The Boundary Coefficient: a Vertex Measure for Visualizing and Finding Structure in Weighted Graphs

Robin Vandaele*
robin.vandaele@ugent.be
Elis, Ghent University

Yvan Saeys
yvan.saeys@irc.vib-ugent.be
DaMBi, VIB-UGent IRC

Tijl De Bie
tijl.debie@ugent.be
Elis, Ghent University

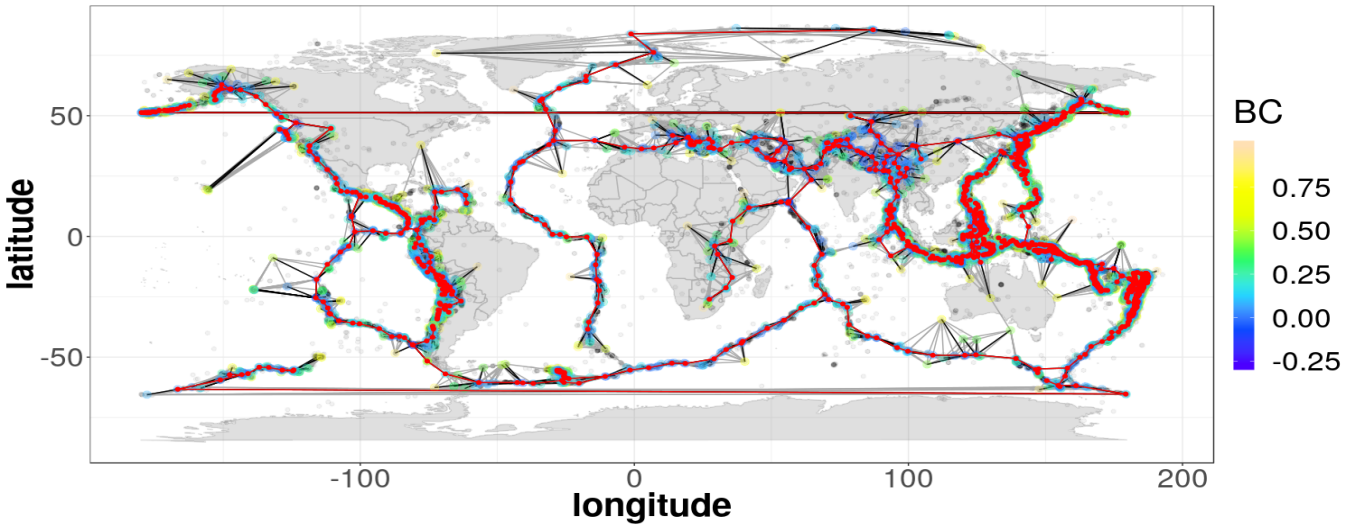


Figure 1: Using boundary coefficients (BC) to derive a backbone (red edges) from a BC-pine (black edges) in a 10NN-graph (grey edges) derived from earthquake locations scattered across the Earth. Note that the left and right side are connected.

ABSTRACT

Graphs have emerged as powerful representations of data, from obvious examples such as social networks, to proximity graphs of high-dimensional metric data. Many of such real-world data sets share a common property: *they have a well-hidden and much simpler graph-structured core, from which all data points emerge.* Uncovering these core structures, in this paper termed *backbones*, often offers great insight into these data sets. However, standard methods for identifying these are computationally inefficient, sensitive to outliers, and lead to *topological bias*, prioritizing low-weight edges in dense regions, instead of spreading out smoothly across the topology underlying the graph. Furthermore, for high-dimensional metric data, standard methods for dimensionality reduction often fail to reveal the hidden topology. We resolve these issues by introducing the *boundary coefficient* (BC), a powerful vertex measure for locating core structure in data sets with an underlying graph-structured topology. Combining the BC with the newly proposed concept of f-pines, we propose a generally applicable method for revealing these structures in such data. We evaluate our method on a number of artificial and real-life data sets, demonstrating its wide range of applicability, superior effectiveness, robustness against noise, and scalability.

KEYWORDS

Topological data analysis, graph theory, social networks, centrality, metric spaces, visualization

ACM Reference Format:
Robin Vandaele, Yvan Saeys, and Tijl De Bie. 2019. The Boundary Coefficient: a Vertex Measure for Visualizing and Finding Structure in Weighted Graphs. In *KDD '19: ACM SIGKDD Conference on Knowledge Discovery and Data Mining, August 04–08, 2019, Anchorage, Alaska USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Motivation. Many real-world graphs, whether given (social networks, road networks, image webs, ...) or derived as neighborhood graphs from point cloud data (gene expression data of differentiating cells, GPS traces, earthquake locations, galaxy coordinates in space, ...), tend to follow a much simpler graph structured topology [1, 3, 6, 7, 13, 15], in this paper referred to as the *backbone* of the graph. Although the topology of the original graph might be

*Also with DaMBi, VIB-UGent IRC.

KDD '19, August 04–08, 2019, Anchorage, Alaska USA

ACM ISBN 978-1-4503-9999-9/18/06...$15.00
https://doi.org/10.1145/1122445.1122456

complex (e.g., in terms of degree sequences, multifurcations, cycles, ...), often many vertices lie close to some core structure, having a much simpler topology, from which they emerge. Identifying and visualizing the topological structure of these backbones, and hence, of the original graphs, is a non-trivial and active topic of research, applicable to many fields of science [1, 3, 7, 11, 13, 15].

E.g., consider the (point cloud) data set of earthquake locations in Fig. 1. Earthquakes clearly do not occur at random locations, as the majority of earthquake locations lie close to a graph-structured topology, i.e., a backbone, mapped onto the surface of the earth.

As the earthquake data set is 2-dimensional, its graph-structured topology is readily noticed by visual inspection. However, it is clear that such topologies, in high-dimensional data, are hard to uncover, and standard dimensionality reduction techniques will fail in most of the non-trivial cases. Furthermore, often there is no clear way of visualizing a given graph, i.e., when it is not embedded into some Euclidean space. Generally, the complexity of state-of-the-art visualizations increases with the complexity of the graph.

(Local) Topological Data Analysis [1, 4, 8, 15] provides some tools for dealing with this type of data. However, they are often computationally inefficient, or highly sensitive to noise and parameters. Furthermore, their primary focus is on geometric data or graphs (i.e. proximity graphs), whereas backbones are present in a wide variety of other networks (Fig. 2).

In this paper, we provide a new approach for revealing such backbones, consisting of two main ingredients:

- A quantification of how *core* a node is within a given graph, called the *boundary coefficient*, by generalizing intuition from the geometric to the non-geometric case.
- Searching a tree (which can subsequently be pruned to just the core) that connects nodes to the core of this graph according to this new measure.

Related work. (Local) Topological Data Analysis [4, 15] provides tools for investigating unknown structure in point cloud data. However, only a few of them are related to the particular case of data approximating graph-structured topologies. Revealing these is the main topic of [1, 15]. Here, the authors make use of local detection techniques to classify edges and multifurcations. Global reconstruction techniques are used to retrieve the underlying topology from this information. However, such methods induce a high parameter sensitivity, and quickly fail in more complex or noisy examples. Furthermore, they are mainly built for investigating topology in point cloud data, and require a *Rips-graph* (Subsec. 2.1) to be built from the data. They are almost always incapable of retrieving the topology when another neighborhood graph, such as a k-nearest neighbor (kNN) graph, is used. *Local persistent (co)homology* [8, 16] provides a more general framework for investigating local structure in—not necessarily graph-structured—data. However, these methods are again sensitive to parameters and noise, computationally inefficient, and do not consider reconstructing the global topology. Another possibility is the *Mapper* algorithm [11], built as a general data visualization tool. However, the quality of its result highly depends on the used filters[1], and the method has been shown to fail to retrieve the underlying topology in simple cases before [15].

[1]These are projections of the data to a lower dimensional space, usually $\mathbb{R}$ or $\mathbb{R}^2$, required by the algorithm.

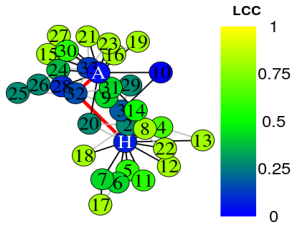


Figure 2: A linear backbone (red) in the karate network [18].

Facility Location Problems [10] deal with finding subgraphs that well approximate a given graph. They consider a trade-off between a distance measure between the given graph and the subgraph on the one hand, and the cost of the subgraph on the other hand, expressed e.g. by the sum of the weights of its edges. However, these methods are often computationally inefficient, sensitive to outliers, or *topologically biased*, by which we mean that the solutions are biased towards including lower weight edges, instead of spreading out smoothly across the graph topology.

Some other vertex measures that might be used to identify core nodes in graphs exist, such as the *local cluster coefficient* (Fig. 2 & Subsec. 2.4), or its generalizations to weighted graphs [17]. However, these were not built for the particular purpose of identifying or visualizing core structure within a wide variety of graph-structured data sets. As such, they lack important properties of the boundary coefficient, such as applicability to fully weighted networks and robustness to outliers, as we confirm in our experiments.

Contributions.

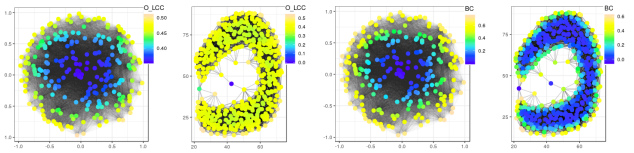
- We introduce a new approach for revealing backbones in graphs, consisting of two components (Sec. 2 & 3).
- We introduce the *boundary coefficient* to locate core structure in graphs, with extensive theoretical analysis and comparisons, and an algorithm for efficient computation (Sec. 2).
- We show how the boundary coefficients are used for efficiently finding the best backbone in a graph by means of the minimum spanning tree (MST) algorithm (Sec. 3).
- We experimentally verify that our method provides a universal approach towards locating simplified structures in a wide variety of real-world data sets (Sec. 4).
- We summarize how our method improves on state-of-the-art approaches, and opens up new possibilities for further improvements and applications (Sec. 5).

2 LOCATING CORE STRUCTURE IN GRAPHS

In this section, we start with intuitive examples to illustrate what the *core* nodes of a graph are, and that these are hard to identify by current existing vertex measures (Subsec. 2.1). Inspired by Euclidean geometry, we introduce the *transmissivity* of a node, which will be higher on average for nodes representing the core of a graph (Subsec. 2.2). Hence, averaging the negative transmissivity will lead to the *boundary coefficient* (BC), marking *boundary* nodes by higher values (Subsec. 2.3). We discuss the properties of BC, as well as its relation to the ordinary *local cluster coefficient* (Subsec. 2.4). Finally, we will show how to efficiently compute the boundary coefficients by means of matrix multiplication (Subsec. 2.5).

2.1 The Core Nodes of a Graph

Real data often contains noise, outliers, and noisy interconnections between different components. In these cases, approximating every single node by one structure, as customary in *Facility Location Problems* [10], will 'shift' and 'wiggle' the resulting structure away from the graph's true core(s). Hence, our goal is not give an exact definition of the core nodes in a given graph, but to find a function that performs well under these conditions to identify and separate core nodes within and between components of this graph. We emphasize that throughout this paper **by core nodes we mean nodes away from the 'boundary', which is not equivalent to nodes close to the global center of the graph** for our purpose. We will illustrate this by means of intuitive toy examples (Fig. 3).



(a) (b) (c) (d)

Figure 3: Onnela's generalized local cluster coefficients (a & b) and boundary coefficients (c & d) for a complete weighted graph representing a disk (a & c), and a Rips-graph ($\epsilon = 10$) representing the letter 'C' with outliers (b & d).

Consider the complete graph $G = (V, E)$ displayed on the left of Fig. 3. This graph is built from a 2D sample of 250 points on the unit disk, where the weight $\omega(\{u, v\})$ of the edge $\{u, v\}$ equals $\|u - v\|$. Since this graph represents a disk, we have a good intuition about our core nodes being the nodes closer to the center of the disk underlying this graph. Our purpose is to find a function from which we can identify how close a node is to this center, based solely on the graph itself, and its weighting function ω.

As this is a complete graph, computing many different measures mapping a node to a local degree of 'clustering', such as the ordinary *local cluster coefficient* (Subsec. 2.4), the *local efficiency*, or some of their generalizations to weighted graphs [17], will result in a uniform mapping from V to 1, providing no further information.

Onella's generalized local cluster coefficient (O_LCC) for weighted graphs [12, 17] provides a good measure for quantifying the 'closeness to the center' in this particular case, as seen in Fig. 3. This coefficient assigns every node v to a value proportional to the *geometric mean* of the edge weights[2], namely $\sqrt[3]{\omega(\{u,v\})\omega(\{v,w\})\omega(\{u,w\})}$, averaged over all triples (u, v, w) *adjacent* to v, i.e., for which $\{u, v\}, \{v, w\} \in E$. However, as can be seen on the second plot of Fig. 3, this coefficient is quite sensitive to outliers. Here, we built a *Rips-graph*, consisting of the set of data points as nodes, and an edge $\{u, v\}$ whenever $0 < \|u - v\| < \epsilon$ for some fixed parameter $\epsilon \in \mathbb{R}^+$, on top of 2D 'C-shaped' point cloud data set with outliers.

Hence, we are clearly in need of a function that is better capable of recognizing the core nodes of a graph. This is where the BC comes into play (Fig. 3). This coefficient will show to drastically outperform the non-weighted standards, as well as the state-of-the art generalizations to weighted graphs [17], for locating core structure in a wide variety of graphs (Fig. 3 & Sec. 4).

[2] $\omega(\{u, v\})$ is defined to be 0 if $\{u, v\} \notin E$.

2.2 The Transmissivity of a Node

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

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

As all of the terms in the fraction may be expressed as distances (between pairs of the triple of vectors $(\bar{x}, \bar{y}, \bar{0})$), we may straightforwardly generalize the concept of angle to arbitrary metric spaces (M, d). Furthermore, since, a 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, we may extend the definition of angle in Euclidean spaces to graphs as well.

Definition 2.1. Let $G = (V, E)$ be a simple, 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

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

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

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

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[3]. $\mathcal{T}(u, v, w)$ 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. 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 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 $u = w$.

2.3 The Boundary Coefficient

The *boundary coefficient* (BC) of a node v is defined as its negative transmissivity averaged over the pairs of neighbors of v. As illustrated by Fig. 3 and Fig. 4, this is a measure for how close vertices are near the 'boundary' of the graph (hence the name), and by this, how far they are from its core.

Definition 2.2. Let $G = (V, E)$ be a simple, undirected, positively weighted graph. For every $v \in V$ we define $\mathcal{N}(v) \subseteq V$ as the set of neighbors of v in G. For every $v \in V$ with *degree* $\delta(v) := |\mathcal{N}(v)| > 0$, we define it's *boundary coefficient* (BC) as

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

The BC attains lower values in nodes lying more near the core of the graph (Fig. 3 & 4). Note that outliers having a low BC coincides with the idea of those nodes lying closer to the core of the graph:

[3] Furthermore, the weights ω do even not have to satisfy the *triangle inequality* in the graph $G = (V, E)$. I.e., 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.

Figure 4: Basic intuition behind 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 dotted line represents the shortest path — not necessarily an edge — between two nodes.

they just represent isolated points in the core, and are separated from its major component(s) through the boundary nodes (Fig. 3).

2.4 Properties of the Boundary Coefficient

We summarize some properties of the newly introduced BC. Proofs and intuition of the following propositions are omitted for conciseness, and are not necessary to comprehend the rest of the paper.

As a sum of values over pairs of adjacent nodes, the BC is closely related to the ordinary local cluster coefficient (LCC) on unweighted graphs:

Proposition 2.3. *Suppose $G = (V, E)$ is a unweighted graph, i.e., a graph in which every edge gets a weight equal to 1. Then for every $v \in V$ with $\delta(v) > 1$, it holds that*

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

Hence, the BC does not fulfill the *general versatility* requirement of an LCC generalization [17]: it does not coincide with the LCC for unweighted graphs. However, it does satisfy three other criteria of generalizations of the LCC to weighted graphs [17]. One of these, namely its *applicability to fully weighted networks*, has been illustrated in Fig. 3. This property, as well as the following two, are important for our purpose of identifying core structure in a wide variety of weighted graphs.

Proposition 2.4. *(Weight-scale invariance). Let $G = (V, E)$ be a simple, undirected graph, 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 ω_λ.*

Proposition 2.5. *(Robustness to noise). Let $G = (V, E)$ be a simple, undirected graph, with weighting function $\omega : E \to \mathbb{R}^+$. Suppose $\mathcal{E}$ is an additive noise matrix defining a new weighting function $\omega_\epsilon : E \to \mathbb{R}^+ : \{u, v\} \mapsto \omega(\{u, v\}) + \mathcal{E}_{u,v}$. Then*

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

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

The crucial differences between the BC, the LCC (and many of its generalizations [17]), and standard global *centrality* measures such as *eccentricity* or *betweenness* [2], as well as the main reasons why the BC outperforms these measures for a wide variety of applications (Fig. 3 & Sec.4), 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, O_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, O_LCC, ...);
- The scope of the BC is local: it does not take into account the shortest paths from v to all other nodes (as often the case with standard centrality measures). Hence, the BC allows us to locate boundary nodes even in the presence of complex, long, or curving underlying topologies.

2.5 Computing the Boundary Coefficient

Given the pairwise distances between nodes in a graph G with n nodes and m edges, a straightforward computation of the BC as defined in Def. 2.2 for all nodes would require three nested for-loops, resulting in a $O(n(\delta_G^{\max})^2)$ algorithm for a graph G with n nodes and m edges. In practice, this can be improved to the computational cost of (sparse) matrix multiplication

Theorem 2.6. *Let $G = (V, E)$ be a simple, undirected, positively weighted graph. Let D be the matrix of pairwise distances in G, and let $D^2_\odot$ and $1/D$ be the pointwise square and inverse of D, respectively. Then for $v \in V$ with $\delta(v) > 0$,*

$$BC(v) = \frac{1}{\delta(v)^2} \left[\sum_{u \in \mathcal{N}(v)}^{n} D_{u,v} \sum_{u \in \mathcal{N}(v)}^{n} \frac{1}{D_{u,v}} - \frac{1}{2} \left(\frac{1}{D} \left(D^2_\odot \right) \frac{1}{D} \right)_{v,v} \right].$$

Proof. This can be shown from algebraic manipulation of the formulas in Def. 2.1 & 2.2. Details are omitted for conciseness. □

For computational efficiency, D is stored as a sparse matrix, removing all entries $D_{u,w}$ where the hop count between u and w is greater than 2. Hence, given a pairwise distance matrix D for G, the computational complexity is that of computing the multiplication of two $n \times n$ matrices, which is $O(n^{2.375})$. The storage cost is $O(n^2)$. If D is not given, running pairwise Dijkstra's algorithm is $O(n(m + n \log n))$, and the total computational cost is $O(n(m + n^{1.375}))$.

A practical method for dealing with the computational cost of computing pairwise Dijkstra's would be to modify the algorithm to not compute $D_{u,w}$ if the hop count distance between u and w is greater than 2. Other options to speed up the computation include parallelizing pairwise Dijkstra's or the matrix multiplication, tresholding the number of hops to approximate $D_{u,w}$, approximating D by the original distance matrix for neighborhood graphs constructed from point cloud data, or combinations of these.

3 CONSTRUCTING THE BACKBONE

In the previous section we introduced the BC, measuring the average (negative) transmissivity of a node, marking how close it lies to the boundary of a graph. In this section, we introduce the *f-pine* of a graph, with leaves located at higher values of a given function f (Subsec. 3.1). Hence, by letting f = BC, we obtain a tree with leaves located at the boundary of the graph (Subsec. 3.2). By iteratively retracting this tree, i.e., pruning leaves, we will move deeper inwards into the backbone of the graph (Subsec. 3.2).

3.1 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 tree with leaves marking higher values of f. We will call such a tree an *f-pine*: its needles 'stick out and point' towards (locally) high values of f.

Definition 3.1. Let $G = (V, E)$ be a graph, and $f : V \to \mathbb{R}$. A spanning tree T of G is called an *f-pine* in G, if

$$\sum_{v \in V} \delta_T(v) f(v) \quad = \quad \min_{\text{spanning trees } T' \text{of } G} \sum_{v \in V} \delta_{T'}(v) f(v), \quad (1)$$

where $\delta_T(v)$ denotes the degree of v in the subgraph T of G.

I.e., an f-pine in G is a spanning tree 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:

Theorem 3.2. *Let $G = (V, E)$ be a graph, $f : V \to \mathbb{R}$, and T an f-pine in G. For every $u \in V$ with $\delta_G(u) > 0$, there exists some $v \in \arg\min\{f(w) : w \in \mathcal{N}_G(u)\}$ such that $\{u, v\} \in E(T)$.*

Proof. 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 (1). □

The intuition behind the theorem 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, Def. 3.1 implies that the centers of these star graphs will be connected through nodes where f attains a low value on average as well (Th. 3.3 & 3.4).

One may 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.

Theorem 3.3. *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 in G, where each edge $\{u, v\}$ is assigned to have weight $f(u) + f(v)$.*

Proof. This can easily be seen by rewriting the cost expressed in (1) in terms of these new edge weights. □

Finally, an f-pine is invariant to affine transformations of f. Hence, one may apply such transformation to f without effecting the resulting f-pine, so that one may easily compare the BC(-pine) to other transitivity or centrality measures on a similar scale.

Theorem 3.4. *Let $G = (V, E)$ be a graph, $f : V \to \mathbb{R}$, and T an f-pine in G. If $g = af + b$ for some $a, b \in \mathbb{R}$, T is also a g-pine in G.*

Proof. This immediately follows from either Def. 3.1 or Th. 3.1. □

3.2 The BC-pine of a Graph

Th. 3.2 states an f-pine attaches a node u to a node where f reaches a minimum over the neighbors of u. As the BC marks nodes inside the core of a graph with lower values (sec. 2.3), we can construct a *BC-pine* with leaves located at the boundary of the graph by using the BC to redefine the weight of every edge, and constructing a minimum spanning tree (Th. 3.3). Furthermore, Th. 3.3 assures that the connections between these local minima remains within the core of the graph.

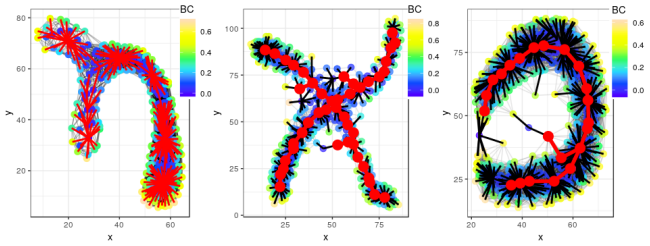


Figure 5: Some example of BC-pines (black + red edges) within Rips-graphs constructed from synthetic 2D point cloud data sets containing 300, 390, and 477 data points, respectively. (Left) Non-pruned. (Middle + Right) Pruning the pines twice results in the red graphs.

In summary, the key properties of a BC-pine are (Fig. 5):

- the BC-pine contains a backbone of G passing smoothly through the core of G without topological bias towards lower weight edges (in terms of the original weighting function);
- the BC-pine contains many (non-significant) leaves marking boundary nodes, that quickly vanish after iteratively pruning the tree, retracting deeper into the backbone of G.

By pruning leaves, the BC-pine quickly retracts towards the backbone, passing smoothly through the graph's core (X & C-shape in Fig. 5). Note that the BC-pines connect to outliers as well, as would any spanning tree do. However, the BC-pine does not wrongly interconnect different (parts of) branches through these. Hence, they will also quickly disappear after more pruning (Fig. 5).

4 EXPERIMENTAL RESULTS

In this section, we show various applications of our boundary coefficients and f-pines, including identifying core structure, graph simplifications, and visualization. Note that graph simplifications have recently been shown to be able to increase the performance of existing graph embedding methods [5]. Our method will show to be applicable to a wide variety of data sets, from social networks, to topological data analysis of high-dimensional point cloud data.

4.1 Social Networks

We have already shown by means of the Karate network [18], that backbones may even be present in social networks (Fig. 2), not only in metric graphs. This backbone (red edges in Fig. 2) was computed by pruning the LCC-pine (black edges in Fig. 2) twice.

However, even in more complex social networks, there may be a core structure present from which all other nodes emerge. Furthermore, having a measure of 'closeness' between two nodes in a network by means of edge weights, may provide further insight into the core structure of the network by constructing a BC-pine.

This is verified on a KDD co-authorship graph, a social network containing 5747 and 18715 edges. The data used to construct this graph is publicly available on aminer.org/citation. Each edge between two nodes marks two authors who co-authored at least one

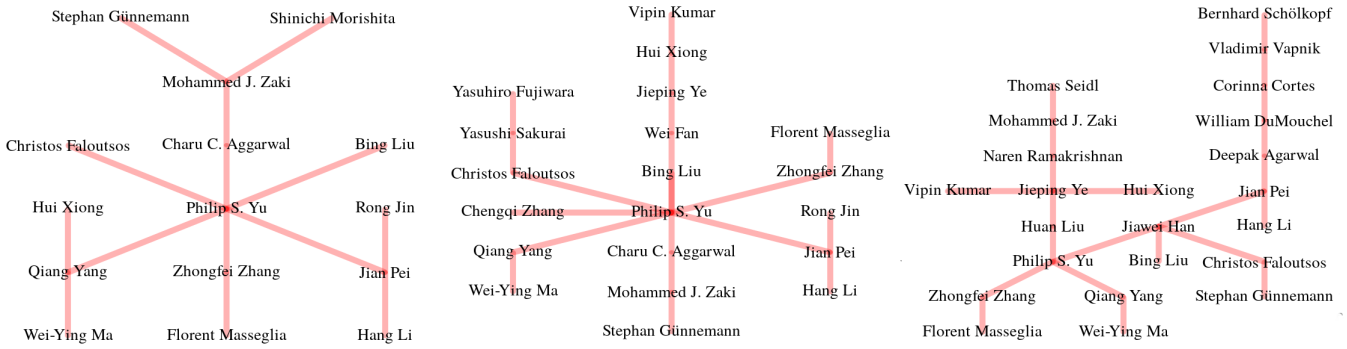


Figure 6: Identifying core structure in the KDD co-authorship network using different pines (all pruned eight times for visualization). (Left) LCC-pine. (Middle) O_LCC-pine. (Right) BC-pine.

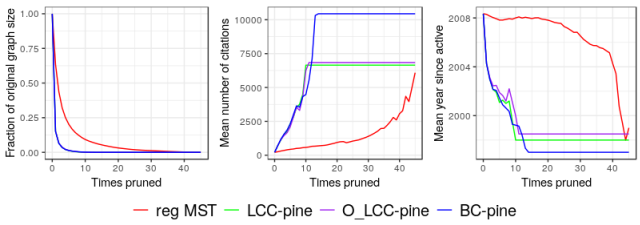


Figure 7: Various measures after iteratively pruning the regular MST (red), the LCC-pine (green), the O_LCC-pine (purple), and the BC-pine (blue). (Left) Fraction of original graph size. (Middle) Average number of (KDD) citations. (Right) Average year of first published (KDD) paper.

KDD paper. The weight of each edge is the inverse of the number of KDD papers co-authorized by the authors. As such, lower weights correspond to authors who are closer to each other in the KDD network. The resulting LCC-pine, O_LCC-pine, and BC-pine, were constructed and pruned eight times for visualization (Fig. 6).

We compared various measures of our network when moving deeper into the backbone by pruning leaves (Fig. 7), namely:

- the fraction of the nodes still included in the backbone;
- the average number of citations of the nodes (authors) still included in the backbone;
- the average year of the first KDD publication across al nodes (authors) still included in the backbone.

By iteratively pruning leaves, we note that all three pines retract much faster to a core structure than the regular MST, discarding the majority of the nodes after the first iteration, a consequence of Th. 3.2. Furthermore, when pruning leaves, all pines quickly retract to a core structure marking authors with many (KDD) citations, and who are already present a long time in the network, as can be expected from the core of this network.

Though all three pines show to satisfy these properties, we do note some differences between them. The LCC-pine and O_LCC-pine reveal more of a 'star-structured' topology within the core of the network, centered around the same author. The BC-pine shows more of a 'coat rack' structure within the core. The similarity between the former two pines may be declared by O_LCC focusing on generalizing the ordinary LCC. The BC focuses on revealing long and branching structures through the core nodes of the graph.

Though most authors were pruned from the pines for visualization, Fig. 7 reveals that many 'core' authors (in terms of number of citations and years active) lie close to these structures.

4.2 Metric Data

There are a wide variety of point cloud data sets embedded in a metric space approximating a graph-structured topology. For such data, our method provides a tool to build the underlying simplified network topology for analysis. Furthermore, it may serve as a tool within the field of Topological Data Analysis, where is it used to identify topological structure that would otherwise be hidden due to the high dimensionality of the the data set, as state-of-the art methods for dimensionality reduction often fail to provide insight into the core structure underlying such data.

4.2.1 Earthquake locations. We already showed the result of our method on a data set of earthquakes scattered across the globe at the start of this paper (Fig. 1).

To obtain this result, we proceeded as follows. We obtained a data set containing information on 80549 earthquakes, which is freely accessible from USGS Earthquake Search. The topology underlying such a data set was already analyzed in [1, 15]. However, contrary to the procedure presented in these papers, we did not restrict to a particular small rectangular domain with a low amount of noise, and did not apply any noise filtering in advance.

A random sample of 5000 earthquakes was taken, from which we constructed an undirected 10NN graph with 31334 edges using the Great circle (geographic) distances between location coordinates. Boundary coefficients where computed using these distances as weights. The BC-pine was pruned once, reducing the number of nodes by 78.5%, and the number of edges by 96.6%. Even though pruning once drastically reduced the graph's size, the resulting tree remained nicely spread out across the underlying topology (Fig. 1).

4.2.2 Cell Trajectory Data. [4] Our method can be used as a tool for data visualization within the framework of Topological Data Analysis as well. We will demonstrate this by means of a synthetic cell trajectory data set of 556 cells in a 3475-dimensional gene

[4] These data sets are publicly available at zenodo.org/record/1443566#.XD7lb8tKhhF.

expression space [14]. This data represents a snapshot of these cells at a specific point during a cell differentiation process, and is visualized by means of a MDS plot on Fig. 8.

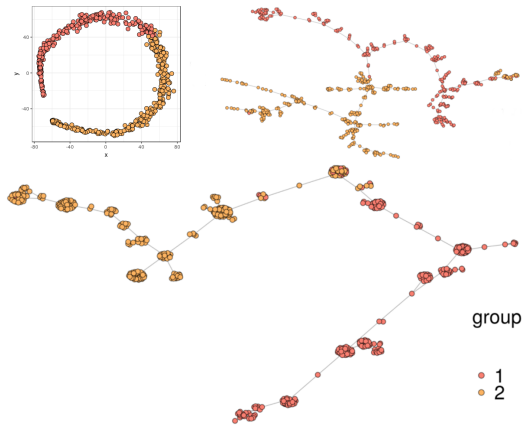


Figure 8: A synthetic cell trajectory data set of 556 cells in a 3475-dimensional gene expression space. Trees are non-pruned and visualized using the Fruchterman-Reingold layout algorithm [9]. (Top left) Data visualized on a MDS plot. (Top right) Data visualization using the regular MST. (Bottom) Data visualization using the BC-pine.

The model underlying this data set is a continuous transition between two different cell groups in the high-dimensional gene expression space. This means that a linear backbone is present in the data. This topology can be noted from the MDS plot as well (Fig. 8). However, in general, even when the ground truth is a simple topological model, identifying and visualizing such structures shows to be a difficult task due to the high dimensionality and amount of noise present in such data [14].

A 10NN graph was constructed from this data, using the Euclidean distance between cells in the expression space as weights. In Fig. 8, a regular minimum spanning tree (MST) of this data is shown. The MST performs reasonably well for clustering the different groups of cells together, except for a small cluster of group 2 cells completely disconnected from the main cluster. The main issue with the regular MST lies clearly in the visualization of the topological structure, as their appear to be many long branching structures.

However, the BC-pine is constructed such that it pulls every node towards the backbone (Th. 3.2), which travels smoothly through the graph without topological bias towards shorter edges. This is clearly seen from the visualization using the BC-pine in Fig. 8, where we note a linear backbone passing through the graph, to which all cells connect to shortly. Furthermore, contrary to the regular MST, both groups of cells are clustered nicely together, with only being mixed at the transition between the two groups.

A similar experiment was conducted on a real cell trajectory data of 355 cells embedded into a 3397-dimensional gene expression space. The ground truth model is that of a Y-shaped topology, and is shown on Fig. 9, along with an MDS plot of the data. This time, the ground-truth model would be much more difficult to infer from the dimensionality reduction without any prior knowledge. A 5NN graph was built from this data, using the Euclidean distance as well.

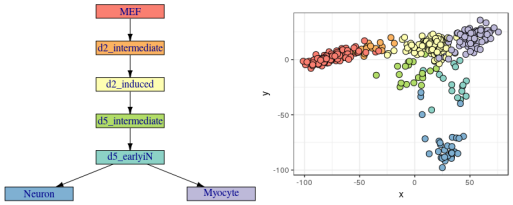


Figure 9: (Left) Ground-truth topological model underlying the real expression data. (Right) MDS plot of the data.

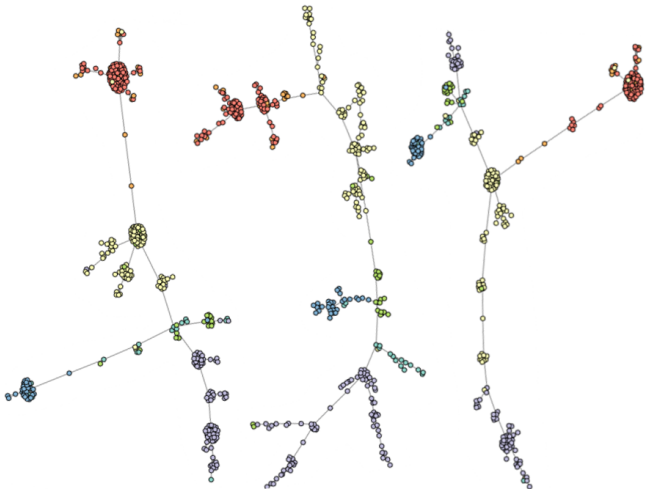


Figure 10: Visualizing trees of our real expression data using the Fruchterman-Reingold layout algorithm. The coloring corresponds to the coloring in Fig. 9. (Left) The BC-pine. (Middle) The regular MST. (Right) The O_LCC-pine.

Fig. 10 shows three different visualizations of this data by means of spanning trees: the BC-pine, the regular MST, and the O_LCC-pine. Again, the regular MST performs well at clustering together similar cell groups, but fails to reveal the hidden Y-structured topology. In contrast, the Y-structure topology is much more notable from the pines. However, the BC-pine improves a lot on the correlation between the placement of cell groups in the pine and the ground-truth. The O_LCC-pine appears to displace two Myocyte clusters completely at opposite ends in the underlying topology, and bifurcates very early at the start of the d2_induced cells.

4.3 Network Sizes and Computation Times

Table 1 summarizes our experimental results. All results were obtained using non-optimized R code on a machine equipped with a Intel® Core™ i7 processor at 2.6GHz and 8GB of RAM.

The unweighted diameter is a rough measure for how fast the tree will retract into its core. As a result from Th. 3.2, all three pines show to retract much faster compared to the regular MST. We note

Table 1: Summary of our results. $|V|$: number of nodes in the original graph. $|E|$: number of edges in the original graph. time_{BC}: computation time for full pairwise Dijkstra's algorithm, a full unweighted distance matrix (to discard unnecessary entries), and obtaining the BC by means of Th. 2.6.[a] $\text{time}_{\text{O\_LCC}}$: computation time of Onella's generalized LCC. diam_{BC}: diameter of the BC-pine. diam_{LCC}: diameter of the LCC-pine. $\text{diam}_{\text{O\_LCC}}$: diameter of the O_LCC-pine. diam_{MST}: diameter of the reg. MST.

	$\|V\|$	$\|E\|$	time_{BC}	$\text{time}_{\text{O\_LCC}}$	diam_{BC}	diam_{LCC}	$\text{diam}_{\text{O\_LCC}}$	diam_{MST}
KDD	5747	18715	28s	5min10s	28	21	23	90
Earthquakes	5000	31334	16s	3min27s	265	259	433	1119
Synth. cells	556	4636	1.6s	0.9s	30	32	34	45
Real cells	355	1543	1.2s	0.5s	15	19	19	26

[a]Due to the R igraph/distances function calling optimized C code, this approach is currently faster than implementing Dijkstra's algorithm with early exit for each node in R.

that our method scales well up to graphs with thousands of nodes, and as discussed in Subsec. 2.6 and Sec. 5, there are still ways to improve this even further for higher order graphs.

5 CONCLUSION AND FURTHER WORK

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 towards this problem, applicable to both general networks, and to point cloud data approaching such topologies. State-of-the-art approaches 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 the *boundary coefficient*, a local vertex measure closely related to the local cluster coefficient (LCC) (Prop. 2.3). Nevertheless, it is built for a completely different purpose, and outperforms other generalizations of the LCC and centrality measures for the task of finding core structure in a wide variety of data sets. Combining these new coefficients with our concept of *f-pines* allowed us to reliably identify the backbone of these graphs, with applications focusing on identifying, visualizing, and/or simplifying to this core topology.

By iteratively pruning leaves in the BC-pine, we quickly retract towards the backbone of a graph, discarding the majority of the nodes. However, in order to reduce the effects of noise and outliers or ease the visualization of the backbone topology, we often need to prune more than we want, retracting our core from the 'true' leaves of the underlying topology as well (Fig. 5 & 6). We already developed two refined pruning algorithms overcoming this issue, as well as an algorithm to automatically add significant cycles to our backbone, which will be discussed in subsequent work.

Further work includes optimizing our code, i.e., avoiding the use of full pairwise Dijkstra's algorithm to compute the boundary coefficients, accompanied with a refined theoretical complexity analysis in terms of the sparsity of the network. Furthermore, our algorithm is easily parallelized, and we plan to use this approach to conduct even more experiments on very large graphs.

ACKNOWLEDGMENTS

This work was funded by the ERC under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC Grant Agreement no. 615517, and the FWO (G091017N, G0F9816N).

We furthermore acknowledge Bo Kang for helpful discussions.

REFERENCES

[1] Mridul Aanjaneya, Frederic Chazal, Daniel Chen, Marc GLisse, Leonidas Guibas, and Dmitriy Morozov. 2012. Metric Graph Reconstruction from Noisy Data. *International Journal of Computational Geometry and Applications* 22, 04 (2012), 305–325.

[2] M. Barthélemy. 2004. Betweenness centrality in large complex networks. *The European Physical Journal B* 38, 2 (01 Mar 2004), 163–168.

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

[4] Gunnar Carlsson. 2009. Topology and data. *Bull. Amer. Math. Soc.* 46, 2 (jan 2009), 255–308. https://doi.org/10.1090/S0273-0979-09-01249-X

[5] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. 2018. HARP: Hierarchical Representation Learning for Networks. In *AAAI*.

[6] Ena Choi, Nicholas A. Bond, Michael A. Strauss, Alison L. Coil, Marc Davis, and Christopher N. A. Willmer. 2010. Tracing the filamentary structure of the galaxy distribution at zâĹij0.8. *Monthly Notices of the Royal Astronomical Society* 406, 1 (jul 2010), 320–328.

[7] Leen De Baets, Sofie Van Gassen, Tom Dhaene, and Yvan Saeys. 2016. Unsupervised Trajectory Inference Using Graph Mining. Springer, Cham, 84–97.

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

[9] Thomas M. J. Fruchterman and Edward M. Reingold. 1991. Graph Drawing by Force-directed Placement. *Softw. Pract. Exper.* 21, 11 (Nov. 1991), 1129–1164.

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

[11] Monica Nicolau, Arnold J. Levine, and Gunnar Carlsson. 2011. 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 (apr 2011), 7265–7270.

[12] Jukka-Pekka Onnela, Jari Saramäki, János Kertész, and Kimmo Kaski. 2005. Intensity and coherence of motifs in weighted complex networks. *Phys. Rev. E* 71 (Jun 2005), 065103. Issue 6. https://link.aps.org/doi/10.1103/PhysRevE.71.065103

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

[14] Wouter Saelens, Robrecht Cannoodt, Helena Todorov, and Yvan Saeys. 2018. A comparison of single-cell trajectory inference methods: towards more accurate and robust tools. *bioRxiv* (2018).

[15] Robin Vandaele, Tijl De Bie, and Yvan Saeys. 2019. Local Topological Data Analysis to Uncover the Global Structure of Data Approaching Graph-Structured Topologies. In *Machine Learning and Knowledge Discovery in Databases*, Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, and Georgiana Ifrim (Eds.). Springer International Publishing, Cham, 19–36.

[16] Bei Wang, Brian Summa, Valerio Pascucci, and Mikael Vejdemo-Johansson. 2011. Branching and Circular Features in High Dimensional Data. *IEEE Transactions on Visualization and Computer Graphics* 17 (2011), 1902–1911.

[17] Yu Wang, Eshwar Ghumare, Rik Vandenberghe, and Patrick Dupont. 2017. Comparison of Different Generalizations of Clustering Coefficient and Local Efficiency for Weighted Undirected Graphs. *Neural Computation* 29, 2 (2017), 313–331.

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

RESEARCH METHODS

Code, data, and instructions to run our experiments can be found on https://www.dropbox.com/s/cf05voqh8a96vfu/f_pine.zip?dl=0.