Figure 2a.

*Farthest Point Sample (FarthestPoint).* Our second baseline method is inspired by the algorithm for solving CLOF in tree graphs (Theorem 17), which includes a farthest point

sampling according to a user-defined cost function. Instead, we apply a similar procedure to the original graph. We start with either the center of the graph (if we only wish to select one node), or the longest shortest path between two nodes of the graph (measured according to the original distances). Consecutively, we connect the next farthest point to the current tree by means of the shortest path between them, until a prespecified number of points has been selected. The result of this method is illustrated in Figure 2b.

*CLOF in the Regular Minimum Spanning Forest (CLOFinMSF).* Our third baseline method is also inspired by our algorithm for solving CLOF in tree graphs. However, this time we solve CLOF in the regular minimum spanning forest (MSF) constructed from the original graph. The purpose of this baseline is to illustrate the effectiveness of representing graphs through BC/LCC-pines for mining topological substructures through CLOF. The result of this method is illustrated in Figure 2c.

*Slingshot.* For our large scale cell trajectory inference experiment, we will consider another baseline method that has been specifically designed for this task. We will compare our new cell trajectory inference method to *Slingshot* (Street et al., 2018), which is currently the top ranked method for cell trajectory inference in terms of accuracy according to Saelens et al. (2019), who developed a wrapper to provide a common input and output model that allows one to compare different cell trajectory inference methods. Slingshot requires a clustering of the cells into groups to start with, builds a minimum spanning tree on these clusters, and refines the such obtained tree by means of principal curves (Hastie and Stuetzle, 1989). Note that the clustering method required for being able to compare Slingshot to other cell trajectory inference methods has been provided by Saelens et al. (2019).

### 3.3 Qualitative Analysis of the Results

*Swiss Roll.* Figure 12 shows our considered point cloud data $D$ lying on a 'Swiss Roll'. Figures 13a-13d show the Rips graph constructed $G$ from this data, as well as the backbones

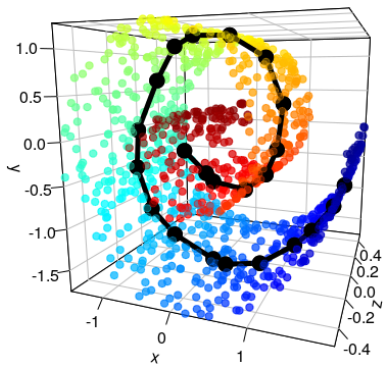
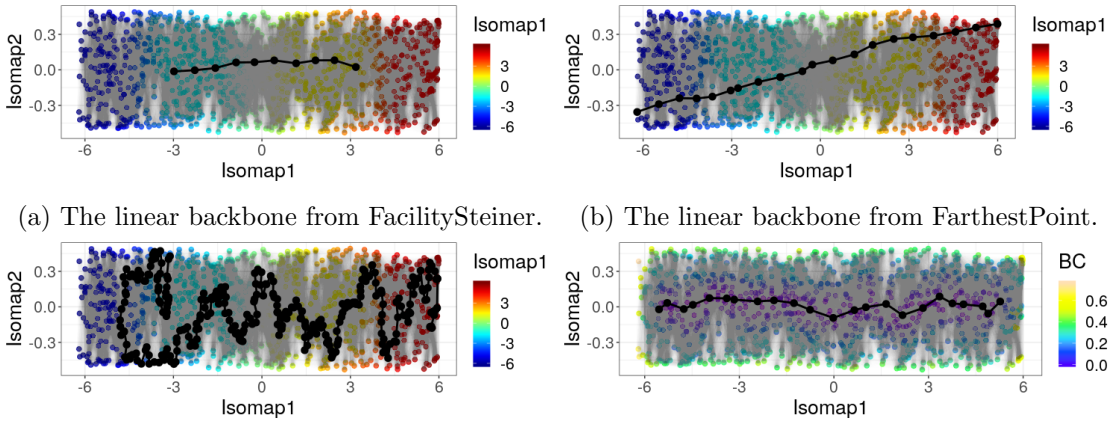



Figure 12: 3D point cloud data $D$ lying on a 'Swiss Roll'. Points are colored according to the first coordinate of the 2D Isomap embedding of $D$ based on $G$. The backbone obtained through our newly introduced method curls all the way around the 'core' of the manifold.




(a) The linear backbone from FacilitySteiner.

(b) The linear backbone from FarthestPoint.

(c) The linear backbone from CLOFinMSF.

(d) A linear backbone from the BC-pine.

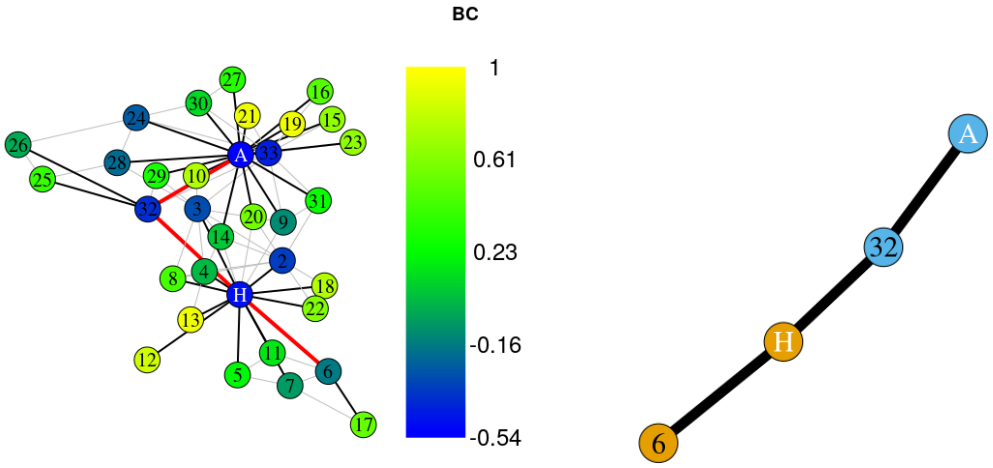
Figure 13: Various backbones (black) through a 'Swiss Roll'-shaped point cloud data set. Only through the BC-pine, we are able to infer a smooth and centered backbone that extends to the true leaves of the linear-structured model that underlies the data.

obtained through our various procedures. The 2D embedding of $D$ was obtained through an Isomap embedding of $D$ based on $G$ (Tenenbaum et al., 2000).

Various observations can be made from Figure 13. First, FacilitySteiner performs well in terms of centering the backbone and its smoothness (Figure 13a). However, it is unable to fully extend to the true 'outside' of the backbone, as the selection of medoids is not analogous to the selection of leaves through CLOF. FarthestPoint performs better in terms of fully extending to the true underlying leaves (Figure 13b). However, the resulting backbone crosses the topology diagonally instead of through its center, as it searches for the *maximal* shortest path between two nodes. CLOFinMSF performs well in approximating a vast majority of the nodes in $G$, as it 'wiggles' through the entire graph. The result of CLOFinMSF is however far from smooth, and it also does not extend to the true underlying leaf on the left (Figure 13c). In contrast, our newly introduced method of mining the backbone through solving CLOF in a BC-pine performs well in terms of centering the backbone, while also fully extending it to the true underlying leaves of $G$ (Figure 13d).

*Karate Network.* Figure 14a shows the original Karate Network, the BC-pine, as well as a linear backbone mined from this BC-pine. Figure 14b displays the backbone where nodes are colored according to their ground truth community. Given the ground-truth separation of two communities, a linear backbone fits our network well. Furthermore, this separation of the two communities is preserved by our backbone (Figure 14b). We further remark that both John A (**A**), as well as Mr. Hi (**H**), achieve a very low BC (Figure 14a), which coincides with these nodes being highly transmissive nodes in the ground-truth model.

*Harry Potter Network.* Figure 16 shows the original Harry Potter Network $G$, an LCC-pine in $G$, a forest-structured backbone $B$ mined from this LCC-pine, and a representative cycle obtained through persistent homology of $(V(B), d_G)$ (Figure 15). It turns out that only our




(a) A linear backbone (red edges) mined from the BC-pine (black edges) constructed from the karate network (grey edges).

(b) The backbone linearly separates the two ground-truth communities defined by the main characters: John A (**A**) and Mr. Hi (**H**).

Figure 14: Our method identifies a linear backbone in the Karate network, consistent with the ground-truth separation of the network into two different communities.

newly proposed method and FacilitySteiner were able to actually capture Harry Potter—the main protagonist—within the major component of the forest-structured backbone.

The other smaller components correspond to special cases. We let the backbone component of the single isolated edge of "Riddles" simply be itself (note that both vertices have betweenness 0 in any spanning forest of the graph). The component in the LCC-pine corresponding to the triangle of "Dursleys", is pruned to a single node (Theorem 21), which is chosen as the representation of this component. Note that we also did this to obtain a representation of Pikachu's nose in Figure 11a. However, unlike for the component representing this nose, all nodes in the triangle of "Dursleys" have an LCC of 1. Hence, there is no meaningful interpretation to the LCC-pine having chosen Vernon Dursley as the inner node of the corresponding linear component consisting of these three nodes in the pine.

Connoisseurs of the saga may be surprised by Harry Potter and Sirius Black being quite distant from Albus Dumbledore, according to the linear backbone component representing the major component of $G$, first needing to pass through Lord Voldemort. This is because

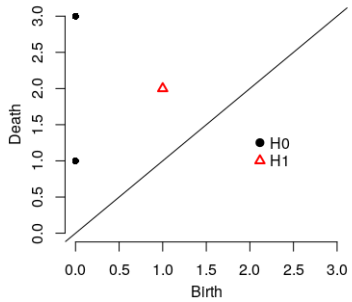



Figure 15: The persistence diagram of $(B, d_G)$ in the Harry Potter Network reveals the presence of one cycle (H1) missing in the forest-structured backbone representation. Note that the multiplicity of the top left point (H0) equals three, i.e., the number of connected components in $B$ according to $d_G$.

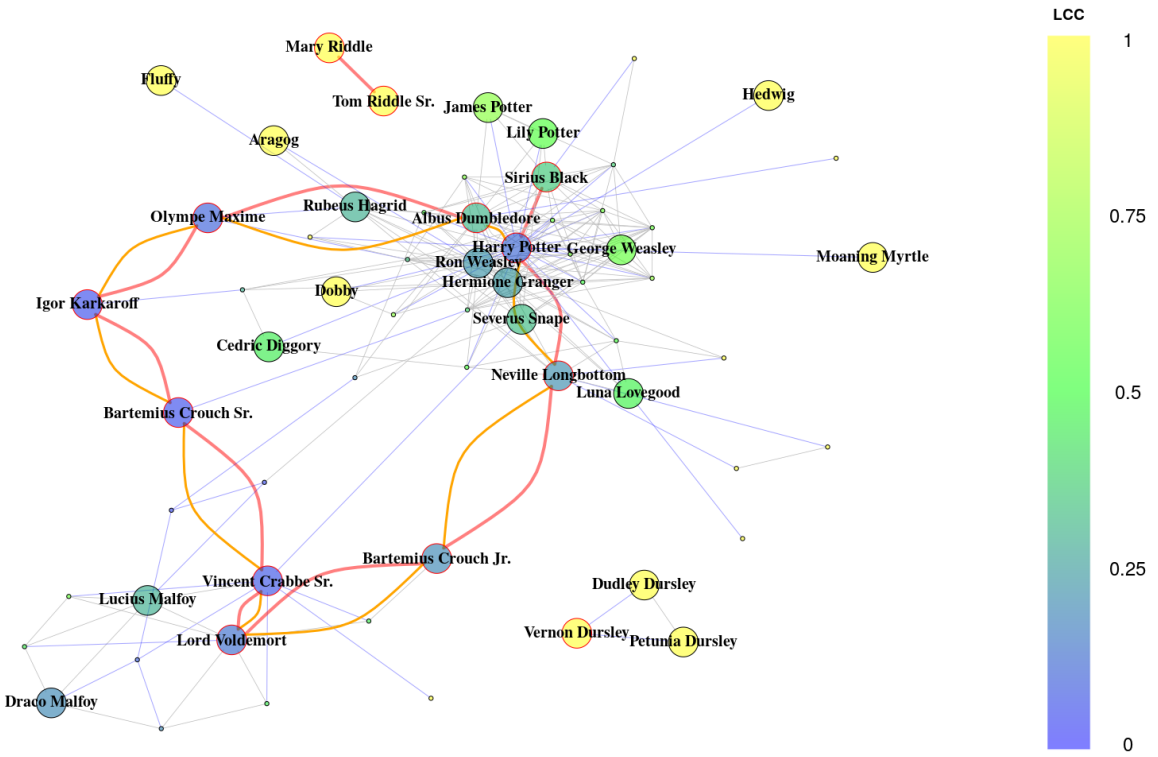



Figure 16: A backbone (red edges + red vertex borders) mined from the LCC-pine (blue edges) constructed from the Harry Potter network (grey edges). Persistent homology of the metric space induced by the original metric in $G$ on the nodes of the backbone can be used to find a representative cycle missing in the backbone (orange edges).

of the lack of ability to include cycles through a (sub)forest representation of our original graph. However, as can be seen from Figure 16, this linear component (red edges) goes all the way around through the corresponding component. Its leaves are actually very close to each other according to $d_G$. As discussed in Section 2.4.3, persistent homology allows us to discover that a cycle is missing from our backbone representation (Figure 15). Furthermore, Figure 16 also displays the representative cycle corresponding to the single identified hole in the underlying topology (orange edges), placing Harry much closer to Dumbledore in the underlying topology of the graph.

*KDD Co-authorship Network.* We applied our method to construct a tree-structured backbone with 5 leaves. The resulting tree graph is shown in Figure 17.

To verify that the cores of our tree representations, i.e., the BC-pines, indeed correspond to meaningful core structures in the original graphs, we studied various measures of our network when moving deeper into the backbone by pruning leaves of the trees, namely:

- the fraction of authors still included in the subtree;
- the average number of citations of the authors still included in the subtree;

- the average year of the first publication in the considered conference across al authors still included in the subtree.

For comparison, we compared this to the same measures for the tree representations corresponding to our baselines. This equals the regular MST for CLOFinMSF, which is the solution to (4) for the maximal possible number of points (leaves) that can be selected through the algorithm in the MST (Theorem 17). Since FacilitySteiner and FarthestPoint do not make use of an intermediate forest representation for inferring a backbone, we analogously consider the trees that result from selecting the maximal number of nodes during the algorithm. For FacilitySteiner, this equals the Steiner tree induced by all nodes, and hence, also the regular MST. For FarthestPoint, we consider the tree that results from continuing the sampling until all nodes all included. We will refer to this tree as the *FPS tree*.

The obtained metrics according to the number of pruning iterations are shown in Figure 18. By iteratively pruning leaves, we note that the BC-pine retracts much faster to a core structure than the regular MST, discarding the majority of the nodes after the first iteration, a consequence of Proposition 11. The FPS tree quickly discards of many nodes as

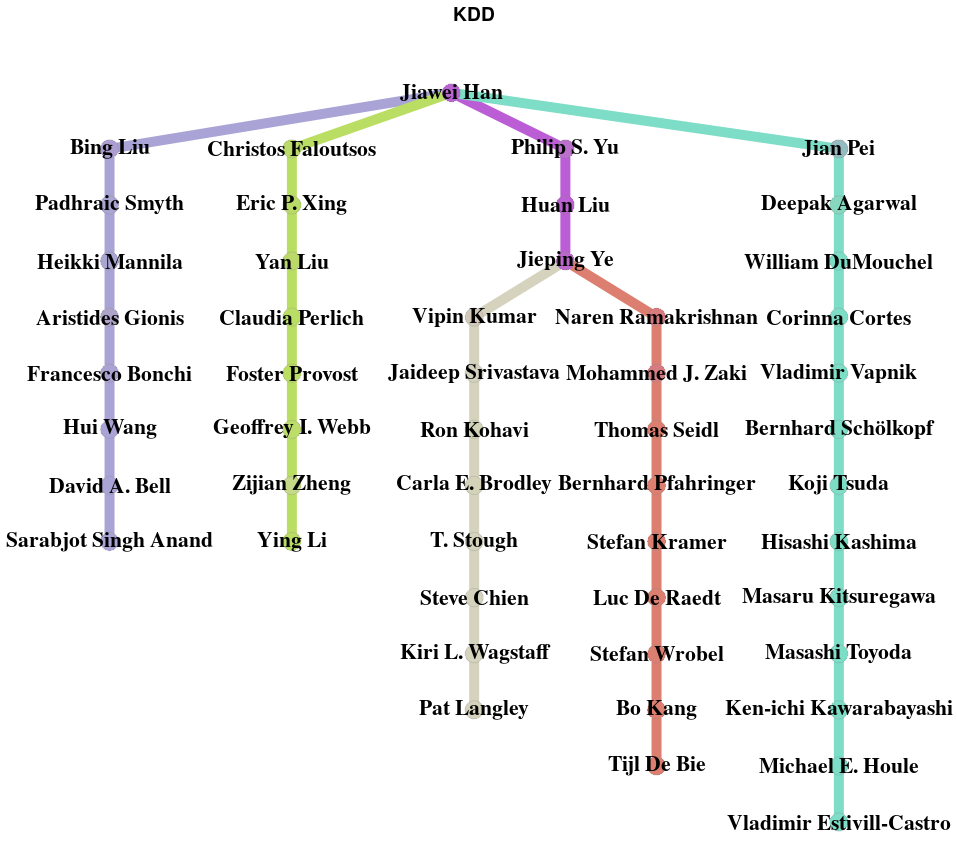



Figure 17: A tree-structured backbone for the KDD co-authorship network with 5 leaves. Nodes and edges are colored according to their closeness to one of the 6 resulting branches.

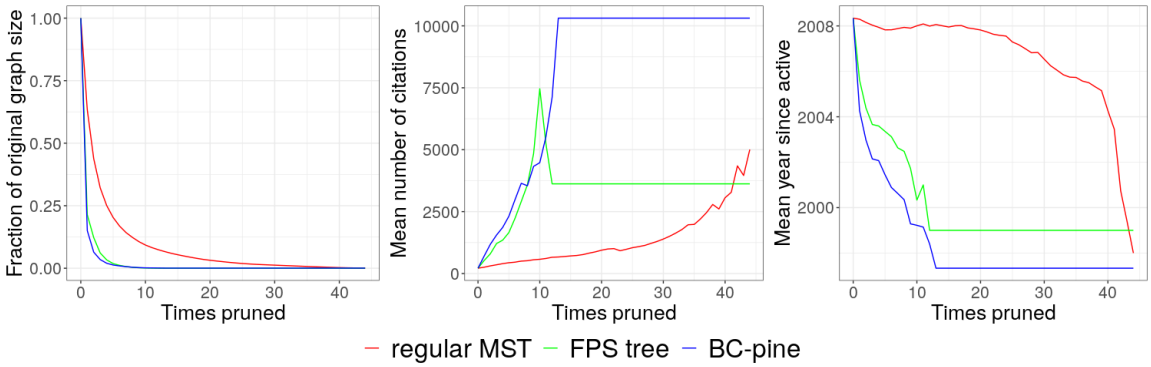



Figure 18: Various measures after iteratively pruning the regular MST (red), the FPS tree (green), and the BC-pine (blue). (Left) Fraction of original graph size. (Middle) Average number of (KDD) citations. (Right) Average year of first published (KDD) paper.

well. However, the BC-pine retracts towards a core structure marking authors with a high number of citations, and who have been present very early in the considered conference. This is exactly what we expect from the core structure of the original graph.

Note that through a previous and similar analysis, we showed the effectiveness of the BC over the LCC even in non-metric weighted graphs (Vandaele et al., 2019b).

*Earthquake Locations.* Figure 20 shows the result of constructing a forest-structured backbone for our graph representing the earthquake locations through the three baseline methods, and our newly introduced method. Each method was specified to select 35 nodes.

FacilitySteiner is again unable to extend across the entire underlying topology, leaving large patches unrepresented. The resulting backbone also only contains 23 leaves. The backbone resulting from FarthestPoint *connects* to many outliers, prone to be selected through this method. Furthermore, the backbone also *passes through* outliers, as shortest paths in the original graph will take any 'shortcut' available. The backbone also only contains 26 leaves. This is due to leaves added at one iteration being connected to other leaves in further iterations. In contrast, both backbones obtained through CLOF exactly contain 35 leaves as specified. Both extend well across the entire underlying topology, while avoiding outliers. Note that the regular MST avoids connecting through outliers through

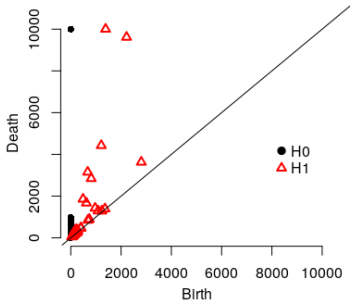



Figure 19: The persistence diagram of $(B, d_G)$ reveals the presence of many small, medium, as well as larger cycles missing from the forest-structured backbone $B$ derived through our newly introduced method.

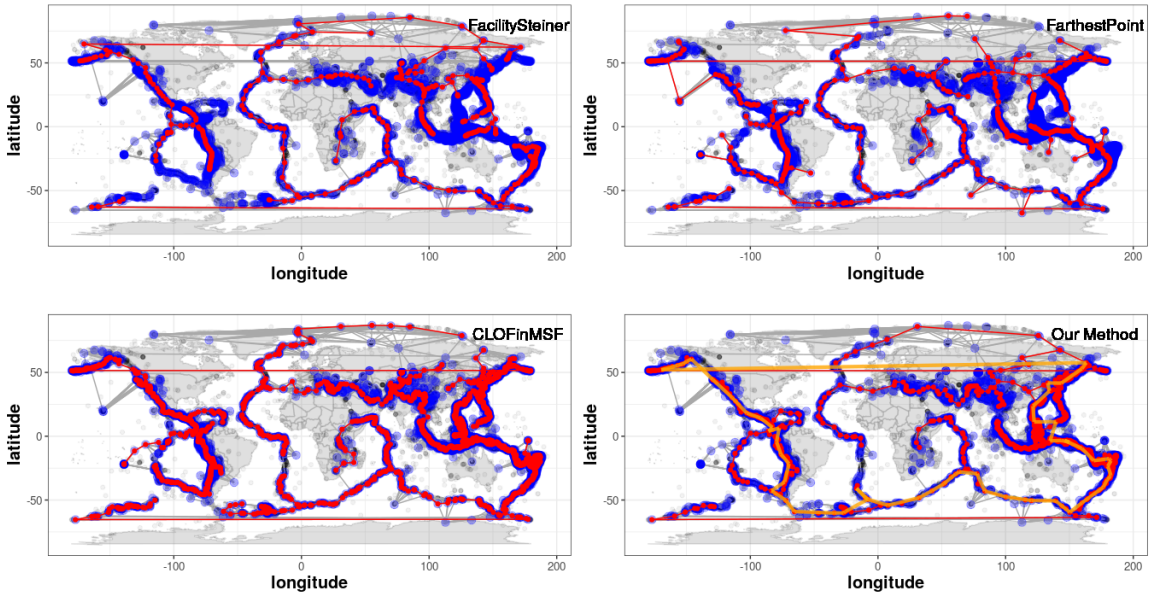



Figure 20: Backbones (red) derived from earthquakes scattered across the Earth using various methods. (Bottom Right) A representative cycle for the most persisting hole in $(B, d_G)$ identified through topological persistence is overlayed in orange. Only through the BC-pine, we are able to infer a backbone that passes smoothly through the entire graph, while avoiding outliers. We quantitatively verify this in Section 3.4

multiple edges as the corresponding weights are usually high, whereas the BC-pine avoids connecting through these as they are separated from the true core by means of boundary nodes. The main difference between the backbones obtained through CLOF is their size: through the BC-pine, we approximate the entire underlying topology of the graph with less than three times the nodes than through the regular MST (Table 2). This is again a consequence of CLOFinMSF resulting in a very 'wiggled' backbone.

Figure 19 shows the persistence diagram of $(V(B), d_G)$—$B$ being the backbone derived through our newly introduced method—which indicates that there are many small, medium, as well as larger cycles missing from the forest-structured backbone. Figure 20 (Bottom Right) also displays a cycle representing the most persisting hole, spanning the entire Earth.

*Cell Trajectories.* Figures 21 & Figures 22 visualize our obtained results for the first two cell trajectory networks discussed in Section 3.1, as well as the known ground-truth underlying topologies (Figures 21e & 22b). The point cloud data, as well as the obtained $k$NN graphs and the resulting backbones, are visualized on multidimensional scaling plots using Pearson correlations as distances between cells.

It should be noted that even though the constructed proximity graph may incorrectly connect different parts of one or multiple branches, our method is well able to both correctly identify and locate the present linear topology. E.g., Figure 21d shows that even though the

$k$NN graph forms a cyclic structure through the entire data, our method is able to correctly identify and locate the ground-truth linear underlying topology. In contrast, each one of our baseline methods incorrectly connects the two underlying leaves (Figures 21a-21c).

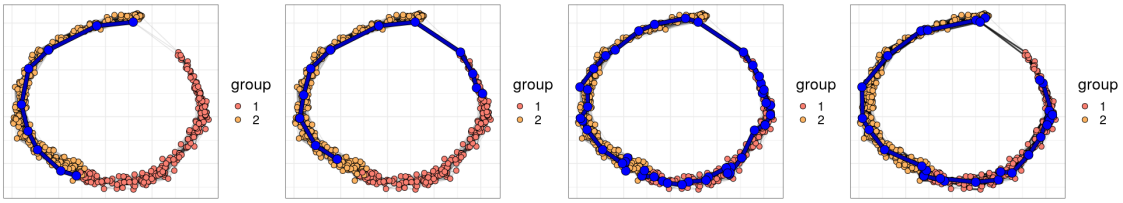
Even without CLOF to infer topologies from our pine, Figure 21g & 22d show that the BC-pine visualizes the ground-truth topologies present in the data well through the Fruchterman-Reingold layout algorithm (Fruchterman and Reingold, 1991). For comparison, we also illustrated the regular minimum spanning tree (MST) of the graphs (Figure 21f & 22c), from which it is a lot more difficult to visually deduce the ground-truth underlying topologies. Furthermore, it can not be deduced from the visualization in Figure 21f that two groups of cells are actually connected through the incorrect region (Figure 21c).

As our method is specifically developed for topological data analysis of graphs, it is important that the ground-truth underlying topology of our point cloud data corresponds to the underlying topology of our graph used to represent this data. This is generally a very difficult task, especially for high-dimensional data sets that commonly suffer from the curse of dimensionality. E.g., the Euclidean distance measure and the concept of closest neighbors become much less meaningful in high-dimensional spaces (Aggarwal et al., 2001). This may lead to a low quality representation of the data's underlying topology through a $k$NN graph. Though our method was able to infer the topology of the underlying cell trajectory network in our previous examples, we do note that the stated observation already applies to the constructed proximity graphs. Their quality for representing the underlying topology is affected by interconnections between different parts of branches (Figure 21d) or 'hubs' that connect to many other points (Figure 22a). The latter is a typical problem occurring in $k$NN graphs constructed from high-dimensional data (Radovanović et al., 2009).

We continue with our third cell trajectory data set, which is an extreme example of how the curse of dimensionality may affect our results. Figure 23a visualizes the point cloud data set $D$ using diffusion map coordinates, as well as the BC-pine of a 10NN graph $G$ constructed from $D$ using Euclidean distances in the original high-dimensional space. The inferred backbone $B$, also shown in Figure 23a, is a single node. This is because a data point $v$ is the closest neighbor of 147 out of the 153 other data points. In the 10NN graph constructed from the high-dimensional data, every one of these 153 data points is connected to $v$, and the obtained BC-pine is a star graph as in Figure 7a. Making use of Theorem 21, pruning discards all of these 153 nodes, and what remains is the single node $v$.

Clearly, Figure 23a shows that the underlying topology of $G$ is not a truthful representation of the underlying ground-truth topology of $D$. However, as commonly used in cell trajectory inference tools (Saelens et al., 2019), a dimensionality reduction may serve as a first step for reducing the amount of noise and improving the quality of our representation.

Figure 23b visualizes the point cloud data set $D$ using the same (first two) diffusion map coordinates, but this time the BC-pine of a 10NN graph $\tilde{G}$ constructed from the first three diffusion coordinates of this embedding. The underlying topology of $\tilde{G}$ is now a much better reflection of the underlying ground-truth topology of $D$ (Figure 23c), and is successfully mined through solving CLOF in the BC-pine. Again, comparing Figures 23d & 23e, we note that the BC-pine serves as a tool for graph visualization as well.

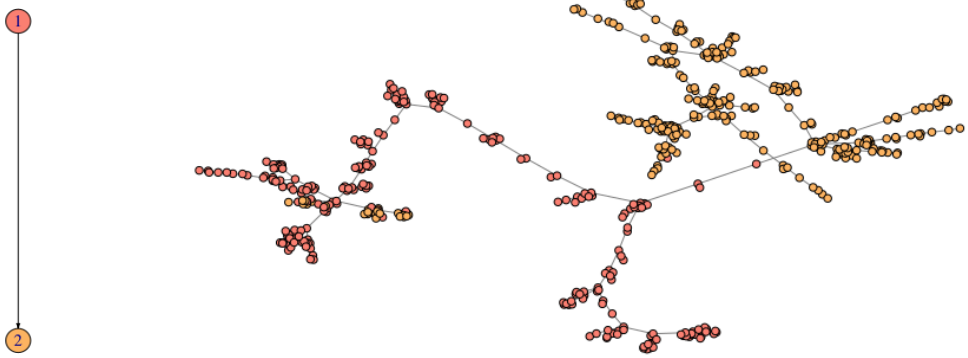



(a) A linear backbone (blue) mined through FacilitySteiner.

(b) A linear backbone (blue) mined through FarthestPoint.

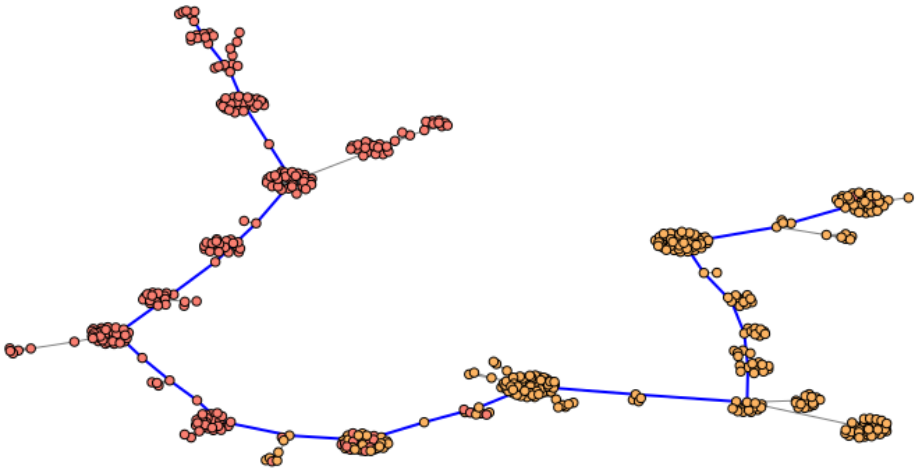
(c) A linear backbone (blue) mined through CLOFinMSF.

(d) A linear backbone (blue) mined from the BC-pine.



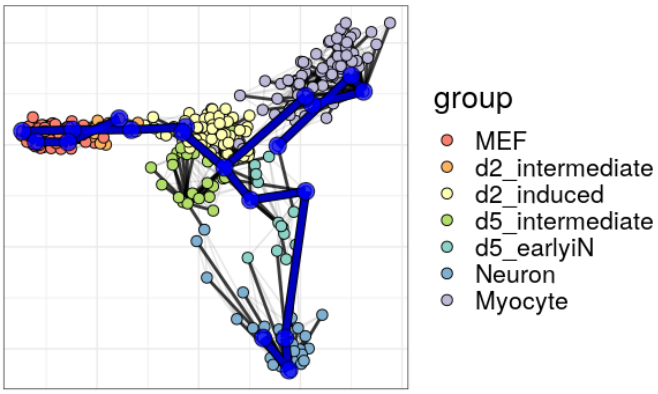

(e) The ground-truth underlying topology of $D$. One group of cells (1) evolves to another (2) through a linear differentiation process.

(f) Visualizing the MST of $G$ using the Fruchterman-Reingold layout algorithm.

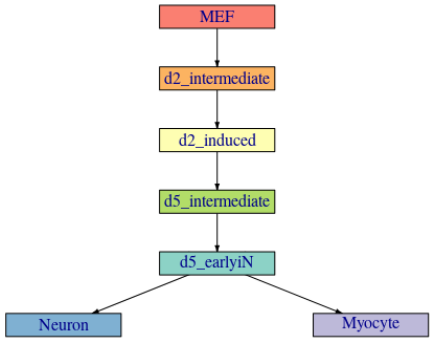


(g) Visualizing the BC-pine of $G$ using the Fruchterman-Reingold layout algorithm, with the edges of the found linear backbone in blue.

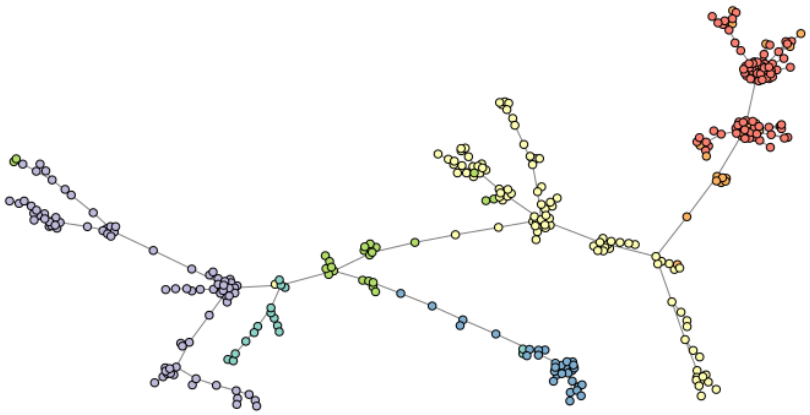
Figure 21: The BC-pine allows one to both visualize and infer the underlying topology from a synthetic high-dimensional gene expression data set through a 10NN graph $G$.



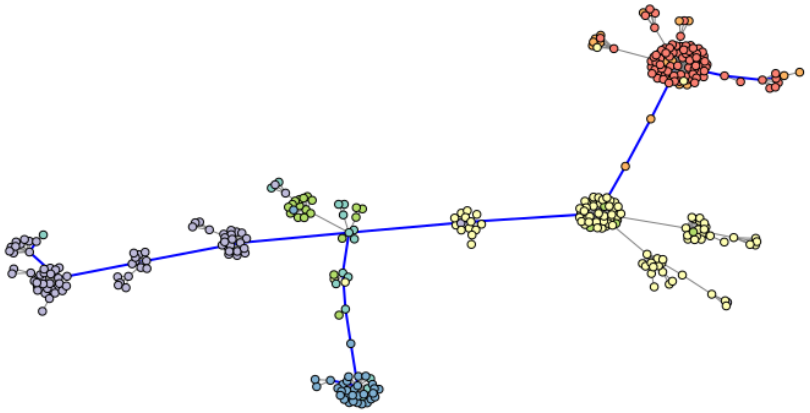

(a) A bifurcating backbone (blue) is mined from the BC-pine (black edges) constructed from a 5NN graph $G$ (grey edges) of real gene expression data $D$.




(b) The ground-truth underlying topology of $D$.

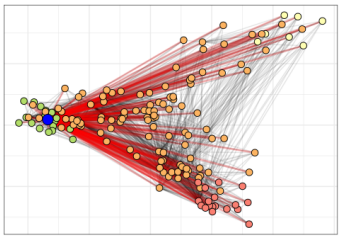


(c) Visualizing the MST of $G$ using the Fruchterman-Reingold layout algorithm.

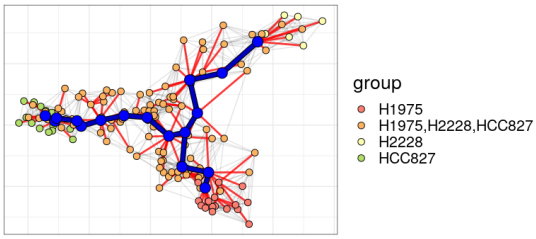


(d) Visualizing the BC-pine of $G$ using the Fruchterman-Reingold layout algorithm, with the edges of the found bifurcating backbone in blue.

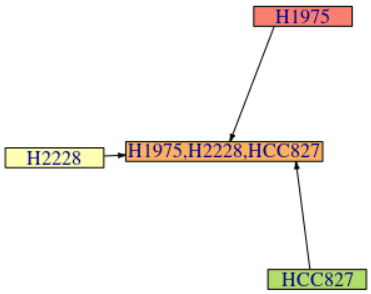
Figure 22: The BC-pine allows one to both visualize and infer the underlying topology from a real high-dimensional gene expression data set through a 5NN graph $G$.



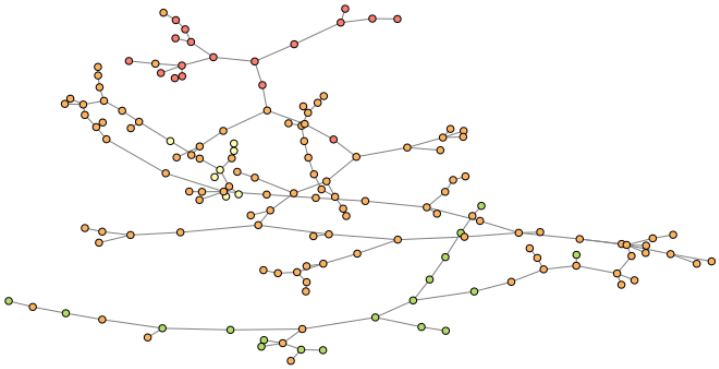

(a) A single node backbone (blue) is mined from the BC-pine (red edges) constructed from a 10NN graph $G$ (grey edges) of real gene expression data $D$.



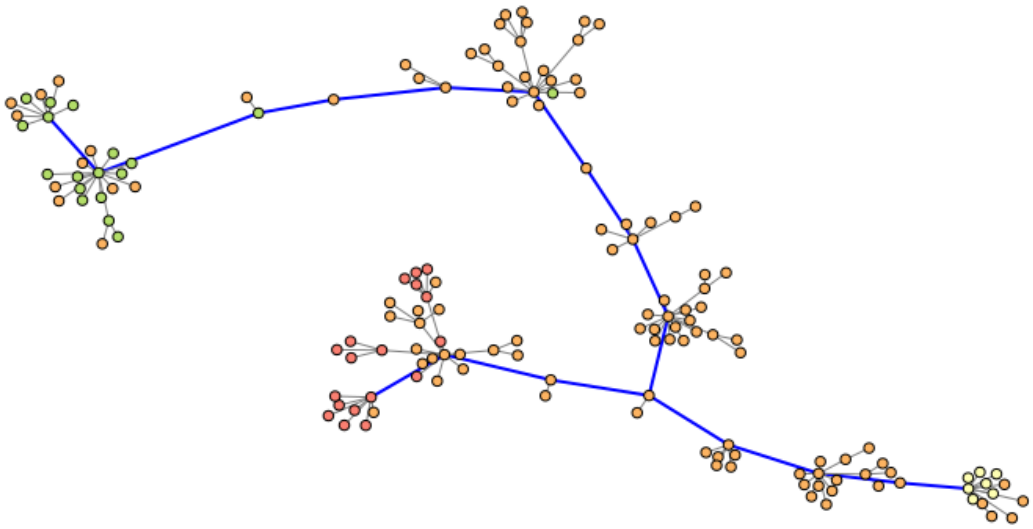

(b) A bifurcating backbone (blue) is mined from the BC-pine (red edges) constructed from a 10NN graph $\tilde{G}$ (grey edges) of a 3-dimensional embedding of $D$ into diffusion map coordinates.




(c) The ground-truth underlying topology of $D$.



(d) Visualizing the MST of $\tilde{G}$ using the Fruchterman-Reingold layout algorithm.



(e) Visualizing the BC-pine of $\tilde{G}$ using the Fruchterman-Reingold layout algorithm, with the edges of the found bifurcating backbone in blue.

Figure 23: The BC-pine allows one to both visualize and infer the underlying topology from a real high-dimensional gene expression data set through a 10NN graph $\tilde{G}$.

### 3.4 Quantitative Analysis of the Results

As to evaluate the performance of our method, ideally one would have access to a ground-truth underlying graph-structured topology. However, not only is it difficult to assess whether there exists a homeomorphic mapping from one graph to another (Lapaugh and Rivest, 1980), more often than not we do not have access to a ground-truth model.

Nevertheless, in Section 3.4.1 we will introduce general metrics allowing us to support and interpret our obtained results, which we will summarize in Section 3.4.2.

Furthermore, we will conduct a separate large scale cell trajectory inference experiment in Section 3.4.3. The knowledge of ground-truth topologies as well as the recent development of quantitative metrics for comparing different cell trajectory inference tools (Saelens et al., 2019), allows us to objectively measure the performance of our method for this purpose.

#### 3.4.1 Introducing General Quantitative Metrics

We will consider a variety of metrics that allow us to meaningfully interpret how well our inferred forest-structured backbone $B$ models the underlying topology of a graph $G$. We distinguish between four different properties we want our backbone to satisfy.

*Backbone Size.* The relative size of $B$ compared to $G$ should be small for a good simplification. Hence, we will consider $n\%$, the percentage of nodes from $G$ still included in the resulting backbone. As there is a direct relation between the original graph, $n\%$, and the percentage of edges still included in a forest-structured backbone, we will omit the latter.

*Goodness of Fit.* Though we prefer a simplification that significantly reduces the size of the original graph $G$, we also want our backbone to be a good approximation of $G$. Therefore, we consider the metric

$$R(B) \coloneqq 1 - \frac{\sum_{v \in V(G)} d_G(v, B)}{\sum_{v \in V(G)} d_G(v, C_G)},$$

where $d_G(v, B)$ denotes the distance of $v$ to its closest node in $B$, and $d_G(v, C_G)$ denotes the distance of $v$ to its closest node in the *center* of $G$, defined as the union of nodes that have minimum eccentricity in their respective connected component of $G$. Inspired by the *coefficient of determination* in linear regression, we consider this metric to be a measure for how much of the 'variance' in our graph $G$ is explained by the model $B$. However, we do not consider squared distance as in the usual definition of this coefficient, to lessen the effect of outliers. As we do not assume $G$ to reside in any vector space, there is no definition of an 'average' node. Hence, our notion of 'mean' is fulfilled by the 'center' in graphs.

*Smoothness.* We want our backbone $B$ to pass 'smoothly' through our graph, instead of 'wiggling' through many nodes (compare to overfitting in linear regression). To this end, we compute the *projection* $G_B$ *of* $G$ *on* $B$, by connecting each node of $G$ through its closest path to $B$. We then compute the ratio $\sigma \coloneqq \frac{d_G(u,v)}{d_{G_B}(u,v)}$, where $u$ and $v$ are the most distant nodes in $G$. Hence, $\sigma(B)$ denotes how well the distance between the two furthest points in $G$ is preserved through $B$. Note that trivially $\sigma(B) = 1$ if $B$ was obtained through FarthestPoint. Furthermore, $\sigma$ is 'penalized', i.e., further from 1, when centering our backbone and preventing it from passing through outliers. However, this is exactly what we want.

*Commute Time Preservation.* One of the most difficult qualities of a forest-structured backbone $B$ to assess, is how well it preserves the metric induced by the original graph $G$. Missing cycles may result in failing to preserve the original metric, whereas in the case of curvature and outliers, we actually wish to not preserve the original metric as close as possible. To this end, we consider the correlation $\text{cor}_{\text{act}}$ between the *average commute times* of the original graph $G$ and the projection of $G$ on $G_B$ (Fouss et al., 2007). These commute times are based on a Markov-chain model of random walk through the graph, which have shown to effectively deal with outliers in graphs (Moonesignhe and Tan, 2006). Note that however, this metric does not account for the fact that we cannot include cycles in a forest-structured backbone.

### 3.4.2 Quantitative Summary of our Results

Table 1 summarizes the settings in terms of leaves we used to obtain the backbones for the graphs discussed in Section 3.3 through our method. Each of our baselines also requires a number of nodes to be selected per component, for which we used the $k_{\text{comp}}$-row of Table 1.

| | SR | K | HP | GoT | NeurIPS | KDD | EQ | SC | RC1 | RC2 |
|---|---|---|---|---|---|---|---|---|---|---|
| $k_{\max}$ | | | | | 10 | 10 | 50 | | | |
| $k_{\text{spec}}$ | | | | 10 | 5 | 5 | 35 | | | |
| $k_{\text{comp}}$ | 2 | 2 | 2-2-0 | 8-2 | 5 | 5 | 35 | 2 | 3 | 3 |

Table 1: Input and output summary to obtain a forest-structured backbone through BCB. $\mathbf{k}_{\max}$: upper bound on the total number of leaves for solving CLOF to increase efficiency (blank if not specified). $\mathbf{k}_{\text{spec}}$: the specified number of leaves to be selected once the subforest had been fully grown or up until the maximum number of leaves (blank if estimated through minimizing second-order finite differences). $\mathbf{k}_{\text{comp}}$: the number of resulting leaves for each connected component (0 meaning a backbone component consisting of one node).

Table 2 summarizes the network sizes for each graph $G$ ($n = |V(G)|$, $m = |E(G)|$), runtimes, and our quantitative results for the metrics introduced in Section 3.4.1. All these results were obtained using non-optimized R code on a machine equipped with an Intel® Core™ i7 processor at 2.6GHz and 8GB of RAM. Note that the specified runtimes in Table 2 for our method are those when given the (possibly infinite) upper bounds $k_{\max}$ from Table 1. The runtimes for the baselines are those when specified to exactly select $k_{\text{comp}}$ nodes. We observe that our method scales well to graphs with thousands of nodes. We will discuss further directions for scaling our method to higher order graphs in Section 5.

Our method also scales well according to the topological complexity, whether given or bounded by a number of leaves. In contrast, FacilitySteiner is significantly slower when selecting more medoids in higher order graphs, taking 8.6min for the earthquakes data. Since the placement of medoids through FacilitySteiner is density based, it also struggles to identify important regions that make up the topology, even when specified to select the correct number of nodes to do so (Figure 2a). Clear examples of this are the bifurcating real cell trajectory data sets, for which FacilitySteiner infers a linear trajectory. Furthermore,

| | | time | $n\%$ | $R$ | $\sigma$ | $\text{cor}_{\text{act}}$ | $k_{\text{comp}}$ |
|---|---|---|---|---|---|---|---|
| Swiss Roll (SR) $n = 1000$ $m = 47\,013$ | FacilitySteiner | 4.19s | 1.0% | 0.69 | 0.997 | 0.48 | 2 |
| | FartherstPoint | 4.71s | 2.1% | 0.88 | 1 | 0.49 | 2 |
| | CLOFinMSF | 2.66s | 28% | 0.93 | 0.62 | 0.16 | 2 |
| | ClOFinBC-pine | 3.41s | 2.3% | 0.89 | 0.97 | 0.49 | 2 |
| Karate (K) $n = 34$ $m = 78$ | FacilitySteiner | 0.046s | 12% | 0.40 | 1 | 0.35 | 2 |
| | FarthestPoint | 0.005s | 21% | 0.48 | 1 | 0.36 | 2 |
| | CLOFinMSF | 0.31s | 26% | 0.54 | 0.90 | 0.39 | 2 |
| | ClOFinBC-pine | 0.018s | 12% | 0.44 | 0.95 | 0.39 | 2 |
| Harry Potter (HP) $n = 63$ $m = 221$ | FacilitySteiner | 0.029s | 10% | 0.50 | 1 | 0.96 | 2-2-0 |
| | FarthestPoint | 0.007s | 16% | 0.53 | 1 | 0.96 | 2-2-0 |
| | CLOFinMSF | 0.067s | 16% | 0.53 | 1 | 0.96 | 2-2-0 |
| | ClOFinLCC-pine | 0.23s | 21% | 0.56 | 1 | 0.96 | 2-2-0 |
| GoT $n = 208$ $m = 326$ | FacilitySteiner | 0.22s | 17% | 0.72 | 1 | 0.93 | 7-2 |
| | FarthestPoint | 0.026s | 29% | 0.76 | 1 | 0.90 | 8-2 |
| | CLOFinMSF | 0.13s | 26% | 0.78 | 0.95 | 0.90 | 8-2 |
| | ClOFinLCC-pine | 0.40s | 29% | 0.81 | 1 | 0.91 | 8-2 |
| KDD $n = 5747$ $m = 19\,751$ | FacilitySteiner | 20s | 0.19% | 0.22 | 0.94 | 0.13 | 3 |
| | FarthestPoint | 11s | 0.77% | 0.21 | 1 | 0.14 | 5 |
| | CLOFinMSF | 4.5s | 3.3% | 0.32 | 0.55 | 0.12 | 5 |
| | ClOFinBC-pine | 32s | 0.87% | 0.31 | 0.87 | 0.16 | 5 |
| NeurIPS $n = 6252$ $m = 15\,770$ | FacilitySteiner | 20s | 0.19% | 0.14 | 1 | 0.15 | 3 |
| | FarthestPoint | 12s | 0.72% | 0.18 | 1 | 0.16 | 5 |
| | CLOFinMSF | 5s | 1.9% | 0.23 | 0.95 | 0.12 | 5 |
| | ClOFinBC-pine | 30s | 0.88% | 0.26 | 0.84 | 0.14 | 5 |
| Earthquakes (EQ) $n = 5000$ $m = 31\,146$ | FacilitySteiner | 8.6min | 8.4% | 0.96 | 0.81 | 0.56 | 23 |
| | FarthestPoint | 14s | 9.6% | 0.97 | 1 | 0.59 | 26 |
| | CLOFinMSF | 14s | 54% | 0.99 | 0.69 | 0.28 | 35 |
| | ClOFinBC-pine | 24s | 16% | 0.99 | 0.79 | 0.58 | 35 |
| Synth. cells (SC) $n = 556$ $m = 4636$ | FacilitySteiner | 0.16s | 1.6% | 0.59 | 0.9994 | 0.52 | 2 |
| | FarthestPoint | 0.14s | 2.2% | 0.59 | 1 | 0.48 | 2 |
| | CLOFinMSF | 0.034s | 7.9% | 0.82 | 0.56 | 0.50 | 2 |
| | ClOFinBC-pine | 0.49s | 5.2% | 0.81 | 0.53 | 0.65 | 2 |
| Real cells 1 (RC1) $n = 355$ $m = 1543$ | FacilitySteiner | 0.073$s$ | 2.3% | 0.50 | 1 | 0.67 | 2 |
| | FarthestPoint | 0.05s | 4.2% | 0.59 | 1 | 0.68 | 3 |
| | CLOFinMSF | 0.052s | 9.6% | 0.67 | 0.52 | 0.64 | 3 |
| | ClOFinBC-pine | 0.15s | 5.4% | 0.64 | 0.78 | 0.67 | 3 |
| Real cells 2 (RC2) $n = 154$ $m = 974$ | FacilitySteiner | 0.048s | 6.5% | 0.60 | 1 | 0.67 | 2 |
| | FarthestPoint | 0.017s | 11% | 0.67 | 1 | 0.69 | 3 |
| | CLOFinMSF | 0.029s | 34% | 0.83 | 0.60 | 0.46 | 3 |
| | ClOFinBC-pine | 0.38s | 11% | 0.73 | 0.81 | 0.68 | 3 |

Table 2: Quantitative summary of our experimental results. Our newly introduced method for mining backbones in graphs through forest representations is marked in blue.

unlike FarthestPoint and the backbones constructed through CLOF, there is no straightforward way to extract the result for a lower number of selections through the output of FacilitySteiner. One needs to rerun the entire algorithm for this, making it difficult to tune the resulting topological complexity through FacilitySteiner.

Overall, not a single metric is able to fully quantify the global performance of each method on its own. Note that we did not mark 'winning' values, as extremes may also indicate a bad performance (e.g., $\sigma$ is always 1 for FarthestPoint). However, we note that these metrics strongly support our observations up until now. E.g, CLOFinMSF often results in the best approximation of the original graph (high $R$), at the cost of including many more nodes (high $n\%$), and 'wiggling' through all of them (low $\sigma$). In contrast, our method provides a backbone that approximates the original graph nearly as well (sometimes even better), using much fewer nodes in order to do so. This is most notably the case with the Swiss Roll ($n\% = 2.3\%$, $R = 0.89$ with our method vs. $n\% = 28\%$, $R = 0.93$ with CLOFinMSF), the KDD network ($n\% = 0.87\%$, $R = 0.31$ vs. $n\% = 3.3\%$, $R = 0.32$), the NeurIPS network ($n\% = 0.88\%$, $R = 0.26$ vs. $n\% = 1.9\%$ and $R = 0.23$), and the earthquakes ($n\% = 16\%$, $R = 0.99$ vs. $n\% = 54\%$, $R = 0.99$). Our method also results in a consistent smoothing for the co-authorship graphs, unlike CLOFinMSF (respectively, $\sigma = 0.87$ (KDD), $\sigma = 0.84$ (NeurIPS) vs. $\sigma = 0.55$ (KDD), $\sigma = 0.95$ (NeurIPS)).

FacilitySteiner and FarthestPoint often result in smaller (low $n\%$) and smoother (high $\sigma$) backbones, however, at the cost of providing worse approximations (low $R$) of the original graphs. This can be either due to failing to span the entire graph (FacilitySteiner), or failing to center the resulting backbone in the graph (FarthestPoint).

All methods perform similarly well in terms of preserving the average commute times. The most notable exceptions are where either CLOFinMSF performs worse, such as the Swiss Roll ($\text{cor}_{\text{act}} = 0.49$ with our method vs. $\text{cor}_{\text{act}} = 0.16$ with CLOFinMSF), the earthquakes ($\text{cor}_{\text{act}} = 0.58$ vs. $\text{cor}_{\text{act}} = 0.28$), and the second real gene expression data set ($\text{cor}_{\text{act}} = 0.68$ vs. $\text{cor}_{\text{act}} = 0.46$), or where our method performs better, i.e., for the synthetic cells ($\text{cor}_{\text{act}} = 0.65$ vs. a maximum of 0.52 for the other methods).

The main networks contradicting the observations above, are the unweighted networks (HP & GoT). Here, our method actually results in the best approximation of the original graph ($R$ equals 0.56 and 0.81, respectively), while remaining a smooth approximation ($\sigma$ equals 1 twice) that well preserves the average commute times ($\text{cor}_{\text{act}}$ equals 0.96 and 0.91, respectively). Furthermore, our method appears to provide the least smooth approximation for the synthetic cell trajectory data set at first sight ($\sigma = 0.53$). However, through Figures 21a-21d, we have shown that our method was the only method capable of identifying the true underlying topology. The longest shortest path in the original graph—used for computing $\sigma$—incorrectly connects the two underlying leaves (Figure 21b), which explains this result.

### 3.4.3 Large Scale Cell Trajectory Inference

The results we obtained through our newly introduced method, in this section abbreviated to **BCB** (**B**oundary **C**oefficients to **B**ackbone), on the cell trajectory data sets in Section 3.3, lead to a new *cell trajectory inference* method.

1. start from a (high-dimensional) gene expression data set $D$;
2. use a dimensionality reduction method to reduce the (high-dimensional) noise in $D$;

3. construct a $k$NN graph $G$ from the lower dimensional representation of $D$;
4. construct the BC-pine in $G$;
5. identify the underlying topology from the pine by means of (4).

Contrary to many of the existing cell trajectory inference tools (Saelens et al., 2019), we do not require the data to be represented in a vector space, can infer more complex topologies than linear, bifurcating, or connected ones, and do not preprocess the data through a clustering method.

Each of our 333 considered data sets described in Section 3.1 was embedded in a 20-dimensional space using diffusion maps with Pearson correlations and standard settings in R. Consecutively, we estimated the intrinsic dimension $d$, $3 \leq d \leq 20$, of our data using an inference method based on the eigen-multipliers of the embedding. We implemented the same inference method as the wrapper for *Slingshot* developed by Saelens et al. (2019). The code for this wrapper can be found on `https://github.com/dynverse`. We evaluated our method over multiple values $k \in \{5, 10, 15, 20, 25\}$ for building $k$NN graphs from our diffusion coordinates, using the Euclidean distance between points. The resulting proximity graph was used to construct a BC-pine, from which we mined the underlying topology using CLOF. A number of leaves $l$, $2 \leq l \leq 30$, was estimated for each connected component in our BC-pine, by minimizing the second-order finite differences of the function mapping the number of leaves to the corresponding vertex betweenness cost, as discussed in Section 2.4.2.

We used the four metrics suggested in Saelens et al. (2019) to quantify the quality of our inferred trajectories, which we summarize below. For full details on the exact computation of these metrics, we refer to Saelens et al. (2019).

- *The correlation between geodesic distances*, measuring if the positioning of cells is similar in the ground-truth and inferred trajectory.
- *The Hamming-Ipsen-Mikhailov (HIM) metric*, measuring the similarity of the weighted adjacency matrices of the ground-truth and inferred trajectory.
- *The F1 score between branch assignments*, measuring the similarity between the assignment to branches in the ground-truth and inferred trajectory.
- *The correlation between important features*, measuring if the same differentially expressed features are found in both the ground-truth and inferred trajectory.

All these metrics lie within $[0, 1]$. Higher values correspond to better performances. We also evaluated the computational cost of our approach in terms of runtime (in seconds) and storage (in GB). Figure 24 visualizes the performance for each considered metric, as well as for various choices of $k$(NN graphs), through cumulative distribution plots.

We first note that the overall performance of our cell trajectory inference method is stable when it comes to the choice of $k$. In terms of the obtained results, our method turns out to be comparable to Slingshot. We especially note an increase in performance when it comes to the correlation between geodesic distances. Furthermore, our method scales at least as well as Slingshot in terms of runtime, which is mostly affected by the choice of $k$, i.e., the density of $\mathcal{H}_2(D)$ (Theorem 8). In terms of storage, our method does scale worse. However, it turns out this is due to computing and storing the entire Pearson correlation matrix for the embedding. Given the dimensionality reduction, our method scales better in

terms of memory than Slingshot, through implementing 8 by means of sparse matrices (see Algorithm 1 in Appendix B), and cleverly making use of Theorem 21.

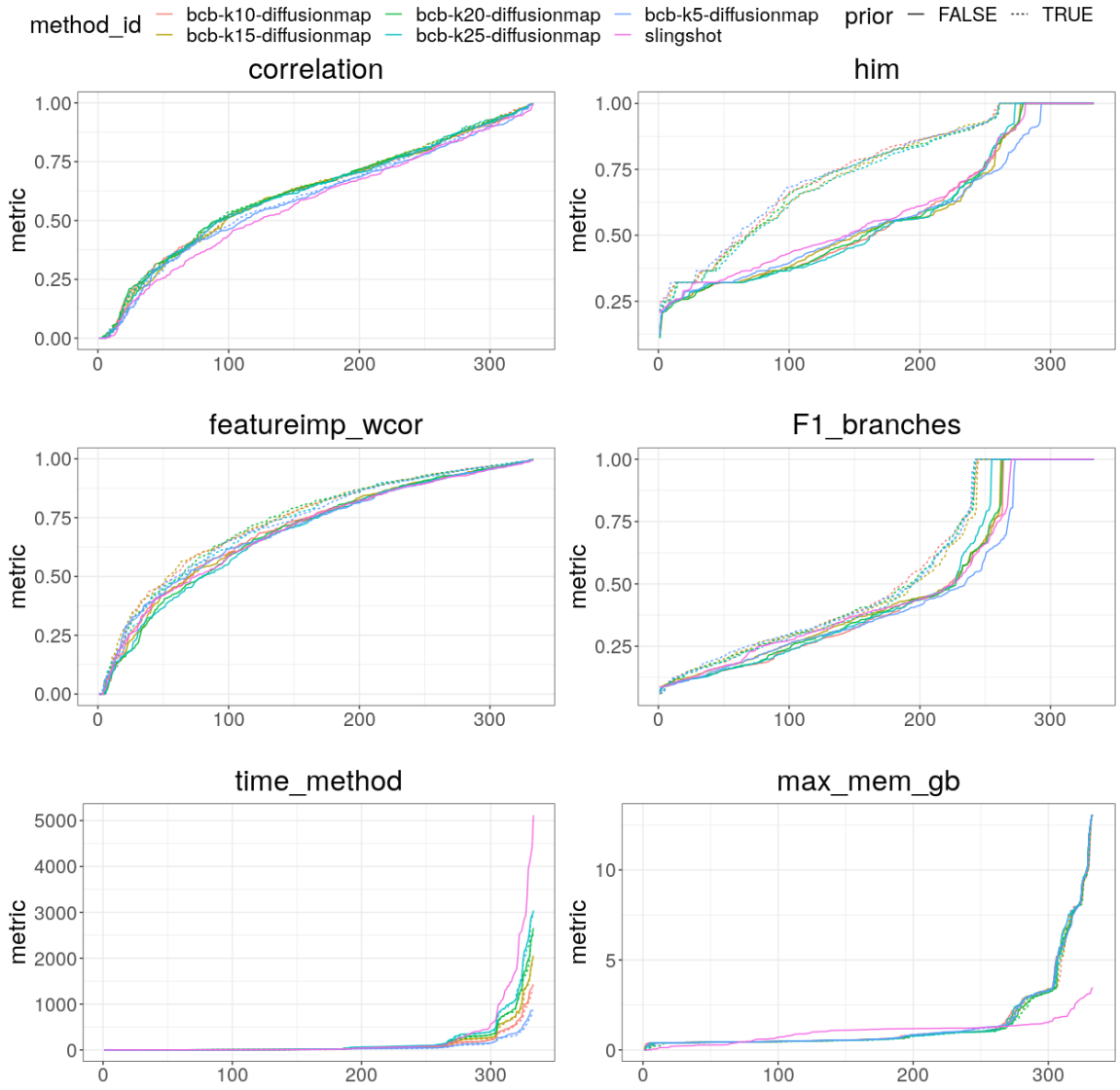



Figure 24: Various metrics for evaluating BCB as a cell trajectory inference tool, with and without the number of leaves as prior, sorted according to performance for each method. Our method shows to be comparable to the state-of-the-art for cell trajectory inference for all considered metrics. However, unlike Slingshot, our method allows us to pass knowledge of the number of leaves to the CLOF-algorithm, increasing the overall performance.

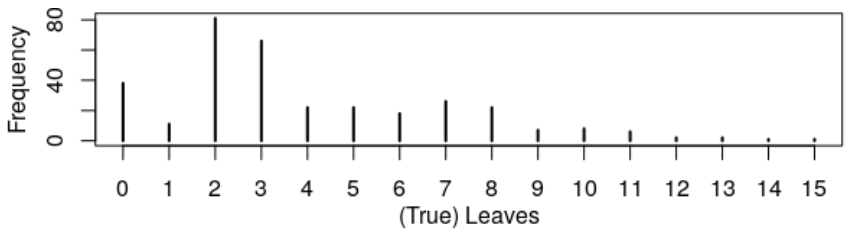



Figure 25: Distribution of the number of leaves across our 333 cell trajectory data sets.

Slingshot does seem to perform better when it comes to the HIM-metric. Investigating this further, the HIM and F1 scores are mostly affected by the prediction of how many leaves are present in the underlying topology. Figure 25 shows the distribution of the number of leaves across our 333 cell trajectory data sets described in Section 3.1. Note that a trajectory with less than two leaves must include a cycle. The distribution of the (true) number of leaves across our cell trajectory data sets in Figure 25 shows that the cumulative plots in Figure 24 may be greatly affected by the performance on linear and bifurcating topologies, making up the majority of the trajectories. Especially on bifurcating topologies, Slingshot seems to provide a better estimate on the number of leaves (Figure 26).

We also evaluated the performance of our method when we know the true number of leaves in our network (Figure 24). In case of less than two leaves, e.g., which may be the case if the ground-truth topology is a cycle, we still estimated the number of leaves as above. Note that Slingshot does not allow one to input the number of leaves as prior.

We observe that the performance of our method now increases in terms of all three of the HIM metric, F1 score, and correlation between important features, most significantly for the latter two. The lack of change in the correlation between geodesic distances may be explained by this metric not being affected that much by shorter branches. These branches are often excluded from our original inferred trajectory, as our estimate based on minimizing second-order-finite differences was generally too low (Figures 25 & 26).

The results summarized in this section show how our newly introduced optimization problem constraining the number of leaves (4) leads to a highly effective cell trajectory inference tool. If prior knowledge on the number of leaves in the ground-truth model is available, BCB outperforms Slingshot—the until now most accurately ranked state-of-the-art method for cell trajectory inference—in terms of the metrics introduced by Saelens et al. (2019). Without this prior knowledge, BCB is comparable to Slingshot by using our currently heuristic (elbow) estimator (Section 2.4.2). Slingshot is however unable to take the number of leaves as input. Hence, unlike Slingshot, BCB allows one to incorporate effective and independent machine learning models for estimating the number of leaves. This may eventually lead to a new best performing cell trajectory inference method, even when no prior knowledge is available. We will discuss this further in Section 5.

## 4. Discussion and Conclusion

Investigating and visualizing simplified graph-structured topologies in data is a core problem in many fields of science. Until now, there was no universal approach applicable to both general networks, as well as to point cloud data approaching such models. State-of-the-art

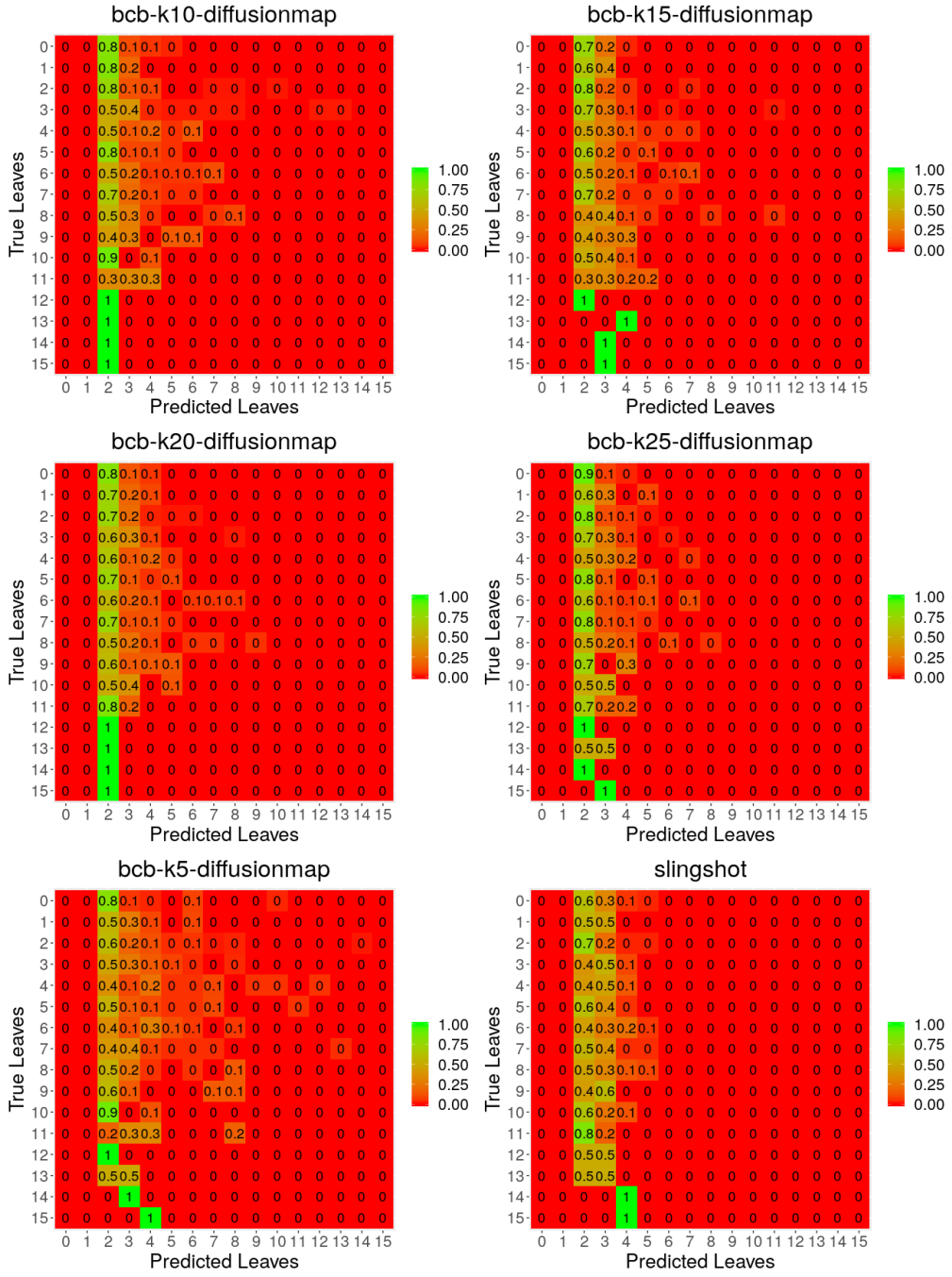



Figure 26: Normalized confusion matrices showing the true vs. the predicted number of leaves. Note that all methods output a trajectory with at least two leaves. In the case of 11 gene expression data sets, the number of neighbors $k = 5$ was too low to represent the connectedness of the true model through a neighborhood graph, resulting in fragmented backbones with many leaves (ranging from 17 to 104). These predictions were discarded from the corresponding confusion matrix for visualization purposes.

methods that focused on either one of them were computationally inefficient, sensitive to parameters, noise and outliers, were topologically biased, or did not generalize well. We solved these issues by introducing a simple but crucial intermediate step that showed to be highly beneficial throughout this entire paper. That is, designing a forest representation from which we may efficiently, robustly, and meaningfully mine topological substructures.

We introduced the *boundary coefficient* (BC), a coefficient that locates core topological structures well in many complex graphs (Vandaele et al., 2019b). Contrary to existing vertex measures, this coefficient is specifically designed for this purpose. Hence, the BC overcomes many difficulties faced with when dealing with this problem, such as applicability to complete networks, robustness to outliers, and the ability to deal with non-uniform branch lengths and curvature. Along with this, we provided extensive theoretical results concerning the computation of the BC, its robustness, as well as its relation to the ordinary local cluster coefficient. We showed that together, the BC and our introduced concept of $f$-pines (Vandaele et al., 2019b), provide effective forest representations in which many concepts of graph theory, such as longest paths and betweenness centrality, become both efficiently computable, and topologically meaningful.

Our newly introduced graph-optimization problem under the name of ***C****onstrained* ***L****eaves* ***O****ptimal sub****F****orest* (CLOF) is already interesting on a purely theoretical level. CLOF induces a nontrivial monotone submodular set function maximization problem subject to a cardinality constraint on tree graphs, for which a greedy approach provides an exact solution in polynomial time. Nevertheless, we also thoroughly illustrated the importance of this problem, as well as the effectiveness of its solution, for mining substructures through forest representations. All together, we provided a new method for *topological data analysis of graph-structured data*. We qualitatively and quantitatively demonstrated that our method leads to effective graph-structured models—balancing their size, goodness of fit, smoothness, and average commute time preservation—in many types of synthetic and real world data sets. These may be given weighted or unweighted graphs, point cloud data sets embedded in (non-)Euclidean metric spaces, or high-dimensional data sets.

Naturally, there is no single best method when it comes to extracting the backbone from a network. There will be cases where our approach will not be the best one as well. Examples are when the connectedness of our graph does coincide with the connectedness of its model, or in case of metric data, when the used proximity graph is not a truthful representation of the underlying model (Figure 23a). Nevertheless, our results convincingly show that we provided a very promising method across a broad spectrum of realistic applications.

## 5. Further Work

*Scalability.* Our method shows to scale well to thousands of nodes. At first sight, this may not be enough for many practical applications, such as large network embedding (Tang et al., 2015; Chen et al., 2018). Nevertheless, there are many practical examples, such as cell trajectory data, where identifying the underlying topology for graphs of this order remains an important problem. Furthermore, our method may scale to larger order graphs by optimizing or even parallelizing our current implementation to compute boundary coefficients, the most expensive part of our method. Algorithms for approximating these

coefficients may be investigated on a theoretical, probabilistic, and experimental level as well.

*Generalization.* One may increase the *local scope* of the boundary coefficient. I.e., the BC currently only averages over pairs of neighbors, but it may as well consider neighbors of neighbors, and so on. In this way, we may be able to effectively mine *topological skeletons* in non graph-structured data with a locally higher intrinsic dimension. An example of this would be the Swiss Roll from Section 3, where we increase the width of the manifold (along the $z$-axis), while also increasing the sparsity of the proximity graph.

We experimentally demonstrated that modeling our topology through a (sub)forest is not a severe limitation when it comes to cycles. These may be more efficiently discovered through persistent homology if our backbone provides a significant size reduction, while remaining spread out across the underlying topology. However, we have yet to provide an effective method for 'lifting' the holes found by means of topological persistence to our forest-structured backbone, i.e., closing the gaps corresponding to these holes. Persistent homology has also been connected to minimum spanning trees and their higher-dimensional analogues of minimum spanning acycles on a theoretical level (Kališnik et al., 2019). Connecting these results to CLOF may lead to new theoretically well-founded approximation algorithms for computing persistence (Silva and Carlsson, 2004; Oudot, 2015).

Another interesting direction is to investigate how our method generalizes to *directed graphs*. From a topological viewpoint, (anti-)arborescences could be very useful to interpret as possible backbone structures in directed graphs, as they have a natural and consistent flow defined on them. However, this may already become too restrictive. Unlike for unweighted graphs, such backbones may become non-existing if we require that each other node in the graph can be connected in some way to this backbone, through either a fixed or varying directionality. Furthermore, the BC may become undefined for nodes in weakly connected components, that are not strongly connected, and a different core measure may be more appropriate. This leads to many new theoretical and practical research questions.

*Cell Trajectory Inference.* In the case of cell trajectory inference, we noted that even the best performing cell trajectory methods, according to Saelens et al. (2019), often struggle in correctly identifying the backbone, including its number of leaves (Figures 24 & 26). Empirical investigation shows that for many data sets, branches are difficult to separate from each other due to a high amount of noise (even after a dimensionality reduction), or that branches which are relatively short according to some main trajectory are left undetected (intuitively, our 'elbow' in Figure 10b will occur too soon). However, we showed how that our method may highly benefit from a more effective leaf inference method for this purpose, as it can use this knowledge to increase its performance. A possible approach is to include a *topological inference* method. Carlsson (2014) presented a method under the name of *functional persistence*, that allows one to deduce 'flare'-like structures. These may be used to distinguish between a Y-shape and an X-shape (but not between an X-shape and an H-shape). In this way, we are able to construct *topological summaries*, i.e., persistence diagrams summarizing *topological features* of data sets, that have recently lead to effective topological inference and classification tools (Rieck and Leitte, 2017; Hofer et al., 2017; Rieck et al., 2019), which may aid us in quantifying or learning the number of leaves independent of our used method.

*Applications.* We only focused on applications within the field of topological data analysis throughout this paper. Correctly identifying the graph-structured model is the exact purpose of cell trajectory inference methods. For other graphs such as social networks, we also showed that we can meaningfully identify backbone structures, both on a qualitative and quantitative level. Our procedure provides a way to visualize or obtain insight into their underlying structure. Nevertheless, this is often not the end goal for these networks. Hence, a wide variety of new applications of backbones, such as community detection, subgroup discovery, and graph embeddings, is yet to be discovered. E.g., Chen et al. (2018) introduced a heuristic algorithm to iteratively simplify a graph, as to increase the performance of any existing graph embedding method through better initializations. Hence, initializing the embedding through a well-chosen backbone can lead to a graph embedding method that respects topological properties of the graph.

Finally, one may investigate how backbone extraction improves existing models for graph-structured data, and vice versa. E.g., *graph convolutional networks* (GCNs) have recently led to many new applications for graphs (Kipf and Welling, 2016a). On the one hand, prior knowledge of nodes near or on the core structure of the graph may enhance the ability to learn from graphs through better initializing a GCN. This is similar to the method described above for improving graph embeddings, which is also one of the many applications of GCNs. On the other hand, similar to how an initial dimensionality reduction improves the performance of cell trajectory inference, graph autoencoders (Kipf and Welling, 2016b) may serve as a tool to find a latent or denoised representation of the graph, prior to the final backbone extraction.

## Acknowledgments

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC Grant Agreement no. 615517, from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme, from the FWO (project no. G091017N, G0F9816N, 3G042220), and from the European Union's Horizon 2020 research and innovation programme and the FWO under the Marie Sklodowska-Curie Grant Agreement no. 665501.

We furthermore acknowledge Bo Kang and Bastian Rieck for helpful discussions, as well as Wouter Saelens and Robrecht Cannoodt for helping us setting up our large-scale cell trajectory inference experiments. Finally, we thank the anomynous reviewers for their many insightful comments and suggestions that had a noticeable positive impact on both the content and presentation of our work

## Appendix A. Supplementary Experiments

In this section, we provide further analysis of our method, by qualitatively discussing our obtained backbones for two of the graphs discussed in Section 3.

*Game of Thrones Network.* Figure A.1 shows the original Game of Thrones Network $G$, an LCC-pine in $G$, a forest-structured backbone $B$ mined from this LCC-pine, and the union

of the representative cycles for the two most persistent holes obtained through persistent homology of $(V(B), d_G)$. Nodes are colored according to their ground-truth community, i.e., the family they belong to.

We obtained a backbone $B$ with 10 leaves using 'standardized' vertex betweenness as cost function. I.e., the betweenness for each node was divided by the total cost of its corresponding connected component. Note that the cost of the smaller component in the backbone is relatively low according to its sum of (original) betweenness centralities, as it contains much fewer nodes than the larger component. Using non-standardized betweenness, one would either need to further increase the number of leaves for the smaller component to be represented in the backbone, or manually specify the number of leaves for each component.

Our backbone $B$ illustrates well how the ground-truth communities make up the entire topology of our graph. E.g., there is a branch corresponding to House Tyrell, to House Greyjoy, to House Martell, to House Targaryen, .... Not only is the backbone able to separate the communities well, but it also is able to infer how different communities are connected. E.g, House Stark, House Lannister, the Boltons and further on House Frey, are all connected through Sansa Stark. Eddark Stark (often referred to as Ned Stark), connects Sansa Stark (and through her the Lannisters, Boltons, and House Frey) to a branch of 'older' Starks, to House Tully, and to House Targaryen through Jon Snow.

Nevertheless, there are again some obvious 'gaps' in our backbone representation $B$ of $G$. E.g., Sansa Stark immediately connects to a branch of Lannisters through Tyrion Lannister, but to reach the other cluster of Lannisters, she must first travel through the Boltons, and then through House Frey, making it apparent that these groups of Lannisters form seperated communities. Another obvious gap is present in House Baratheon, as we first need to travel through House Lannister, House Stark, and House Targaryen before reconnecting this community. Though some smaller gaps seem to be present as well (and may also be identified using persistent homology), the ones discussed above correspond to the two most persistent holes one obtains by computing persistent homology of the metric space $(V(B), d_G)$ (Section 2.4.3). The union of the two corresponding representative cycles are shown in Figure A.1 as well. Note that these cycles overlap through a path between Jaime Lannister and Sansa Stark.

*NeurIPS Co-authorship Network.* We again applied our method to construct a tree-structured backbone with 5 leaves. The resulting tree graph is shown in Figure A.2.

As we did for the KDD network, we verified that the cores of our tree representations, i.e., the BC-pines, indeed correspond to meaningful core structures (Figure A.3). We observe that our method provides consistent results, as again, the BC-pine retract towards a core structure marking authors with a high number of citations, and who have been present very early in the considered conference.

## Appendix B. Algorithms and Computational Costs

In this section, we analyze the computational cost and provide basic pseudocode for the algorithms used in our method for topological data analysis of graph-structured data sets.

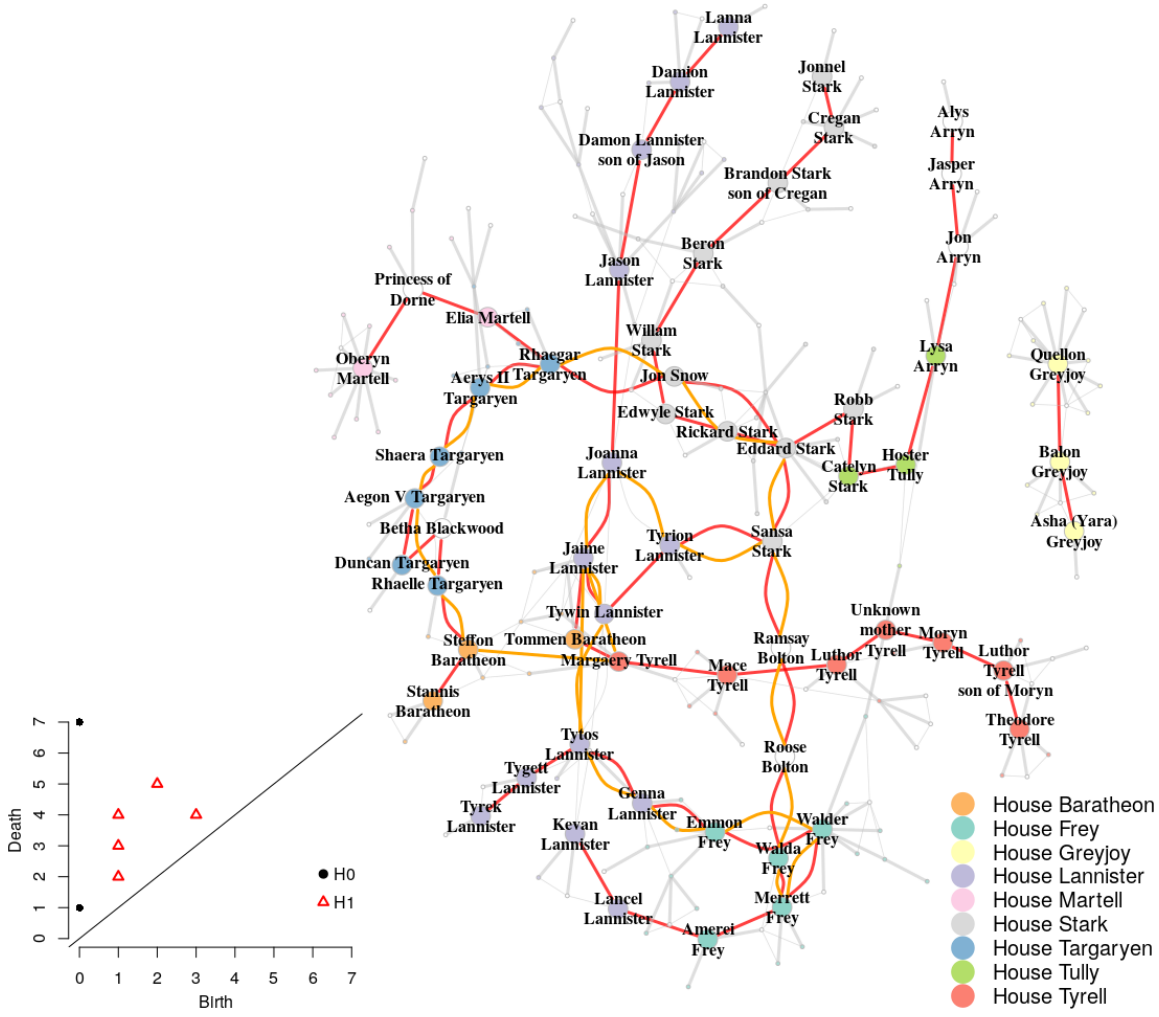



Figure A.1: A forest-structured backbone with 10 leaves (red edges) mined from the LCC-pine constructed from the Game of Thrones network (grey edges). Nodes are colored according to their ground-truth community, i.e., the House they belong to. White nodes correspond to members that are unassigned to any house. Persistent homology of the metric space induced by the original metric in $G$ on the nodes of the backbone is used to identify cycles missing in the backbone (orange edges).

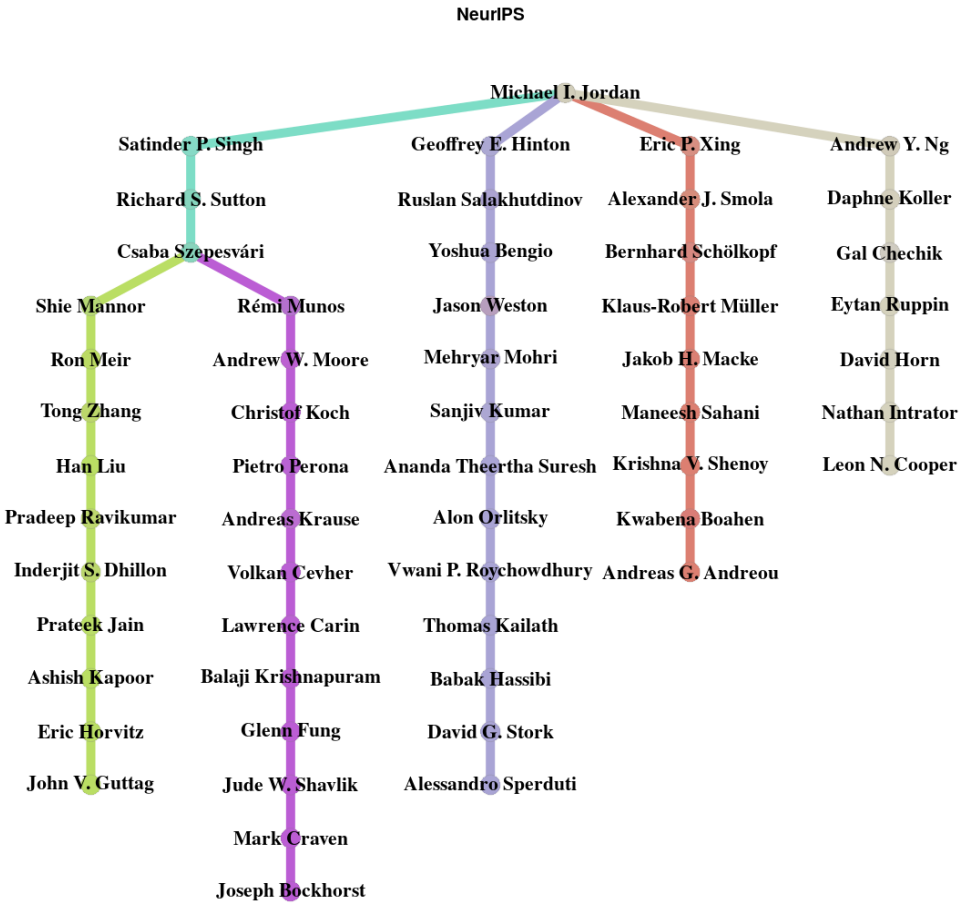



Figure A.2: A tree-structured backbone for the NeurIPS co-authorship network with 5 leaves. Nodes and edges are colored according to their closeness to one of the 6 branches.

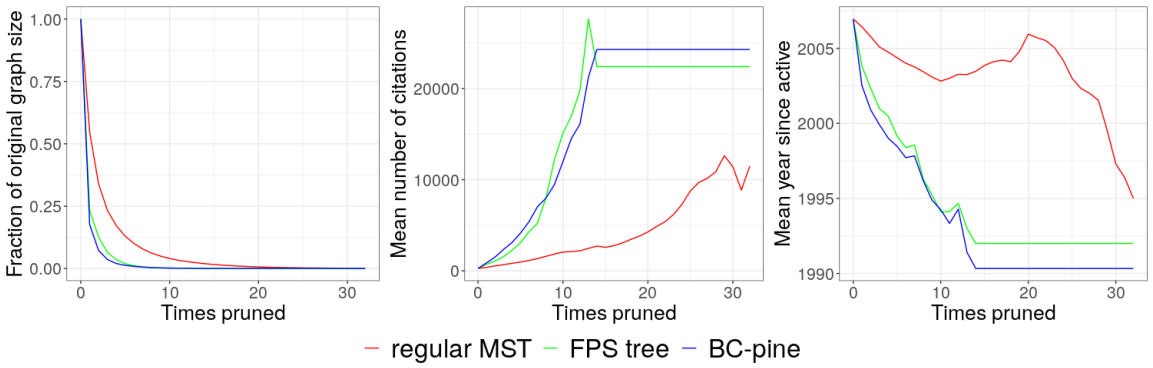



Figure A.3: Various measures after iteratively pruning the regular MST (red), the FPS tree (green), and the BC-pine (blue), for the NeurIPS network. (Left) Fraction of original graph size. (Middle) Average number of (NeurIPS) citations. (Right) Average year of first published (NeurIPS) paper.

**Input:** weighted graph $G$
**Output:** vector of boundary coefficients BC

```
hop1 = spam(hop_k_approx(G, k=1)) #compute hop-1-approximation
hop2 = spam(hop_k_approx(G, k=2)) #compute hop-2-approximation
BC = (apply.spam(hop1, 1, sum) * apply.spam((1 / hop1), 1, sum)
        - diag((1 / hop1) %*% hop2^2 %*% (1 / hop1)) / 2) / (degree(G)^2)
                                    #compute boundary coefficients
return(BC)
```

**Algorithm 1:** Computing the boundary coefficients through sparse matrices. 'spam' converts matrices to sparse matrix format. 'apply.spam($A$, 1, $f$)' applies the function $f$ to each row of the sparse matrix $A$. The operations '*' and '^2' denote element-wise multiplication, whereas '%*%' denotes matrix multiplication. '1 / $A$' returns the element-wise inverse of a sparse matrix $A$. 'diag($A$)' returns the diagonal of matrix $A$.

**Theorem B.1** *Let $G$ be an undirected, positively weighted graph, with $n$ nodes and $m$ edges. The boundary coefficients of all nodes in $V$ can be determined in $\mathcal{O}(n(m+n^{1.374}))$ time using Algorithm 1.*

**Proof** Obtaining the hop-$k$-approximations can be done $\mathcal{O}(n(m + n \log n))$ time through Dijkstra's algorithm. The matrix operations can be done in $\mathcal{O}(n^{2.374})$ time, which is the computational cost of matrix multiplication (Davie and Stothers, 2013). ■

**Remark B.2** *The computational cost stated in Theorem B.1 is worst-case, since it does not account for the sparsity of $\mathcal{H}_k(D)$, $k \in \{1, 2\}$. In practice, we note that working with sparse matrices significantly improves the efficiency of computing the boundary coefficients.*

**Theorem B.3** *Let $G$ be a graph with $n$ nodes and $m$ edges, and $f : V \to \mathbb{R}$. Then an $f$-pine of $G$ can be determined in $\mathcal{O}(\alpha(m, n)m)$ time using Algorithm 2. Here $\alpha$ is the classical functional inverse of the Ackermann function, which for all practical purposes may be considered a constant no greater than 4 (Sundblad, 1971).*

**Proof** The stated complexity is that of computing the regular minimum spanning forest (Chazelle, 2000). Reweighing the edges can be done in $\mathcal{O}(m)$ time. ■

**Theorem B.4** *Let $T = (V, E)$ be a tree graph with $n$ nodes (and $n - 1$ edges), and $f$ a real-valued function associating a positive cost to either each vertex or each edge of $T$. Then*

**Input:** graph $G$, vertex-valued cost function $f$ on $G$
**Output:** an $f$-pine of $G$

```
return(mst(G, weights=f[E(G)[,1]] + f[E(G)[,2]]))
```

**Algorithm 2:** Computing the $f$-pine. 'mst' is a function that computes the minimum spanning of a graph. The used weights follow from the result in Proposition 13.

**Input:** tree $T$, positive cost function $f$, upper bound $k \geq 2$ on leaves (standard $\infty$)
**Output:** A list L such that the subgraph of $T$ induced by the nodes in L[1:j] equals the solution to CLOF for $j \leq k$.

```
L = list(NULL, longest_path(T, cost=f))
        #the solution for k ∈ {0,1} is the empty graph by convention
        #the solution for k = 2 is the longest path according to f
current_leaves = 2 #number of leaves in the current solution
while current_leaves < k & L != T do
    v = which.max(distances(T, leaves(T), L), cost=f)[1]
            #determine furthest leaf from current solution according to f
    current_leaves += 1
    L[[current_leaves]] = path(from=L, to=v) #update the current solution
end
return(L)
```

**Algorithm 3: C**onstrained **L**eaves **O**ptimal sub**F**orest (CLOF) in trees. The function 'distances($G$, $U$, $V$)' returns the distances between the nodes in $U$ and $V$ of a graph $G$.

*for given $k \in \mathbb{N}_{\geq 2}$, an exact solution to (4) can be computed in $\mathcal{O}(n \min(k,l)l)$ time using Algorithm 3, where $l = |\{v \in V : \delta(v) = 1\}|$, i.e., $l$ denotes the number of leaves in $T$.*

**Proof** The advantage of working with tree (or forest) graphs, is that the path between a pair of nodes is always unique. The pairwise distance matrix $D_f$ between all leaves and all nodes in $T$, according to $f$, can be obtained in $\mathcal{O}(n \cdot l)$ time by using a bread-first-search for every leaf. For each of the no more than $\min(k,l)$ iterations of the algorithm described in Theorem 17, we can add the new path in $\mathcal{O}(n \cdot l)$ time. This is clear for the first iteration: search for the maximal entry of $D_f$ and add the corresponding path to the current (empty) structure. After the first path has been added, each consecutive leaf to be added can also be determined in $\mathcal{O}(n \cdot l)$ time, after which the path can be added in $\mathcal{O}(n)$ time. ■

**Theorem B.5** *Let $F = (V, E)$ be a forest graph with $n$ nodes, and suppose $f$ is a real-valued function, associating a positive cost to either each vertex or each edge of $F$. Given $k \in \mathbb{N}_{\geq 2}$, an exact solution to (4) can be computed in $\mathcal{O}\left(n + \beta_0(F)(\min(k, l_c)l_c n_c + l_c^{\beta_0(F)})\right)$ time using Algorithm 4, where $\beta_0(F)$ is the number of connected components in $F$, and $l_c$ and $n_c$, respectively, are the maximal number of leaves and nodes included in a connected component of $F$.*

**Proof** The components of $F$ can be determined in $\mathcal{O}(n)$ time. The complexity term $\mathcal{O}(\beta_0(F) \min(k, l_c)l_c n_c)$ comes from applying Algorithm 3 to each connected component of $F$. After this, we can iterate over all possible combinations $\left(k_1, \ldots, k_{\beta_0(F)}\right)$ of leaves for each connected component, to obtain the overall best corresponding sum of total costs, in $\mathcal{O}\left(\beta_0(F) l_c^{\beta_0(F)}\right)$ time. Note that the total cost from 2 up to $\min(k, l_c)$ for each component may also be stored during Algorithm 3, and does not need to be recomputed. ■

**Input:** forest $F$, positive cost function $f$, number of leaves $k$.
**Output:** A subgraph of $F$ corresponding to the solution of CLOF.

```
treeSols = lapply(components(F), function(T) Algorithm3(T, f, k))
        #apply Algorithm 3 to each connected component of F
        #return a list of all solutions (note that each solution is a list itself)
currentCost = -Inf #cost of the current best solution in F
for 0 ≤ l_1, ..., l_{β_0(F)} ≤ k do
    if(sum(l_1, ..., l_{β_0(F)}) > k) continue #skip if total number of leaves is > k
    thisCost = sum(cost(treeSols[[1]][1:l_1]), ..., cost(treeSols[[β_0(G)]][1:l_{β_0(F)}]))
            #evaluate the cost of the current potential solution according to f
            #the cost of each subtree can be stored during the execution
            #of Algorithm 3 for fast evaluation
    if thisCost > currentCost then
        currentCost = thisCost
        bestSol = subgraph(F, c(treeSols[[1]][1:l_1], ..., treeSols[[β_0(F)]][1:l_{β_0(F)}]))
                #update the current optimal solution
                #the solution is determined as the subgraph in F
                #induced by all partial solutions
    end
end
return(bestSol)
```

**Algorithm 4: C**onstrained **L**eaves **O**ptimal sub**F**orest (CLOF) in forests.

The additional exponential complexity term shows to be negligible in practice: often $\beta_0(F)$ is 1 or small. Theorem 21 allows us to significantly reduce $l_c$ in practice as well.

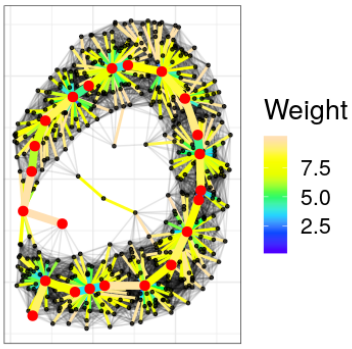
## Appendix C. Other Cost Functions for CLOF

In Section 2.4.1, we demonstrated the usefulness of betweenness centrality as vertex-valued cost function $g$ used for identifying a subforest by means of (4) for the purpose of topological data analysis of graph-structured data. Here, we discuss some other interesting choices.

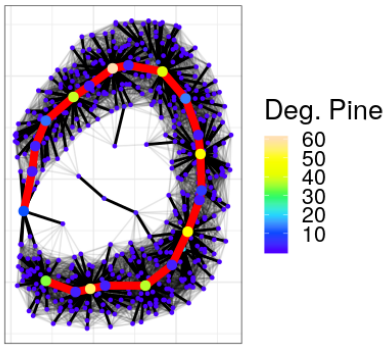
*The original edge weights.* As our backbone is meant to span the entire underlying topology of our given graph seen as a (shortest path) metric space, we may consider a longest or multiple longest paths in our $f$-pine to make up the backbone. E.g., the longest path shown in Figure C.4a identifies the correct underlying model, apart from the location of its leaves, chosen to be the furthest points in the local noise around the true leaves.

In terms of performance, this method is affected by the presence of outliers. E.g., the linear backbone in the pine shown in Figure C.4a is of similar length as the path in the pine that takes a turn to pass pass through the centered outlier. Though we got lucky in this case, we note that such 'interbranching regions of outliers' are often present in many practical examples, such as in cell trajectory data, as discussed by Saelens et al. (2019).

In terms of scalability, note that our $f$-pine generally has many leaves due to Theorem 11. If we take a look at the case where our original graph is connected, i.e., where the $f$-pine $F$ is a tree graph, then this implies that the number of leaves $l$ in $F$ may be of order $n$,




(a) Optimal subgraph with 2 leaves in a BC-pine according to the original edge weights (vertices in red). A single edge weight is however not representative for whether the corresponding edge is important for inclusion in the backbone or not, as it is not based on the resulting $f$-pine. Using our current implementation, the algorithm described in Theorem B.4 takes 56s to execute for the resulting BC-pine with 477 nodes, with no upper bound on $k$.




(b) Optimal subgraph with 2 leaves in a pruned BC-pine according to the vertex degree (edges in red). High degree nodes represent important nodes that should be included in the backbone, according to Theorem 21. The algorithm described in Theorem B.4 now takes 0.1s to execute with no upper bound on $k$, a significant improvement compared to using the original edge weights as cost. This illustrates the power of Theorem 21 when working with pines.

Figure C.4: Examples of solutions to (4) for cost functions other than vertex betweenness.

where $n$ is the number of nodes in $F$. If one fully grows the pine by through Algorithm 3, and consequently estimates an appropriate number of leaves $k$ as discussed in Section 2.4.2, Theorem B.4 implies that this computation may be close to cubic in $n$. Its memory usage will be close to squared in $n$, due to storing the pairwise distance matrix between leaves and all other nodes. Hence, apart from often not having a meaningful interpretation (Figure C.4a), allowing the inclusion of any leaf of the pine makes our current implementation for solving (4) difficult to scale to larger data sets.

*The degree of a vertex.* By Proposition 11, (locally) high degree nodes represent local minima of $f$ in an $f$-pine $F$. Given $f$ is a core measure where low values indicate core nodes, these are exactly the nodes where we want our backbone to pass through. Hence, we may use the vertex-valued degree function $g \equiv \delta_F$ for optimizing (4). The result for $g \equiv \delta_F$ is less affected by outliers due to their low density in the original graph.

This cost function $g$ is constant on leaves by definition. Hence, we may apply Theorem 21 to first prune $F$, often leading to a significant reduction of the graph size due to Proposition 11. In terms of Theorem B.4, this implies that both terms $l$ and $n$ decrease significantly, leading to a much better computation time, as well as storage cost (which is $\mathcal{O}(l \cdot n)$).

Figure C.4b shows the resulting solution of (4) in a pruned BC-pine. High degree nodes correspond to core nodes in the backbone, but not necessarily conversely. Though extending the two leaves of the linear backbone by connecting each one of them to an arbitrarily chosen neighboring leaf results in the optimal solution of (4) in the original pine (Theorem 21), we do not conduct this step as this will again introduce randomness to the choice of leaves.

## Appendix D. Theorems and Proofs

**Proof of Proposition 3** Using the equalities given in (1), we have

$$\mathrm{BC}(v) = \frac{\frac{1}{2}\left|\{u,w \in \mathcal{N}(v) : \{u,w\} \in E\}\right| - \left|\{u,w \in \mathcal{N}(v) : \{u,w\} \notin E \wedge u \neq w\}\right| + \delta(v)}{\delta(v)^2}.$$

Since

$$\left|\{u,w \in \mathcal{N}(v) : \{u,w\} \notin E \wedge u \neq w\}\right| = \delta(v)^2 - \left|\{u,w \in \mathcal{N}(v) : \{u,w\} \in E\}\right| - \delta(v),$$

we find

$$\begin{aligned}
\mathrm{BC}(v) &= \frac{\frac{3}{2}\left|\{u,w \in \mathcal{N}(v) : \{u,w\} \in E\}\right| + 2\delta(v) - \delta(v)^2}{\delta(v)^2} \\
&= \frac{\delta(v)-1}{\delta(v)}\left(\frac{3}{2}\frac{\sum_{u,w\in\mathcal{N}(v)} \mathbf{1}_{\{u,w\}\in E}}{\delta(v)(\delta(v)-1)}\right) + \frac{2-\delta(v)}{\delta(v)} \\
&= \frac{\delta(v)-1}{\delta(v)}\left(\frac{3}{2}\mathrm{LCC}(v) - 1\right) + \frac{1}{\delta(v)}.
\end{aligned}$$

Note that for graphs $G = (V, E)$ without loops, $\{u, v\} \in E \implies u \neq v$. ■

Proposition D.6 until Proposition D.9 present the other essential properties of generalizations of the LCC that the BC satisfies (Wang et al., 2017)—apart from its applicability to fully weighted networks—as discussed in Section 2.1.3.

**Proposition D.6** *(Weight-scale invariance, Vandaele et al. (2019b)). Let $G = (V, E)$ be an undirected graph without selfloops, with weighting function $\omega : E \to \mathbb{R}^+$. Let $\omega_\lambda : E \to \mathbb{R} : \{u, v\} \mapsto \lambda\omega(\{u, v\})$ for a global scale factor $\lambda > 0$. Then for every $v \in V$, $BC_\lambda(v) = BC(v)$, where $BC_\lambda(v)$ equals the boundary coefficient of $v$ for the new weighting function $\omega_\lambda$.*

**Proof** By multiplying each edge weight with a global scale factor $\lambda > 0$, the shortest path distance between any two nodes is also scaled by the same factor $\lambda$. Hence, the stated equality easily follows from Definitions 1 & 2. ■

**Lemma D.7** *Let $G = (V, E)$ be an undirected graph, with weighting function $\omega : E \to \mathbb{R}^+$. Suppose that $\mathcal{E}$ denotes an additive noise matrix, which defines a new weighting function $\omega_\epsilon : E \to \mathbb{R}^+ : \{u, v\} \mapsto \omega(\{u, v\}) + \mathcal{E}_{u,v}$. Then the following statements are valid:*

1. *$\lim_{\|\mathcal{E}\|_\infty \to 0} \|d - d_\epsilon\|_\infty = 0$, where $d$ denote the shortest path metric on $V$ according to $\omega$, and $d_\epsilon$ according $\omega_\epsilon$, where we follow the convention that $d(u, v) - d_\varepsilon(u, v) = 0$ if $u$ and $v$ lie in different connected components of $G$;*
2. *for any $u, v, w \in V$ belonging to the same connected component of $G$, with $u \neq v \neq w$, $\lim_{\epsilon\to 0} \mathcal{T}_\epsilon(u, v, w) = \mathcal{T}(u, v, w)$, where $\mathcal{T}(u, v, w)$ denotes transmissivity of $v$ for $u$ and $w$ according to $\omega$, and $\mathcal{T}_\varepsilon(u, v, w)$ according to $\omega_\epsilon$.*

**Proof** 1. Suppose $u, v \in V$, and $P$ is a shortest path from $u$ to $v$ according to $\omega$ with length $d(u, v)$. Then the length of the same path $P$ according to $\omega_\epsilon$ is bounded from above by $d(u, v) + |E(P)| \cdot \|\mathcal{E}\|_\infty \leq d(u, v) + |E| \cdot \|\mathcal{E}\|_\infty$, where $|E(P)|$ denotes the number of edges on $P$. Since the length of the shortest path from $u$ to $v$ according to $\omega_\epsilon$ is at most the length of $P$ according to $\omega_\epsilon$, it holds that $d_\epsilon(u, v) \leq d(u, v) + |E|\|\mathcal{E}\|_\infty$. Analogously, we have $d(u, v) \leq d(u, v)_\epsilon + |E| \cdot \|\mathcal{E}\|_\infty$, so that $\lim_{\|\mathcal{E}\|_\infty \to 0} |d_\epsilon(u, v) - d(u, v)| = 0$. As $u$ and $v$ were chosen arbitrarily, the stated theorem holds.

2. This easily follows from Proposition D.7.1. and Definition 1. ■

**Proposition D.8** *(Continuity). Let $G = (V, E)$ be an undirected graph without selfloops, with weighting function $\omega : E \to \mathbb{R}^+$. Suppose $\omega_\epsilon$ is a new weighting function on $E$ that differs in exactly one edge $e \in E$ by an additive constant $\epsilon \in \mathbb{R}$, i.e., $\omega_\epsilon(e) = \omega(e) + \epsilon \in \mathbb{R}^+$, and $\omega_\varepsilon|_{E\setminus\{e\}} \equiv \omega|_{E\setminus\{e\}}$. If $BC_\epsilon$ denotes the boundary coefficient function according to the new weighting function $\omega_\epsilon$, then $\lim_{\epsilon\to 0} BC_\epsilon(v) = BC(v)$ for all $v \in V$ with $\delta(v) > 0$.*

**Proof** This easily follows from Lemma D.7 and Definition 2. ■

The problem in the ordinary formulation of the 'robustness to noise' property stated by Wang et al. (2017), is that the error-value $\Delta(\mathcal{E})$ is not well-defined when a node has a boundary coefficient of 0. Hence, we consider a slight variant below.

**Proposition D.9** *(Robustness to noise, Vandaele et al. (2019b)). Let $G = (V, E)$ be an undirected graph, with weighting function $\omega : E \to \mathbb{R}^+$. Suppose that $\mathcal{E}$ denotes an additive noise matrix, which defines a new weighting function $\omega_\epsilon : E \to \mathbb{R}^+ : \{u, v\} \mapsto \omega(\{u, v\}) + \mathcal{E}_{u,v}$. If $f : \mathbb{R} \to \mathbb{R}$ is any continuous function such that $f \circ BC(v) \neq 0$ for any $v \in V$, then*

$$\Delta(\mathcal{E}) \coloneqq \frac{100}{|V|} \sum_{v\in V} \left| \frac{f \circ BC_\epsilon(v) - f \circ BC(v)}{f \circ BC(v)} \right| \xrightarrow[\|\mathcal{E}\|_\infty \to 0]{} 0 \,,$$

*where $BC_\epsilon(v)$ equals the boundary coefficient of $v$ for the new weighting function $\omega_\epsilon$.*

**Proof** This follows from Lemma D.7 and $f$ being continuous. ■

**Remark D.10** *The fact that we consider our variant to robustness of noise an equally important property for our method for topological data analysis of graph-structured data, is due to Proposition 12 stating that BC-pines are invariant under affine transformations of the BC with a positive scaling factor. The BC may always be mapped to an interval excluding 0 using such continuous transformation.*

Note that the theoretical rate of convergence of $\Delta(\mathcal{E})$ in Theorem D.9 depends on $|E|$ (see the proof of Lemma D.7.1.), a consequence of allowing arbitrary long paths (in terms of number of edges) between the endpoints of a triple adjacent to a node, to compute the BC.

The following proposition justifies the naming 'hop-$k$-approximation', introduced in Section 2.1.4.

**Proposition D.11** *Let $G = (V, E)$ be a connected, undirected, positively weighted graph, and $D$ the matrix of pairwise shortest path distances between the nodes of $G$. Then the following statements are valid:*

1. $\mathcal{H}_0(D) = (0)_{u,v \in V}$ ;
2. $\mathcal{H}_{diam_{unw}(G)}(D) = D$ ;
3. *for any* $k, l \in \mathbb{N}$, $k < l \implies \|D - \mathcal{H}_l(D)\|_\infty \leq \|D - \mathcal{H}_k(D)\|_\infty$ ;

*where $diam_{unw}(G)$ denotes the unweighted diameter of $G$.*

**Proof** 1. If $u$ can be reached from $v$ in 0 steps, then $u = v$, which implies that $D_{u,v} = 0$.
2. For any nodes $u, v \in V$, $v$ can be reached from $u$ within $\mathrm{diam}_{\mathrm{unw}}(G)$ steps.
3. This is clear from the definition of $\mathcal{H}_k(D)$. ■

**Proof of Theorem 8** For $v \in V$, with $\delta(v) > 0$, we have

$$\begin{aligned}\delta(v)^2 \mathrm{BC}(v) &= -\sum_{u,w \in \mathcal{N}(v)} \mathcal{T}(u,v,w) = \sum_{u,w \in \mathcal{N}(v)} \left( \frac{D_{u,v}^{\odot 2} + D_{v,w}^{\odot 2} - D_{u,w}^{\odot 2}}{2 D_{u,v} D_{v,w}} \right) \\ &= \sum_{u,w \in \mathcal{N}(v)} \frac{D_{u,v}}{2 D_{v,w}} + \sum_{u,w \in \mathcal{N}(v)} \frac{D_{v,w}}{2 D_{u,v}} - \sum_{u,w \in \mathcal{N}(v)} \frac{D_{u,w}^{\odot 2}}{2 D_{u,v} D_{v,w}}.\end{aligned}$$

The first two summations are equal by a change of variables. Hence, we find

$$\begin{aligned}\delta(v)^2 \mathrm{BC}(v) &= \sum_{u,w \in \mathcal{N}(v)} \frac{D_{u,v}}{D_{v,w}} - \sum_{u,w \in \mathcal{N}(v)} \frac{D_{u,w}^{\odot 2}}{2 D_{u,v} D_{v,w}}. \\ &= \sum_{u \in \mathcal{N}(v)} D_{u,v} \sum_{w \in \mathcal{N}(v)} \frac{1}{D_{v,w}} - \frac{1}{2} \sum_{u,w \in \mathcal{N}(v)} \frac{1}{D_{u,v}} D_{u,w}^{\odot 2} \frac{1}{D_{v,w}} \\ &= \sum_{u \in V} \mathcal{H}_1(D)_{u,v} \sum_{u \in V} \mathcal{H}_1^{\odot -1}(D)_{u,v} - \frac{1}{2} \sum_{u,w \in V} \mathcal{H}_1^{\odot -1}(D)_{u,v} \mathcal{H}_2^{\odot 2}(D)_{u,w} \mathcal{H}_1^{\odot -1}(D)_{v,w} \\ &= \sum_{u \in V} \mathcal{H}_1(D)_{u,v} \sum_{u \in V} \mathcal{H}_1^{\odot -1}(D)_{u,v} - \frac{1}{2} \mathrm{diag}\left( \mathcal{H}_1^{\odot -1}(D) \mathcal{H}_2^{\odot 2}(D) \mathcal{H}_1^{\odot -1}(D) \right)_v,\end{aligned}$$

which concludes the proof. ■

**Proof of Proposition 11** Assume $u \in V$ with $\delta_G(u) > 0$. If $\{u, v\} \notin E(T)$ for every $v \in \arg\min\{f(w) : w \in \mathcal{N}_G(u)\}$, then choose such $v$. Let $P = (u = x_0, x_1, \ldots, x_k = v)$ be the unique path from $u$ to $v$ in $T$. Since $\{u, x_1\} \in E(T)$, $f(x_1) > f(v)$, and we can replace $\{u, x_1\}$ by $\{u, v\}$ to obtain a tree attaining a lower cost as expressed by (3). ■

**Proof of Proposition 12** We have

$$\sum_{v\in V}\delta_F(v)g(v) = a\sum_{v\in V}\delta_F(v)f(v) + b\sum_{v\in V}\delta_F(v) = a\sum_{v\in V}\delta_F(v)f(v) + 2b(|V| - \beta_0(G)),$$

where $\beta_0(G)$ denotes the number of connected components of $G$. This follows from:

- the sum of degrees over all vertices of a graph is twice its number of edges;
- the number of edges in any forest graph containing $n$ vertices, is $n$ minus its number of connected components;
- by Definition 9, the number of connected components of a graph equals the number of connected components of a spanning forest of that graph.

As $a > 0$ and the second term in the right hand side is independent of $F$, minimization of the left hand side over all spanning forests is equivalent to minimizing $\sum_{v\in V}\delta_F(v)f(v)$. ■

**Proof of Proposition 13** It holds that

$$\sum_{v\in V}\delta_F(v)f(v) = \sum_{\{u,v\}\in E(F)}(f(u) + f(v)),$$

where $E(F)$ denotes the edges in the subgraph $F$ of $G$. ■

**Proof of Theorem 16** First observe that for any given set of leaves $L = \{l_1, \ldots, l_k\}$ of $T$, there is a unique subtree $T_L$ of $T$ that contains exactly the same set as leaves. Hence, if $V \supseteq \mathcal{L}$ is the set of all leaves of $T$, we may define $\tilde{f} : 2^{\mathcal{L}} \to \mathbb{R}^+ : L \mapsto f(T_L)$, where $f(T_L)$ is the cost of the subtree $T_L$ as defined in Definition 14. Suppose now that $L \subseteq L' \subsetneq \mathcal{L}$, and take any $l \in \mathcal{L}\backslash\{L'\}$. Observe that $T_L$ is a subtree of $T_{L'}$, for which the unique path from $l$ to $T_L$ includes the unique path from $l$ to $T'_L$. Hence, $\tilde{f}(L\cup\{l\}) - \tilde{f}(L) \geq \tilde{f}(L'\cup\{l\}) - \tilde{f}(L')$, i.e., $\tilde{f}$ is a submodular set function on $\mathcal{L}$. Furthermore $\tilde{f}$ is clearly monotone, as $L \subseteq L' \subseteq \mathcal{L} \implies \tilde{f}(L) \leq \tilde{f}(L')$. Finally, by definition of $\tilde{f}$, (4) is equivalent to maximizing $\tilde{f}$ subject to the cardinality constrained given by $k$. ■

**Proof of Theorem 17** We will first assume that $f$ is an edge-valued function. The proof goes by induction on the number of leaves $k$. The claim is trivially valid for $k = 2$ leaves, or if $k$ is at least the number of leaves in $T$. Suppose now that $2 < k < |\{v \in V : \delta_T(v) = 1\}|$, and that the greedy algorithm iteratively added the paths $P_1, \ldots, P_k$, in this order, resulting in the subtree $T^k_{gr}$. Suppose an optimal subtree $T^k_{opt}$ with at most $k$ leaves achieves a cost strictly higher than the cost of $T^k_{gr}$. Note that both $T^k_{opt}$ and $T^k_{gr}$ exactly contain $k$ leaves. By the induction hypothesis, the subtree $T^{k-1}_{gr}$ consisting of the paths $P_1, \ldots, P_{k-1}$, is the optimal subtree of $T$ with $k-1$ leaves. Hence, for every subset of size $k-1$ of the $k$ leaves $\{l_1, \ldots, l_k\}$ of $T^k_{opt}$, the cost of the tree induced by this subset is at most the cost of $T^{k-1}_{gr}$. As, by assumption, the cost of $T^k_{opt}$ is strictly higher than the cost of $T^k_{gr}$, this implies that for every $1 \leq i \leq k$, the cost of the path from $l_i$ to the tree induced by the leaves $\{l_1, \ldots, l_{i-1}, l_{i+1}, \ldots, l_k\}$ is strictly higher than the cost of $P_k$, which we will denote by $f(P_k)$. We now consider two possible cases.

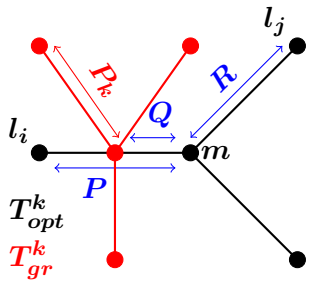
1. *There exists a leaf $l_i$ of $T_{opt}^k$, such that the path $P$ that connects $l_i$ to the tree induced by the leaves $\{l_1, \ldots, l_{i-1}, l_{i+1}, \ldots, l_k\}$ is edge-disjoint from $T_{gr}^{k-1}$.*

   The endpoint of $P$ different from $l_i$ is a multifurcation point $m$ of $T_{opt}^k$. If $m \in V(T_{gr}^{k-1})$, then since $f(P) > f(P_k)$, the algorithm would have chosen to add $P$ instead of $P_k$ to obtain $T_{gr}^k$, a contradiction, so that $m \notin V(T_{gr}^{k-1})$. Let $Q$ be the unique path from $m$ to $T_{gr}^{k-1}$. If $P$ is edge-disjoint from $Q$, then $P + Q$ would have been chosen instead of $P_k$ by the greedy algorithm, so that $P$ and $Q$ partially overlap. Now let $l_j$ be any leaf in $T_{opt}^k$ different from $l_i$. The path $R$ from $m$ to $l_j$ in $T_{opt}^k$ is now both edge- and vertex-disjoint from $T_{gr}^{k-1}$ (see Figure D.5a for a sketch of this case). Furthermore $f(R + Q) > f(R) > f(P_k)$, and the greedy algorithm would have chosen to add the path $R + Q$ instead of $P_k$, a contradiction.

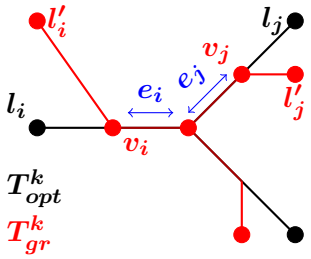
2. *For every leaf $l_i$ of $T_{opt}^k$, the path $P$ that connects $l_i$ to the tree induced by the leaves $\{l_1, \ldots, l_{i-1}, l_{i+1}, \ldots, l_k\}$ contains edges from $T_{gr}^{k-1}$.*

   Consider an arbitrary leaf $l_i$ of $T_{opt}^k$, and let $v_i$ be the point closest to $l_i$ on the first edge $e_i$ on the path from $l_i$ to the tree induced by $\{l_1, \ldots, l_{i-1}, l_{i+1}, \ldots, l_k\}$, that is also contained in $E(T_{gr}^{k-1})$. Note that possibly $l_i = v_i$. Now let $l_i'$ be any leaf of $T_{gr}^{k-1}$ that is reachable from $v_i$ after removing $e_i$ from in $T_{gr}^{k-1}$. If $l_i \neq l_j$ are both leaves of $T_{opt}^k$, then $l_i' \neq l_j'$. To see this, observe that for $l_i \neq l_j$, by definition of $v_i$, we have $v_i \neq v_j$, and that the path from $v_i$ to $v_j$ in $T_{opt}^k$ must go through both $e_i$ and $e_j$. As this path is unique in $T$, it is also fully contained in $T_{gr}^{k-1}$. Hence, the path $v_i \to v_j \to l_j'$ is the unique path from $v_i$ to $l_j'$ in $T_{gr}^{k-1}$, and passes through $e_i$. Hence, $l_j'$ is not reachable from $v_i$ after removing $e_i$ from $T_{gr}^{k-1}$. As such, we obtain an injection $l_i \mapsto l_i'$ of the $k$ leaves in $T_{opt}^k$ to the $k - 1$ leaves in $T_{gr}^{k-1}$, a contradiction. Note that this case is simply not possible, independent of the used cost function. We provided a sketch for the closest possible case in Figure D.5b.

Since both cases lead to a contradiction, we conclude that $T_{opt}^k$ cannot achieve a cost strictly higher than $T_{gr}^k$. This implies that $T_{gr}^k$ is an exact solution to (4).




(a) Sketch for the first case in the proof of Theorem 17. The cost of $Q + R$ must be bounded from above by the cost of $P_k$ due to the definition of the greedy algorithm.




(b) Sketch for the second case in the proof of Theorem 17. There is a systematically defined injection from the leaves of $T_{opt}^k$ to the leaves of $T_{gr}^k$.

Figure D.5: Sketches for the different cases in the proof of Theorem 17.

For $f$ a vertex-valued function, the proof goes analogous to the proof of Theorem 17. However, the increase in cost of adding a new path $P$ to the current tree is now the sum of the cost over all vertices in $P$, minus the cost of the connecting node. The only resulting change we need to apply in the proof of Theorem 17, is that instead of writing '$f(R+Q) > f(R) > f(P_k)$' in the first considered case, we now write '$f(R+Q) > f(P_k)$', as $f(R)$ may not be well-defined according to this convention. ■

**Proof of Theorem 21** Let $S'$ be a solution to (4) for $T'$. Note that $S'$ has at least two leaves. Extending all leaves of $S'$ by an arbitrarily chosen neighboring leaf in $T$, and consecutively adding leaves until $S'$ has $\min(k, \{v \in V : \delta_T(v) = 1\})$ leaves, results in a solution to (4) in $T$. ■

## References


Mridul Aanjaneya, Frederic Chazal, Daniel Chen, Marc GLisse, Leonidas Guibas, and Dmitriy Morozov. Metric graph reconstruction from noisy data. *International Journal of Computational Geometry and Applications*, 22(04):305–325, 2012.

Waleed Abu-Ain, Siti Norul Huda Sheikh Abdullah, Bilal Bataineh, Tarik Abu-Ain, and Khairuddin Omar. Skeletonization algorithm for binary images. *Procedia Technology*, 11: 704 – 709, 2013. ISSN 2212-0173. 4th International Conference on Electrical Engineering and Informatics, ICEEI 2013.

Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory — ICDT 2001*, pages 420–434, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44503-6.

Leman Akoglu, Jilles Vreeken, Hanghang Tong, Duen Horng Chau, Nikolaj Tatti, and Christos Faloutsos. Mining connection pathways for marked nodes in large graphs. In *Proceedings of the 2013 SIAM International Conference on Data Mining, SDM 2013*, pages 37–45. Siam Society, 2013. ISBN 9781611972627.

Y.P. Aneja and K.P.K. Nair. Location of a tree shaped facility in a network. *INFOR: Information Systems and Operational Research*, 30(4):319–324, 1992.

Pasquale Avella, Maurizio Boccia, Antonio Sforza, and Igor Vasil'Ev. A branch-and-cut algorithm for the median-path problem. *Computational Optimization and Applications*, 32(3):215–230, Dec 2005. ISSN 1573-2894.

M. Barthélemy. Betweenness centrality in large complex networks. *The European Physical Journal B*, 38(2):163–168, Mar 2004. ISSN 1434-6036.

Punam Bedi and Chhavi Sharma. Community detection in social networks. *WIREs Data Mining and Knowledge Discovery*, 6(3):115–135, 2016.

Robrecht Cannoodt, Wouter Saelens, and Yvan Saeys. Computational methods for trajectory inference from single-cell transcriptomics. *European Journal of Immunology*, 46(11): 2496–2506, nov 2016. ISSN 00142980.

Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46 (2):255–308, jan 2009. ISSN 0273-0979.

Gunnar Carlsson. Topological pattern recognition for point cloud data. *Acta Numerica*, 23: 289–368, 2014.

Nicholas J. Cavanna, Mahmoodreza Jahanseir, and Donald R. Sheehy. A geometric perspective on sparse filtrations. In *Proceedings of the Canadian Conference on Computational Geometry*, 2015.

Bernard Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. *Journal of the ACM (JACM)*, 47(6):1028–1047, 2000.

Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. Harp: Hierarchical representation learning for networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Ena Choi, Nicholas A. Bond, Michael A. Strauss, Alison L. Coil, Marc Davis, and Christopher N. A. Willmer. Tracing the filamentary structure of the galaxy distribution at $z \sim 0.8$. *Monthly Notices of the Royal Astronomical Society*, 406(1):320–328, jul 2010. ISSN 00358711.

T.G. Crainic and G. Laporte. *Fleet Management and Logistics*. Centre for Research on Transportation. Springer US, 1998. ISBN 9780792381617.

A. Davie and AJ Stothers. Improved bound for complexity of matrix multiplication. *Proceedings of the Royal Society of Edinburgh: Section A Mathematics*, 143, 04 2013.

Leen De Baets, Sofie Van Gassen, Tom Dhaene, and Yvan Saeys. Unsupervised trajectory inference using graph mining. In *International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics*, pages 84–97. Springer, 2015.

Brittany Terese Fasy and Bei Wang. Exploring persistent local homology in topological data analysis. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6430–6434, 2016.

Brittany Terese Fasy, Jisu Kim, Fabrizio Lecci, and Clément Maria. Introduction to the R package TDA. *arXiv preprint arXiv:1411.1830*, 2014.

F. Fouss, A. Pirotte, J. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, March 2007. ISSN 2326-3865.

Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1990. ISBN 0716710455.

Kathryn Garside, Robin Henderson, Irina Makarenko, and Cristina Masoller. Topological data analysis of high resolution diabetic retinopathy images. *PloS one*, 14(5):e0217413, 2019.

Robert Ghrist. Barcodes: The Persistent Topology of Data. *Bulletin (New Series) of the American Mathematical Society*, 45(107):61–75, 2008.

Alexander N Gorban and Andrei Y Zinovyev. Principal graphs and manifolds. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 28–59. IGI Global, 2010.

Per Hage and Frank Harary. Eccentricity and centrality in networks. *Social Networks*, 17 (1):57–63, 1995.

John A Hartigan. *Clustering algorithms*. John Wiley & Sons, Inc., 1975.

Trevor Hastie and Werner Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

Allen Hatcher. *Algebraic topology*. Cambridge University Press, 2002. ISBN 0521795400.

Songtao He, Favyen Bastani, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, and Sam Madden. Roadrunner: improving the precision of road network inference from gps trajectories. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 3–12, 2018.

Christoph Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl. Deep learning with topological signatures. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 1633–1643, USA, 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4.

Yingpeng Hu, Kaixi Zhang, Jing Yang, and Yanghui Wu. Application of hierarchical facility location-routing problem with optimization of an underground logistic system: A case study in china. *Mathematical Problems in Engineering*, 2018:1–10, 09 2018.

Dakai Jin, Krishna S. Iyer, Cheng Chen, Eric A. Hoffman, and Punam K. Saha. A robust and efficient curve skeletonization algorithm for tree-like objects using minimum cost paths. *Pattern Recognition Letters*, 76:32 – 40, 2016. ISSN 0167-8655. Special Issue on Skeletonization and its Application.

Sara Kališnik, Vitaliy Kurlin, and Davorin Lešnik. A higher-dimensional homologically persistent skeleton. *Advances in Applied Mathematics*, 102:113 – 142, 2019. ISSN 0196-8858.

Tae Kim, Timothy Lowe, James Ward, and Richard Francis. A minimum length covering subgraph of a network. *Annals of Operations Research*, 18:245–259, 12 1989.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

Douglas Klein. Centrality measure in graphs. *Journal of Mathematical Chemistry*, 47: 1209–1223, 05 2010.

Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability*, 3: 71–104, 01 2011.

Andrea S. Lapaugh and Ronald L. Rivest. The subgraph homeomorphism problem. *Journal of Computer and System Sciences*, 20(2):133 – 149, 1980. ISSN 0022-0000.

Patrick Medina and R Doerge. Statistical Methods in Topological Data Analysis for Complex, High-Dimensional Data. *Annual Conference on Applied Statistics in Agriculture*, 2015.

Juan Mesa and T. Brian Boffey. A review of extensive facility location in networks. *European Journal of Operational Research*, 95:592–603, 12 1996.

Shubhadip Mitra, Priya Saraf, and Arnab Bhattacharya. Tips: mining top-k locations to minimize user-inconvenience for trajectory-aware services. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

HDK Moonesignhe and Pang-Ning Tan. Outlier detection using random walks. In *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, pages 532–539. IEEE, 2006.

Michael Moor, Max Horn, Bastian Rieck, and Karsten Borgwardt. Topological autoencoders. *arXiv preprint arXiv:1906.00722*, 2019.

Monica Nicolau, Arnold J. Levine, and Gunnar Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108(17):7265–7270, apr 2011. ISSN 0027-8424.

Nina Otter, Mason A. Porter, Ulrike Tillmann, Peter Grindrod, and Heather A. Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6(1):17, Aug 2017. ISSN 2193-1127.

Steve Y Oudot. *Persistence theory: from quiver representations to data analysis*, volume 209. American Mathematical Society Providence, 2015.

Peng Si Ow and Thomas E. Morton. Filtered beam search in scheduling. *International Journal of Production Research*, 26(1):35–62, 1988.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Nearest neighbors in high-dimensional data: The emergence and influence of hubs. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 865–872, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1.

Michael B. Richey. Optimal location of a path or tree on a network with cycles. *Networks*, 20(4):391–407, 1990.

B. Rieck and H. Leitte. Persistent homology for the evaluation of dimensionality reduction schemes. *Computer Graphics Forum*, 34(3):431–440, 2015.

Bastian Rieck and Heike Leitte. Agreement analysis of quality measures for dimensionality reduction. In Hamish Carr, Christoph Garth, and Tino Weinkauf, editors, *Topological Methods in Data Analysis and Visualization IV*, pages 103–117, Cham, 2017. Springer International Publishing. ISBN 978-3-319-44684-4.

Bastian Rieck, Christian Bock, and Karsten Borgwardt. A persistent weisfeiler-lehman procedure for graph classification. In *International Conference on Machine Learning*, pages 5448–5458, 2019.

Abbas Haider Rizvi, Pablo G. Cámara, Elena K. Kandror, Tom Roberts, Ira Schieren, Tom Maniatis, and Raúl Rabadán. Single-cell topological rna-seq analysis reveals insights into cellular differentiation and development. In *Nature Biotechnology*, 2017.

Afshin Sadeghi and Holger Fröhlich. Steiner tree methods for optimal sub-network identification: An empirical study. *BMC bioinformatics*, 14:144, 04 2013.

Wouter Saelens, Robrecht Cannoodt, Helena Todorov, and Yvan Saeys. A comparison of single-cell trajectory inference methods. *Nature Biotechnology*, 37:1, 04 2019.

Vin Silva and Gunnar Carlsson. Topological estimation using witness complexes. *Proc. Sympos. Point-Based Graphics*, 06 2004.

Nikhil Singh, Heather D. Couture, J. S. Marron, Charles Perou, and Marc Niethammer. Topological descriptors of histology images. In Guorong Wu, Daoqiang Zhang, and Luping Zhou, editors, *Machine Learning in Medical Imaging*, pages 231–239, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10581-9.

Kelly Street, Davide Risso, Russell Fletcher, Diya Das, John Ngai, Nir Yosef, Elizabeth Purdom, and Sandrine Dudoit. Slingshot: Cell lineage and pseudotime inference for single-cell transcriptomics. *BMC Genomics*, 19, 12 2018.

Yngve Sundblad. The ackermann function. a theoretical, computational, and formula manipulative study. *BIT Numerical Mathematics*, 11(1):107–119, Mar 1971. ISSN 1572-9125.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1067–1077, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-3469-3.

Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. ISSN 0036-8075.

Ryuhei Uehara and Yushi Uno. Efficient algorithms for the longest path problem. In Rudolf Fleischer and Gerhard Trippen, editors, *Algorithms and Computation*, pages 871–883, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-30551-4.

Robin Vandaele, Tijl De Bie, and Yvan Saeys. Local topological data analysis to uncover the global structure of data approaching graph-structured topologies. In Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, and Georgiana Ifrim, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 19–36, Cham, 2019a. Springer International Publishing.

Robin Vandaele, Yvan Saeys, and Tijl De Bie. The boundary coefficient: a vertex measure for visualizing and finding structure in weighted graphs. In *Proceedings of the 15th International Workshop on Mining and Learning with Graphs (MLG)*, 2019b.

Ulrike Von Luxburg and Morteza Alamgir. Density estimation from unweighted k-nearest neighbor graphs: a roadmap. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 225–233. Curran Associates, Inc., 2013.

Bei Wang, Brian Summa, Valerio Pascucci, and Mikael Vejdemo-Johansson. Branching and circular features in high dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 17:1902–1911, 2011.

Suyi Wang, Xu Li, Partha Mitra, and Yusu Wang. Topological skeletonization and tree-summarization of neurons using discrete morse theory. *arXiv preprint arXiv:1805.04997*, 2018.

Yu Wang, Eshwar Ghumare, Rik Vandenberghe, and Patrick Dupont. Comparison of different generalizations of clustering coefficient and local efficiency for weighted undirected graphs. *Neural Computation*, 29(2):313–331, 2017.

Larry Wasserman. Topological Data Analysis. *Annual Review of Statistics and Its Application*, 5(1), 2018.

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998. ISSN 0028-0836.

W.W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

Yuanyuan Zhu, Hao Zhang, Lu Qin, and Hong Cheng. Efficient mapreduce algorithms for triangle listing in billion-scale graphs. *Distributed and Parallel Databases*, 35(2):149–176, Jun 2017. ISSN 1573-7578.