UNIVERSITY OF CALIFORNIA SAN DIEGO

Extracting Graph Structure from Data via Topological Methods with Applications to Neuroscience

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Lucas James Magee

Committee in charge:

Professor Yusu Wang, Chair
Professor Yoav Freund
Professor Gal Mishne
Professor Hao Su

2024

The Dissertation of Lucas James Magee is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2024

DEDICATION

In loving memory of my late uncle, James Magee, who was taken from us far too soon.

EPIGRAPH

Don't worry Lucas, you will graduate in five years.

Dr. Ryan Šlechta

Nothing in this world can take the place of persistence. Talent will not; nothing is more common than unsuccessful men with talent. Genius will not; unrewarded genius is almost a proverb. Education will not; the world is full of educated derelicts. Persistence and determination alone are omnipotent. The slogan 'Press On!' has solved and always will solve the problems of the human race.

Calvin Coolidge

I never conquered, rarely came
Tomorrow holds such better days
Days when I can still feel alive
When I can't wait to get outside
The world is wide, the time goes by
The tour is over, I've survived
I can't wait 'til I get home
To pass the time in my room alone

Mark Hoppus

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my dog, Eros, who entered my life in September 2018. I could write a separate dissertation on how his unconditional love has been vital in getting me across the finish line. In November 2022, I returned to my studio apartment after spending 3 weeks away, feeling like a complete loser who had still accomplished nothing in life. And there was Eros, also returning from his 3 weeks of dog vacation spent with family who walked him every few hours, other dog friends to constantly play with, and yards that I have only been able to promise him will be in our future. But I would not have been able to have guessed that is what he had been up to during our time apart, because he was running around the apartment and excitedly jumping on me, just happy to be back with me in our small space. I am about to thank many others below - none of whom would be thrilled to find out they would have to live with me in a studio apartment.

Secondly, I'd like to thank my collaborator, mentor, and friend Dr. Michael Hawrylycz. I would not have finished this program without him - and not only because our collaborative work is the final piece of this dissertation. I encourage any Ph.D. student reading this to find a mentor like Mike. I'd also like to thank Uygar Sümbül and Rohan Gala from Mike's group at the Allen Institute for their time and thoughts that have gone a long way in helping me finish this dissertation.

I would like to thank my parents for all of the sacrifices they have made for me and my sister. It is a testament to their love and support that both my sister and I are in graduate school. I would like to thank my sister, Emma Magee, for her love and support. Her example of what true belief in oneself looks like has helped me finish graduate school. I would like to thank all of my extended family. I'd like to specifically shout out uncle Seth and aunt Arlen for letting me stay with them during the summer of 2019, aunt Mary and uncle Ed for letting my stay at their place October 2021 and for every holiday since I moved to California, and Mary Liz Horan and Bob Lutz for lending me their vacation home and for watching Eros.

I'd like to thank Conor Gerety, for being there for me since first grade. He is the brother

I never had and his outside perspective has helped keep me (relatively) sane during graduate school. I'd like to thank Ben Call, Brian Desnoyers, and Melanie Kaplan-Cohen for their support and friendship since college. I'd like to thank the Arlington High School group for always welcoming me back with open arms every time I return to the 845 and for supporting me when I am far away. Thank you Anthony Bettina, Nick Brown, Stephen Caruso, Kate Lisewski, Tyler Muldoon, Allysa O'Neill, Matt Spendley, and Kyle Steubing. I'd like to thank my friends from THE Ohio State University - Jayson Boubin, Kelly Boubin, Erik Clarke, Puoya Koosha, and Ryan Šlechta. Transferring robbed us of celebrating moments like this together, but your love and support has not dwindled and is appreciated more than I can ever put into words. I'd also like to thank my San Diego friends - Tristan Brugère, Sam Chen, Nate Conlon, Jesse He, Chester Holtz, Dhruv Kohli, David Lightfoot, Neil Mallinar, Sowmya Manojna Narasimha, Riley Nerem, John Schaff, and Ram Dyuthi Sristi. When I first moved to San Diego I knew no one, and y'all have made this city feel like home. I'd also like to thank my therapist for helping me get across the finish line.

I'd like to thank my advisor, Dr. Yusu Wang, and my committee, Dr. Yoav Freund, Dr. Gal Mishne, and Dr. Hao Su, for their time and thoughts that made finishing this dissertation possible.

Chapter 2, in full, is a reprint of the Preliminaries section as it appears in Graph skeletonization of high-dimensional point cloud data via topological method in Journal of Computational Geometry. Magee, Lucas; Wang, Yusu. 2022. The author of the dissertation was the primary investigator and the author of this article.

Chapter 3, in part, is a reprint of the material as it appears in Semantic segmentation of microscopic neuroanatomical data by combining topological priors with encoder-decoder deep networks in Nature Machine Intelligence. Banerjee, Samik; Magee, Lucas; Dingkang, Wang; Li, Xu; Huo, Bingxing; Jayakumar, Jaik-ishan; Matho, Katie; Lin, Adam; Ram, Keerthi; Sivaprakasam, Mohanasankar; Huang, Josh; Wang, Yusu; Mitra, Partha. 2020. The author of the dissertation was the second author of this article.

Chapter 3, in part, is a reprint of the material as it appears in Detection and skeletonization of single neurons and tracer injections using topological methods in bioRxiv. Wang, Dingkang; Magee, Lucas; Huo, Bingxing; Banerjee, Samik; Li, Xu; Jayakumar, Jaikishan; Lin, Meng Kuan; Ram, Keerthi; Wang, Suyi; Wang, Yusu; Mitra, Partha. 2020. The author of the dissertation was the second author of this article.

Chapter 4, in full, is a reprint of the material as it appears in Minimum Monotone Tree Decomposition of Density Functions Defined on Graphs in Lecture Notes in Computer Science. Magee, Lucas; Wang, Yusu. 2024. The dissertation author was the primary investigator and author of this paper.

Chapter 5, in full, is a reprint of the material as it appears in Graph skeletonization of high-dimensional point cloud data via topological method in Journal of Computational Geometry. Magee, Lucas; Wang, Yusu. 2022. The author of the dissertation was the primary investigator and the author of this article.

Chapter 6, in full, is a reprint of the material to be submitted in Discrete Morse Graph Reconstruction for High-Dimensional Transcriptomic Data. Magee, Lucas; Gala, Rohan; Travaglini, Kyle; Sümbül, Uygar; Wang, Yusu; Hawrylycz, Michael. 2024. The dissertation author was the primary investigator and author of this paper.

VITA

2017	Bachelor of Science, Northeastern University
2020	Master of Science, THE Ohio State University
2024	Doctor of Philosophy, University of California San Diego

PUBLICATIONS

Lucas Magee, Rohan Gala, Kyle Travaglini, Uygar Sümbül, Yusu Wang, Michael Hawrylycz. Discrete Morse Graph Reconstruction for High-Dimensional Transcriptomic Data. *Preprint.* 2024.

Lucas Magee, Yusu Wang. Minimum Monotone Tree Decomposition of Densities Defined on Graphs. In *Combinatorial Optimization and Applications. COCOA 2023. Lecture Notes in Computer Science*. 2024.

Lucas Magee, Yusu Wang. Graph Skeletonization of High-Dimensional Point Cloud Data via Topological Method. In *Journal of Computational Geometry*, Volume 13, Number 1. 2022.

Dingkang Wang, **Lucas Magee**, Bing-Xing Huo, Samik Banerjee, Xu Li, Jaikishan Jayakumar, Meng Kuan Lin, Keerthi Ram, Suyi Wang, Yusu Wang, Partha P. Mitra. Detection and Skeletonization of Single Neurons and Tracer Injections Using Topological Methods. bioRxiv. doi: 10.1101/2020.03.21.000323. 2020.

Samik Banerjee, **Lucas Magee**, Dingkang Wang, Xu Li, Bingxing Huo, Jaik-ishan Jayakumar, Katie Matho, Adam Lin, Keerthi Ram, Mohanasankar Sivaprakasam, Josh Huang, Yusu Wang, Partha P. Mitra. Semantic Segmentation of Microscopic Neuroanatomical Data by Combining Topological Priors with Encoder-Decoder Deep Networks. In *Nature Machine Intelligence*, Volume 2, Number 10. 2020.

ABSTRACT OF THE DISSERTATION

Extracting Graph Structure from Data via Topological Methods with Applications to Neuroscience

by

Lucas James Magee

Doctor of Philosophy in Computer Science

University of California San Diego, 2024

Professor Yusu Wang, Chair

Datasets are often noisy, high-dimensional, and complex, but they frequently contain intrinsic structures that can aid in both understanding the data and enabling downstream applications. One such structure is graphs, and in particular, trees. This dissertation develops efficient methodologies using geometric and topological ideas to extract graph-like structures from both low- and high-dimensional datasets, with applications in neuroscience.

The first direction focuses on extracting tree structures from 2D and 3D imaging data. Specifically, we aim to extract neuronal tree morphologies from mouse brain imaging datasets. We employ discrete Morse (DM) graph reconstruction to improve neuronal process segmentation

and single neuron skeletonization. Additionally, we have published both 2D and 3D full brain skeletonization frameworks on Github.

In the second direction, we explore decomposing full brain neuronal process skeletonizations into the individual neurons that make up the graph. This involves decomposing a graph with node density into a sum of monotone trees, which model individual neurons. We demonstrate that this generalization and several related problems are NP-complete, establish approximation bounds, and present approximation algorithms for solving these problems.

In the third direction, we extend our approach to handle high-dimensional, noisy point cloud datasets (PCDs). This requires us to view the DM algorithm from a filtration perspective instead of a density perspective. We propose a generalized algorithm that guarantees lex-optimal cycles in output graphs and combine this with the sparse weighted Rips filtration for efficient and effective graph extraction from PCDs.

In the final direction, we combine the generalized algorithm with a filtration defined with respect to Jaccard index to develop a DM graph reconstruction algorithm for scRNA-seq datasets. Output graphs are then used to accurately analyze gene expression gradients between cell types, develop cell type taxonomies, and quantify changes in gene expression over Alzheimer's disease progression.

Chapter 1

Introduction

1.1 Overview

Modern datasets are often very large, both in number of samples and number of dimensions, and are, in practice, frequently very noisy. This makes it inherently challenging not only to analyze the datasets themselves, but also to properly use the datasets in downstream applications. A key principle in modern data analysis is that datasets, no matter how large or noisy, often have an underlying, simpler structure. An example of one such structure is a graph - a 1-D singular manifold made up of the union of individual 1-manifolds glued together at endpoints. Graphs are an important structure because they are versatile and are very commonly seen in practice, such as in a variety of networks (road, river, etc.), data trend models, and cosmic webs formed in dark matter. Trees, which are graphs without cycles, are also observed frequently in practice. Extracting the inherent graph (or tree when appropriate) structure behind datasets can greatly improve our understanding of the datasets themselves, as well as simplify the input and improve the performance of downstream applications of datasets. Thus, it is critical to develop meaningful graph extraction algorithms to capture true underlying 1-D structure from noisy, high dimensional datasets.

Non-linear dimensionality reduction has been successfully applied to a broad range of applications to analyze high dimensional data [125, 135, 9, 46, 99, 139, 84]. Nevertheless, there are still key challenges: In particular, given that this process is often an inherent lossy process,

what is being preserved in the low-dimensional embedding is ultimately determined by the objective function as well as the optimization procedure used to optimize this highly non-convex problem. One way to alleviate this problem is to directly extract meaningful structures from high dimensional space and perform analysis based on that, or develop dimensionality reduction methods that can better preserve structure from the original, high-dimensional space. An example of one such meaningful structure of high-dimensional datasets is its underlying graph skeleton. While several methodologies have been developed to extract the true underlying graph structure of datasets [69, 82, 110], there are some challenges in doing so. Identifying the nodes of the graph and defining the connections between the nodes is particularly difficult, with difficulty only increasing with high-dimensional datasets. Many approaches rely on local information to make such decisions, which makes them ineffective when dealing with real world datasets that are often noisy or contain gaps in the data. **The goal of my dissertation is to develop effective graph extraction algorithms using topological methods with neuroscientific applications in mind.**

1.1.1 Homology

To this end, topological methods can help overcome such deficiencies, as they are able better capture coarser but essential key structures in data and are better suited in defining global connectivity. Intuitively, the focus of analysis remains on features that remain present as long as the connectivity does not change. Recent years have witnessed great advancements in topological data analysis (TDA), see e.g, surveys and books [23, 44]. Roughly speaking, TDA consists of algorithms used to quantify discrete datasets using concepts from topology, a field of mathematics concerned with categorizing continuous spaces by their connectivity. Topological objects and language intuitively provide us tools to characterize essential structures in data. For example, homology groups of a continuous space, capture the independent holes of a space. The 0th homology group captures connected components, the 1st homology group captures loops, the 2nd homology group captures voids, etc.

In this dissertation, we will consider the so-called simplicial homology. Essentially, we will model the space of interest using simplicial complexes, and study the homology group induced by them. In particular, the domain of datasets is modeled by simplicial complexes. A simplicial complex is an object consisting of building blocks called simplices. Geometrically, a d-simplex is the convex hull spanned by $d+1$ points. Thus, a 0-simplex is a vertex, a 1-simplex is an edge, a 2-simplex is a triangle, a 3-simplex is a tetrahedron, etc. As we will see later, given that we are focus on reconstructing graphs, we will only deal with 2 dimensional simplicial complexes, consisting of vertices, edges and triangles.

1.1.2 Persistent Homology

While the topology of a simplicial complex can be interesting to study on its own, this work makes use of *persistent homology*. Persistent homology is a very important, modern extension of homology that has enabled the summarization of homology features across multiple scales with respect to certain evolution of the space (usually induced by a descriptor function). Instead of the homology of a fixed space, now we inspect and track the "birth" and "death" of topological (homological) features through the evolution of a space or a function. Examples of persistent homology are shown in Figure 1.1. Considering the first example of Figure 1.1, one can now capture not just the two independent holes in the space from which the points are sampled, but also their size or importance as well. This extension has significantly broadened the use of homological features, and its various variants and follow-up developments have formed the cornerstone of TDA. More information and explicit definitions relating to persistent homology can be found in Chapter 2.

1.1.3 Morse Theory

Morse theory studies Morse functions, which are intuitively well-behaved smooth functions of which all critical points of the function are non-degenerate. Critical points are points in the domain at which the gradient vanishes. The integral line is a path between two points

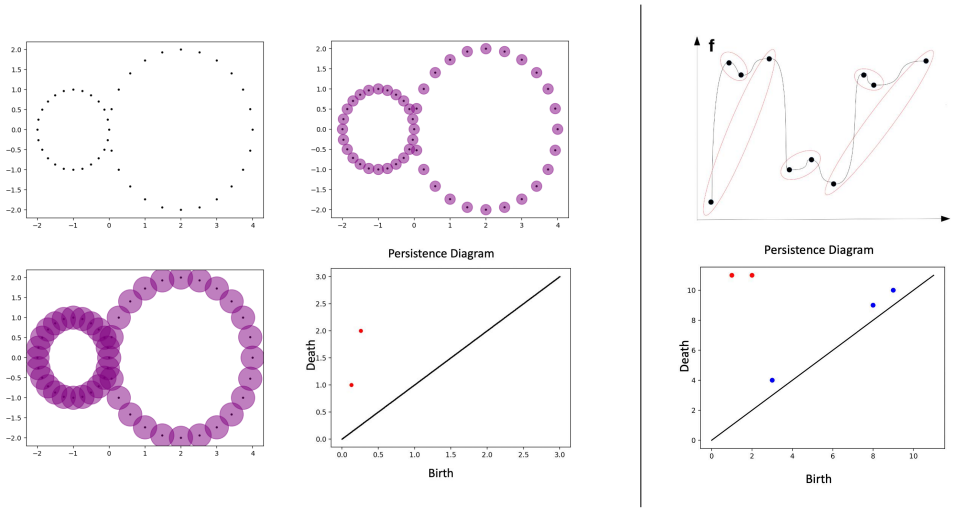


Figure 1.1. Two examples of persistent homology. On the left, circles with increasing radii are centered at each point, with the 1st dimension persistence diagram capturing two meaningful features. On the right, the domain is swept in increasing function order, with the 0th dimension persistence diagram capturing two significant features.

that agrees with the gradient along the entire path. Intuitively, integral lines (gradient flow) usually flow into minima. However, sometimes they may terminate at other types of critical points (saddle points of different indices). We are especially interested in the so-called *unstable 1-manifolds*, which are integral lines that "start" at maxima and "end" at saddle points of index $d-1$. Intuitively, these curves trace from mountain peaks (maxima) to saddles and back to peaks, separating different "basins" of the graph of the function (viewed as a terrain). Thus, unstable 1-manifolds can capture "mountain ridges" of the graph of an input scale function $f : \mathbb{R}^d \to \mathbb{R}$. An example of the unstable 1-manifolds is shown in Figure 1.2 (D). More information and explicit definitions related to Morse theory can be found in Chapter 2.

1.1.4 Discrete Morse Graph Reconstruction

The previous description suggests that one might be able to use the unstable 1-manifolds associated with a function $f : \mathbb{R}^d \to \mathbb{R}$ to capture the mountain ridges of the graph of this function (viewed as a terrain). In practice, however, we usually operate in a discrete setting where the domain of interest (e.g, a compact subset of $\mathbb{R}^d$) is approximated by a triangulation K. One can

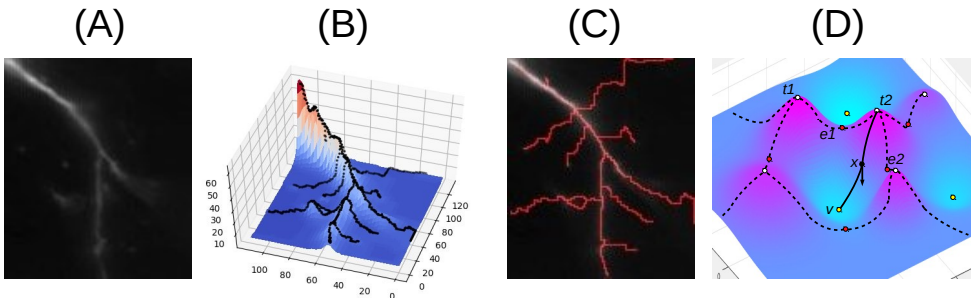


Figure 1.2. A practical example of DM graph reconstruction. (A) The image (downloaded from www.brainimagelibrary.org/) contains neuronal branches that we aim to reconstruct. (B) View the image as a density function, we show the graph of this function, and mountain ridges of this terrain. (C) These ridges capture potential neuronal branches in the image in (A). (D) Gradient and the integral line passing x. Dashed curves are union of unstable 1-manifolds.

aim to use piecewise-linear or higher-order approximation of f over K to compute the unstable manifolds. However, given that these are differential objects, they are sensitive to noise. In addition, usually the computation and simplification is non-trivial beyond 2-dimensional case. To this end, one can leverage the so-called *discrete Morse theory* (DMT) proposed by [51, 52]. DMT is not a "discretization" of the smooth Morse theory. Rather, this is a combinatorial analog of the classical Morse theory; see Chapter 2 for details. Due to the combinatorial nature, there are robust and efficient algorithms for both constructing and simplifying resulting unstable 1-manifolds.

A line of recent works have developed a persistence-guided discrete Morse theory based methodology to extract a graph from a given density field [38, 63, 123, 134, 147]. Given a density field, with density accumulated around an underlying graph one wishes to extract, one can take the mountain ridges of the density function to be the underlying graph. A practical 2D example is shown in Figure 1.2 to build intuition, but we note that this methodology works for higher dimensions as well. Figure 1.2 (A) shows a zoom-in on a splitting neuronal branch from a mouse brain image. In this case, the density function is defined as the pixel values. Viewing the graph of the density via plotting density function values in the third dimension, we observe the true graph we wish to capture corresponds with the mountain ridges of our new terrain (Figure

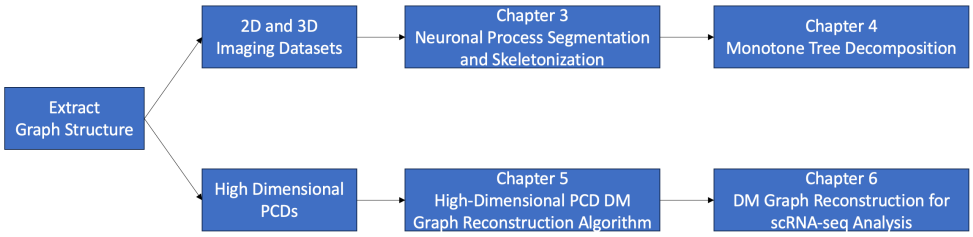


Figure 1.3. Outline for the contributions of this dissertation.

1.2 (B)). Projecting these mountain ridges onto the original image, we see that they entirely capture the neuron branch, including the splitting (Figure 1.2 (C)). Discrete Morse theory is used to capture the unstable 1-manifolds that represent the mountain ridges from the density function. More information, explicit definitions, and computational details relating to discrete Morse Graph Reconstruction can be found in Chapter 2.

Discrete Morse graph reconstruction has shown to be a robust method for extracting underlying graph structure from noisy datasets in many domains. One field in which the development of such methods is of particular interest is neuroscience. Reconstructing neurons from mouse brain imaging datasets is a well-studied problem in the neuroscience community [152, 65, 29]. Individual neurons can be represented as trees, and the entire neuron network within a single brain can be represented as a graph. On the higher dimensional front, scRNA-seq technology has exploded and there are now datasets with millions of cells and tens of thousands of genes [156, 157, 155, 120, 3, 74, 101]. Many tools for analyzing these datasets rely on dimensionality reduction [129, 66, 62], which is known to not be able to perfectly preserve true structure of raw, higher dimensional input data, with metric distortion being one of the many severe consequences. Extracting structure from the raw, noisy, higher dimensional space is required to properly understand and analyze the true nature of these datasets.

1.2 Contributions

The overall contribution of this dissertation is the development of graph extraction methodologies that are developed with neuroscientific problems in mind. The contributions

fall into two categories: (1) extracting graph structure from 2D and 3D imaging datasets and (2) extracting graph structure from high dimensional PCDs. An outline of our contributions is shown in Figure 1.3.

1.2.1 Extracting graph structure from 2D and 3D imaging datasets

Topologically Constrained Neuronal Cell Process Segmentation and Skeletonization

As we mentioned earlier, there has already been work for DM (discrete Morse)-based graph construction algorithms for 2D and 3D data [38, 63, 123, 134, 147]. Thus, our focus in Chapter 3 is to successfully apply the existing methodology to neuroanatomical datasets. Standard neural based segmentation and skeletonization methods are oblivious of topological structures, and can produce broken pieces where signal is weak. An explicit example is shown in Figure 1.4, which contains a 2D mouse brain image with several neuronal branches. One such branch has a drop in intensity. The standard UNET [124] framework is unable to detect the neuronal branch where intensity drops (Figure 1.4(B)). In contrast, DM graph reconstruction perfectly captures the neuronal branch in this region (Figure 1.4(C)). The goal is to use DM graph reconstruction to help produce topologically constrained segmentation and skeletonization outputs that better respect the true connectivity of neuronal branches. In joint work with researchers at Cold Spring Harbor Laboratory, we developed a neuronal process segmentation network for 2D mouse brain images, a single neuron skeletonization pipeline for 3D mouse brain images, a tracer injection summarization pipeline for 3D mouse brain images, and full brain skeletonization pipelines for both 2D and 3D mouse brain images.

We have four main contributions. First, for the neuron process segmentation network, we design a DM graph reconstruction pipeline for 2D mouse brain imaging datasets that outputs a gray-scale mask representing the Morse graph. These gray-scale masks are integrated with a novel neural process segmentation framework that we co-developed with collaborators at Cold Spring Harbor Laboratory [4]. Additionally, the highly persistent vertices on the Morse graph are used for bouton detection [4], which are high-intensity balls that appear along neuronal branches.

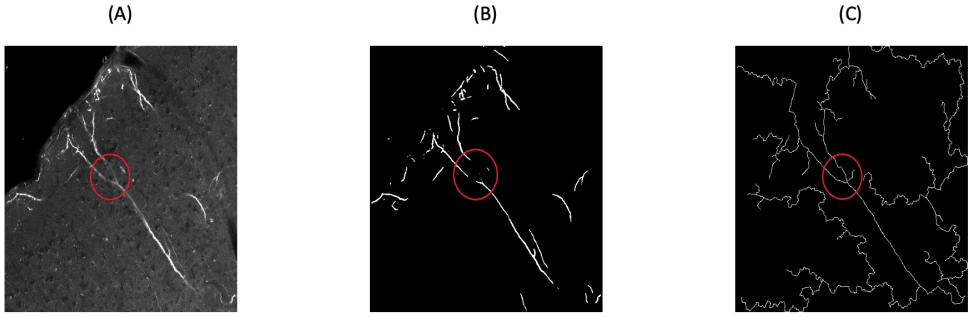


Figure 1.4. (A) 2D mouse brain image taken (downloaded from www.brainimagelibrary.org/). (B) UNET neuronal process segmentation output of image in (A). (C) Gray-scale mask of DM graph reconstruction output of image in (A). The red circle highlights a neuron branch in the original image (A) that is not connected in the UNET output (B) but is captured by DM graph reconstruction (C).

Second, we design a DM graph reconstruction pipeline for 3D mouse brain imagining datasets. These Morse graphs are the starting point for a 3D single neuron skeletonization pipeline that we co-developed with collaborators at Cold Spring Harbor Laboratory [146]. Third, we design a DM graph reconstruction pipeline for 3D tracer injection imaging datasets. These Morse graphs are used in a larger pipeline for summarizing the distribution of neuronal signal that we co-developed with our collaborators at Cold Spring Harbor Laboratory [146]. Finally, we developed full brain skeletonization pipelines for both 2D and 3D mouse brain imaging datasets - a significant step toward full brain neuronal process skeletonization.

Monotone Tree Decomposition

The ultimate goal for high-resolution full mouse brain imaging datasets would be to extract each individual neuron in the entire brain. With pipelines now developed to extract the full neuronal network graph from mouse brain imaging, it is natural to ask if one could decompose the full graph into individual neurons. We explore this question in Chapter 4.

More specifically, we first generalize the above problem. A monotone tree - a tree with a function defined on its vertices that decreases the further one travels from its root - is a natural model for an acyclic process that weakens the further one travels from its source -

such as the individual neurons in the mouse brain imaging datasets. The full brain DM graph reconstruction output represents an aggregation of all such trees. Thus, given a graph representing the aggregation of monotone trees, we wish to decompose the graph into individual monotone trees. A polynomial time algorithm exists for computing the minimum cardinality collection of monotone trees, which we refer to as an M-Tree Set, when the input is restricted to a density tree, but no such algorithm exists when the input is a density graph that may contain cycles.

We have two main contributions. First, we prove that extracting such minimum M-Tree Sets of density graphs is NP-Complete, as well as prove three additional variations of the problem - such as the minimum M-Tree Set such that the intersection between any two monotone trees is either empty or contractible (SM-Tree Set) - are also NP-Complete. Second, on the algorithmic front, we provide several approximation algorithms, concluding with a 3-approximation algorithm for computing the minimum SM-Tree Set for density cactus graphs.

1.2.2 Extracting graph structure from high-dimensional PCDs and scRNA-seq Datasets

DM Graph Reconstruction for High-Dimensional PCDs with Theoretical Justification

The existing DM graph reconstruction algorithm does not work well on high-dimensional PCDs. It is far too expensive to compute a triangulation of the ambient space for high-dimensional PCDs, and even then it can be non-trivial to compute a triangulation that appropriately approximates the space such that all meaningful topological features can be accounted for.

In Chapter 5, we aim to develop an efficient and effective graph skeletonization algorithm for PCDs. We have four main contributions. First, we propose a new algorithm by changing our perspective of the DM graph reconstruction algorithm from a density perspective to a filtration perspective. This generalized algorithm greatly increases the flexibility of this procedure. Second, we provide theoretical justification for this new algorithm by proving output graphs contain a so-called lex-optimal persistent cycle basis of the input filtration. This proves that the output graph contains meaningful information with respect to the input. Next, we design a practical filtration

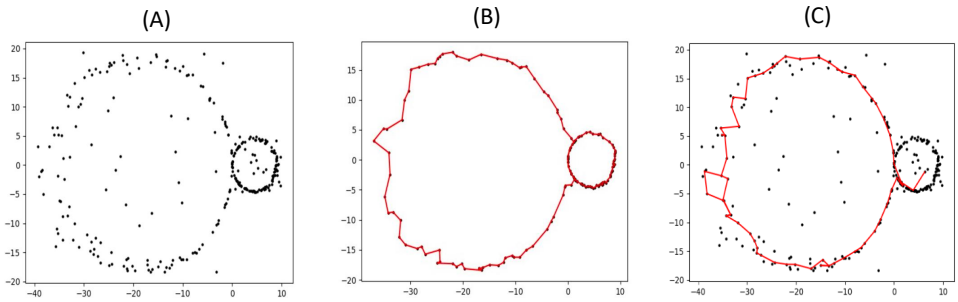


Figure 1.5. (A) A 2D PCD with points forming two circles of different scales and additional white noise. (B) Output of our PCD DM graph reconstruction method at persistence threshold $\delta = 1.2$. (C) A graph reconstruction of the dataset, using a baseline density-based DM graph reconstruction approach, detailed in Chapter 5, that fails to capture both features.

for high dimensional PCDs. We use the sparsified weighted Rips filtration of [16] as input for the generalized algorithm to create a DM graph reconstruction algorithm for PCDs. Finally, we include several empirical results to demonstrate both the effectiveness and efficiency of our high-dimensional PCD DM graph reconstruction algorithm. An example is shown in Figure 1.5, which shows our algorithm correctly extracting the graph skeleton of a PCD that contains two circles of different scales with additional white noise (Figure 1.5. Figure 1.5(C) contains a failed graph reconstruction performed by a baseline density-based DM graph reconstruction algorithm. This failed reconstruction highlights the need for our high-dimensional PCD DM graph reconstruction algorithm. It is not practical to adapt the density-based DM graph reconstruction algorithm to extract structure from PCDs. Challenges such an approach faces include difficulty choosing an appropriate scheme for building an input triangulation, running-time concerns should an appropriate triangulation be particularly large, and dealing with points sampled at non-uniform resolutions. Our high-dimensional PCD algorithm can successfully extract meaningful graph structure, with the sparification of the input filtration also keeping the algorithm efficient.

Discrete Morse Graph Reconstruction for High-Dimensional Transcriptomic Data

The advancement and use of single-cell and spatial genomics techniques in neuroscience have facilitated the large-scale creation of high-dimensional datasets and brain atlases across

various species [3, 120, 74, 156]. Single-cell RNA sequencing (scRNA-seq) is now commonly employed to explore anatomical transcriptomic structure [156, 157, 155, 120, 95], as well as developmental stages [138], evolutionary trajectories [108], disease progression [101], and other experimental scenarios. Analyzing these datasets requires thorough investigation and quantification of the transcriptomic relationships among various brain features, which is crucial for understanding their roles across different structures and biological contexts [36]. Goals of such data analysis include identifying individual cell types, examining gene expression gradient between different cell types, developing cell type taxonomies to explicitly define cell type relationships, and quantifying gene expression changes over disease development cycles.

Analysis of scRNA-seq datasets faces all of the challenges of high-dimensional data analysis. Manifold learning methods such as [99, 139, 84, 125, 135, 9, 46], as well as standard dimensionality reduction techniques like principal component analysis [53], are used in many scRNA-seq data analysis tools [73]. However, because its has been theoretically suggested [78] and experimentally demonstrated [35] that these techniques fail to preserve structure, extracting graph structure from the high-dimensional raw gene space is needed to analyze the true structure of scRNA-seq datasets.

In Chapter 6, we aim to develop a graph reconstruction algorithm to extract meaningful graph structure specifically from scRNA-seq datasets. We have two main contributions. First, we develop an efficient and effective algorithm to construct the graph skeleton for high-dimensional scRNA-seq datas, which are ultimately very high-dimensional, noisy PCDs. To this end, we note that our high-dimensional PCD algorithm presented in Chapter 5 does not produce good results when directly applied to scRNA-seq data, potentially due to the very high dimensionality and high level of sparsity, as well as non-uniform sampling density of gene expression data across different cell types. Instead, we combine the generalized algorithm in Chapter 5 with a lower-star filtration with respect to Jaccard index to develop a graph extraction algorithm for scRNA-seq datasets.

In our second contribution, we carry out various experiments both to validate that DM

graph reconstruction output graphs are indeed meaningful structure and an effective tool for data analysis of scRNA-seq datasets. We provide several experiments to demonstrate that Morse graphs are (1) a more accurate representation of distance in the higher dimensional space than lower dimensional embeddings, (2) compact, faithful representations of the entire input datasets, and (3) contain important dataset exemplars. On the data analysis, we examine DM graph reconstruction output graphs to define cell type identity, study gene expression gradient between different cell types, and construct cell type taxonomies to define cell type relationships. We conclude by using DM graph reconstruction outputs to quantify the difference between cell type loss and gene expression changes along Alzheimer's disease progression.

Chapter 2

Preliminaries

We now briefly introduce some notions needed to describe the idea behind the DM-graph algorithm of [147, 40]. In this dissertation, we will use the **simplicial setting**, where the space of interest is modeled by a *simplicial complex* K, consisting of basic building blocks called *simplices*. Intuitively, a geometric d-simplex is the convex combination of $d+1$ affinely independent vertices: a 0-, 1-, 2-, or 3-simplex is just a vertex, an edge, a triangle, or a tetrahedron, respectively. Ignoring the geometry, *an abstract d-simplex* $\sigma = (v_0, \ldots, v_d)$ is simply a set of $d+1$ vertices. Any subset τ of the vertices of a d-simplex σ is *a face* of σ, and τ is called a *facet* of σ if its dimension is $d-1$. A simplicial complex K is a collection of simplices with the property that if a simplex σ is in K, then any of its face must be in K as well. Given a simplicial complex K, its *q-skeleton* K^q consists of all simplices in K of dimension at most q.

2.1 Persistent Homology

Instead of introducing persistent homology in its full general form, below we focus on the simplicial complex setting. See e.g., [47, 28] for more detailed exposition.

Boundaries, cycles, homology groups.

Given a simplicial complex K, let K^q denote the set of q-simplices of K. Under $\mathbb{Z}_2$ field coefficient (which we use throughout this dissertation), a q-chain $C = \sum_{\sigma \in K^q} c_\sigma \sigma$ where $c_\sigma \in \{0,1\}$; equivalently C is a subset of K^q (those with $c_\sigma = 1$). The set of q-chains together

with addition operation gives rise to the so-called *q-th chain group* $\mathsf{C}_q(K)$. Given any q simplex σ, its boundary $\partial_q\sigma$ consists of all of its faces of dimension q-1. This in turn gives a linear map, called the q-th boundary map $\partial_q : \mathsf{C}_q(K) \to \mathsf{C}_{q-1}(K)$, where $\partial_q C = \sum_{\sigma\in K^1} c_\sigma \partial_q(\sigma)$ for any q-chain $C = \sum_{\sigma\in K^q} c_\sigma\sigma$. A q-chain C is a q-cycle if its boundary $\partial_q C = 0$. The collection of all q-cycles form the *q-th cycle group* Z_q; that is, $\mathsf{Z}_q = \text{kernel } \partial_q$. A q-chain C is a q-boundary if it is the image of some (q+1)-chain C'; i.e., $C = \partial_{q+1}C'$. The collection of q-boundaries form the *q-th boundary group* B_q; that is, $\mathsf{B}_q = \text{image } \partial_{q+1}$. By the fundamental property of boundary map, i.e, $\partial_q \circ \partial_{q+1} = 0$, it follows that B_q is a subgroup of Z_q. The *q-th homology group* H_q is defined as $\mathsf{H}_q = \mathsf{Z}_q/\mathsf{B}_q$. In particular, given any q-cycle C, its homology class $[C]$ is the equivalent class of all q-cycles in $q + \mathsf{B}_q(K)$; and two q-cycles C_1, C_2 are homologous if $[C_1] = [C_2]$, implying that $C_1 + C_2$ is a boundary (i.e, $C_1 + C_2 \in \mathsf{B}_q(K)$). The *q-th homology classes* intuitively capture q-dimensional "holes" in K; i.e., connected components (0D), loops (1D), closed surfaces that are not "filled" (2D) and their higher dimensional analogs. The qth homology group is the vector space spanned by such topological features, and its rank, called the *q-th Betti number* $\beta_q(K)$, gives the number of independent topological "holes".

Filtration, persistent modules.

Suppose we have a finite sequence of simplicial complexes connected by inclusions, called a *filtration of K*, denoted by $\mathscr{F} : K_1 \subseteq K_2 \subseteq \cdots K_m = K$. Applying the homology functor to this sequence (with $\mathbb{Z}_2$ coefficients), we obtain a sequence of vector spaces (over field $\mathbb{Z}_2$) connected by linear maps induced from inclusions, which is called a persistence module; in particular, for any dimension $q \geq 0$, we have:

$$\mathbb{P}\mathscr{F} : \quad \mathsf{H}_q(K_1) \to \mathsf{H}_q(K_2) \to \cdots \to \mathsf{H}_q(K_m).$$

where maps are induced by inclusions. We will assume that the persistence module is indexed by a finite set $[1,m]$ instead of $\mathbb{Z}$ for the remainder of the dissertation.

A special class of persistence modules is the so-called *interval modules*. (i) $I_i = \mathbb{Z}_2$

for any $\ell \in [s,t]$ and $I_i = 0$ otherwise; and (ii) $v^{i,j}$ is identity map for $s \leq i \leq j \leq t$ and 0 map otherwise. We abuse the notation slightly and allow $t = \infty$, in which case the interval is really $[s,\infty)$. A pictorial version of an interval module is as follows:

$$\cdots \to 0 \to \mathbb{Z}_2 \to \mathbb{Z}_2 \to \cdots \mathbb{Z}_2 \to 0 \to \cdots .$$

Persistence diagram.

It turns out that a given persistence module $\mathbb{V}$ can be uniquely decomposed into direct sums of interval modules (up to isomorphisms) $\mathbb{V} = \bigoplus_{[b,d]\in J} \mathbb{I}^{[b,d]}$, where J is a multiset of intervals $J = \{[b,d]\}$. We call $\bigoplus_{[b,d]\in J} \mathbb{I}^{[b,d]}$ the interval decomposition of $\mathbb{V}$. Again, note that the intervals in J could be of two forms: $[b,d]$ for finite $b,d \in \mathbb{Z}$, and $[b,\infty)$; the former is called a *finite interval*. Note that each interval $[b,d]$ can also be viewed as a point in $\mathbb{R}^2$. Given a filtration $\mathscr{F}$, its *persistence diagram* $\mathrm{dgm}\mathscr{F}$ is the multiset of points in J where $\mathbb{P}\mathscr{F} = \bigoplus_{[b,d]\in J} \mathbb{I}^{[b,d]}$ is the interval decomposition of $\mathbb{P}\mathscr{F}$. Each point in J is called a *persistence point*. Assuming that we are given a monotone function $f : \mathbb{Z} \to \mathbb{R}$, then the persistence of $\mathsf{p} = [b,d] \in \mathrm{dgm}\mathscr{F}$ w.r.t. f is defined as $\mathrm{pers}(\mathsf{p}) = f(d) - f(b)$ [1]. To make the dependency on the function f explicit, we now write the filtration together with this function as $\mathscr{F}_f$, and the persistence diagram is denoted by $\mathrm{dgm}\mathscr{F}_f$. For example, a common choice of f in the literature is simply $f(i) = i$.

Simplex-wise setting.

In the remainder of this dissertation, we assume that we are given a *simplex-wise filtration* $\mathscr{F}$ of K, such that there is an ordering of all simplices in K, $\sigma_1, \ldots, \sigma_N$, and the filtration is given by:

$$\mathscr{F} : \emptyset = K_0 \subset K_1 \subset \cdots \subset K_N = K, \text{ where } K_i := \{\sigma_1, \ldots, \sigma_i\}. \tag{2.1}$$

[1] We note that in the literature, the persistence of a pair is often defined using some indices ($\mathbb{Z}$ or $\mathbb{R}$) of the filtration. Here we decouple the two to make the presentation cleaner.

Suppose we are also given a monotone function $\rho : [1,N] \to \mathbb{R}$ (i.e, $\rho(j) \geq \rho(i)$ for $j > i$), which we use to define the persistence of points in the persistence diagram $\mathrm{dgm}\mathscr{F}_\rho$. (If no function ρ is explicitly given, we take ρ to be $\rho(i) = i$.)

Furthermore, note that for any i, K_i is obtained by adding σ_i to K_{i-1}. Let $\mathsf{ind} : K \to [1,N]$ be this bijection, where we set $\mathsf{ind}(\sigma_i) = i$. That is, $\mathsf{ind}(\sigma)$ in general is the index of simplex σ in the ordered sequence of simplices that induce simplex-wise filtration $\mathscr{F}$; or, the time it will be inserted into a complex (i.e $K_{\mathsf{ind}(\sigma)}$) in the filtration. Given this bijection, a function on the simplices in K also gives rise to a function on $[1,N]$. In what follows, for convenience, we do not differentiate a function on simplices in K and a function on $[1,N]$; that is, $\rho(\sigma) = \rho(\mathsf{ind}(\sigma))$, and if simplices are ordered as in Eqn (2.1), then $\rho(\sigma_i) = \rho(i)$. If ρ is defined on simplices in K, we also call it a *simplex-wise function* $\rho : K \to \mathbb{R}$.

Given any persistence point $[b,d] \in \mathrm{dgm}\mathscr{F}_\rho$ with $b,d \in [1,N]$, we say its corresponding *persistence pair* is (σ_b, σ_d) and it is necessary that $dim(\sigma_d) = dim(\sigma_b) + 1$. We set $\mathrm{pers}(\sigma_b) = \mathrm{pers}(\sigma_d) = \mathrm{pers}([b,d]) = \rho(d) - \rho(b)$. If $d = \infty$, then we say σ_b is unpaired, and $\mathrm{pers}(\sigma_b) = \rho(\infty) := \infty$. Finally, consider each persistence pair (σ, τ), we say that σ is *positive* and τ is *negative*, as the q-simplex σ will create a new homology class that will become trivial (be killed) when the (q+1)-simplex τ is added to the filtration.

A common way to induce a filtration is via a descriptor function $\rho : V(K) \to \mathbb{R}$ given at vertices $V(K)$ of K. For simplicity of presentation, assume that ρ is *injective*. We can extend ρ to a simplex-wise function $\rho : K \to \mathbb{R}$ by setting $\rho(\sigma) = max_{v \in \sigma} \rho(v)$. Consider an ordering of simplices $\mathscr{S}_\rho : \sigma_1, \ldots, \sigma_m$ that is *consistent with* ρ; i.e, (i) $\rho(\sigma_i) \leq \rho(\sigma_j)$ for any $i \leq j$ and (ii) for any simplex σ_i, its faces appear before it in the ordering. This order induces the so-called *lower-star filtration* $\mathscr{F}_\rho$ w.r.t. ρ. That is, assume $\rho(v_1) < \ldots < \rho(v_n)$. Intuitively, we inspect the domain in increasing values of ρ and the lower-star filtration is obtained by adding each vertex v_i and its lower-star (simplices incident on v_i with function value at most $\rho(v_i)$) in ascending order of i. The persistence diagram $\mathrm{dgm}_p\mathscr{F}_\rho$ encodes birth and death of features during this course. In this case, we modify the persistence to reflect function values: For a persistence point

$(b,d) \in \mathrm{dgm}_p \mathscr{F}_\rho$, we set $\mathrm{pers}((b,d)) = \mathrm{pers}((\sigma_b,\sigma_d)) := |\rho(\sigma_d) - \rho(\sigma_b)|$. Features with large persistence survive for a long range of function values and are considered as more important w.r.t. ρ.

2.2 Discrete Morse Theory

Below we very briefly introduce some concepts from discrete Morse theory, so that we can introduce both the original algorithm of [147] (to provide intuition) and the simplified algorithm of [40]. See [51, 52] for more detailed exposition of discrete Morse theory.

We again consider the simplicial complex setting. Given a simplicial complex K, a *discrete gradient vector* is a combinatorial pair of simplices (σ^q, τ^{q+1}) where σ is a face of τ of co-dimension 1 (i.e, σ is a vertex of an edge τ, or an edge of a triangle τ), and we sometimes include the superscript to make its dimension explicit. Given a collection $M(K)$ of such discrete gradient vectors over K, a *V-path* is a sequence of simplices of alternating dimensions: $\sigma_1^q, \tau_1^{q+1}, \ldots, \sigma_\ell^q, \tau_\ell^{q+1}, \sigma_{\ell+1}^q$ such that for each $i \in [1,\ell]$, we have (1) $(\sigma_i^q, \tau_i^{q+1}) \in M(K)$ and (2) σ_{i+1}^q is a face of τ_i^{q+1}. We say that a V-path as above is a *non-trivial closed V-path* (or *cyclic*) if $\sigma_1 = \sigma_{\ell+1}$; otherwise, it is *acyclic*.

Definition 2.2.1 (Discrete Morse gradient vector field)**.** *A collection of discrete gradient vectors $M(K)$ of K is a* discrete Morse gradient vector field*, or* DM-vector field *for short, if (i) any simplex in K is in at most one vector in $M(K)$; and (ii) no V-path in $M(K)$ is cyclic.*

A simplex in K is critical *w.r.t. a DM-vector field $M(K)$ if it does not appear in any gradient vector in $M(K)$.*

Now suppose we are given a critical edge e in $M(K)$. The *stable 1-manifold* of e is the union of vertex-edge V-paths $v_1, e_1, \ldots, v_\ell, e_\ell, v_{\ell+1}$ such that v_1 is an endpoint of e, while $v_{\ell+1}$ is a critical vertex. Such stable 1-manifolds correspond to the "valley ridges" in a continuous function $f : \mathbb{R}^d \to \mathbb{R}$ (the graph of which can be viewed as a terrain), connecting index-1 saddles with minima. They are the opposite of "mountain ridges" (unstable 1-manifolds), connecting

saddles to maxima and separating different valleys.

Finally, we note that there is a *Morse cancellation* operation that allows one to cancel a pair of critical simplices, and thus reduce both the number of critical simplices as well as the complexity of (un)stable 1-manifolds. In particular, a pair of critical simplices $\langle \sigma^q, \tau^{q+1} \rangle$ is *cancellable* if there is a unique V-path $\sigma_1, \tau_1 ..., \sigma_\ell, \tau_\ell, \sigma_{\ell+1} = \sigma^q$ in $M(K)$ such that σ_1 is a face of τ^{q+1}. The Morse cancellation operation will essentially invert the gradient vectors along this V-path and render σ^q and τ^{q+1} no longer critical afterwards.

2.3 Graph Reconstruction Algorithm for Density Field Based on Morse Theory

Below we first introduce the intuition behind the original discrete Morse based graph reconstruction algorithm from density field by [134, 147] in the smooth setting. We will then describe the discrete setting, and its simplification DM-graph by [40]. First, assume we are given a smooth function $\rho : \Omega \to \mathbb{R}$ on a hypercube Ω in $\mathbb{R}^d$. View ρ as a density function which concentrates around a hidden geometric graph (e.g, Figure 1.2 (A) where $\Omega \subset \mathbb{R}^2$). Consider the graph of this function $\{(x, \rho(x)) \mid x \in \Omega\}$, which is a terrain in $\mathbb{R}^{d+1}$ and which we will refer to as the terrain of ρ; see Figure 1.2 (B). Intuitively, the "mountain ridge" of this terrain identifies the hidden graphs, as locally on the hidden graph, the density should be higher than points off it. To capture these mountain ridges, one can use the so-called *unstable 1-manifolds* of the function ρ as in [134, 147].

Roughly speaking, given ρ, the gradient vector at $x \in \Omega$, $\nabla\rho(x) = -[\frac{\partial \rho}{\partial x_1}(x), \ldots, \frac{\partial \rho}{\partial x_d}(x)]^T$, indicates the steepest descending direction of ρ at x. See Figure 1.2 (D). *Critical points* of ρ are points whose gradient vector vanishes. For a smooth function on d-D domain, non-degenerate critical points include minima, maxima, and d-1 types of saddle points. An integral line is intuitively the flow-line traced out by following the gradient direction at every point. Flow-lines (integral lines) start and end (in the limit) at critical points. The *unstable 1-manifold* of a saddle

(of index d-1) is the union of flow-lines starting at some maximum and ending at this saddle. Intuitively, unstable 1-manifolds connect mountain peaks to saddles to peaks, separating different valleys (around minima), and thus can be used to capture mountain ridges.

Hence one can compute the union of unstable 1-manifolds of ρ as its graph skeleton. Furthermore, the density map ρ may be noisy. To denoise the graph skeleton, previous approaches use persistent homology to keep only unstable 1-manifolds corresponding to "important" saddles.

Algorithm in the discrete setting.

In the discrete setting imagine K is the 2-skeleton of a domain Ω of interest, ρ is a density function defined on Ω but is only accessible at vertices $V(K)$ of K, i.e., $\rho : V(K) \to \mathbb{R}$. Algorithm $\mathsf{firstDM\text{-}graph}(K, \rho : V(K) \to \mathbb{R}, \delta)$ will output a graph consisting of edges of K capturing a graph skeleton of the density field ρ by the following three steps:

- (Step 1): Compute persistence pairing $\mathscr{P}$ induced by the lower-star filtration w.r.t. -ρ.

  Specifically, we use $f = -\rho$ as it is easier to algorithmicly compute the discrete analog of "valley ridges" using discrete Morse theory than "mountain ridges" – The valley ridges are the stable 1-manifolds (vertex-edge V-paths) for critical edges, and thus only 2-skeleton of input complex K is needed. To compute the importance of critical points in the simplicial setting when we are given $f : V(K) \to \mathbb{R}$, we use the standard lower-star filtration to simulate the so-called sublevel-set filtration in the smooth case. In particular, given $f : V(K) \to \mathbb{R}$, let $v_1 \dots v_n$ be the set of vertices in K sorted in non-decreasing order of f values. Given any vertex $v_i \in V(K)$, its *lower-star* $\mathsf{lowSt}(v_i)$ consists of the set of simplices incident on v_i spanned by only vertices from $V_i := \{v_1, \dots, v_i\}$. The lower-star filtration w.r.t. f is the following:

$$\widehat{K}_1 \subset \widehat{K}_2 \subset \cdots \widehat{K}_n = K; \quad \text{where } K_i = K_{i-1} \cup \mathsf{lowSt}(v_i). \tag{2.2}$$

  Equivalently, we can think that this filtration is induced by a simplex-wise function

$\hat{f} : K \to \mathbb{R}$ where $\hat{f}(\sigma) = \max_{\text{vertex } v \text{ of } \sigma} f(v)$.

- (Step 2): Initialize vector field $M(K)$ to be the trivial one where all simplices are critical. Then in order of increasing persistence, for each pair $(\sigma, \tau) \in \mathscr{P}$ with $\text{pers}(\sigma, \tau) \leq \delta$, perform discrete Morse cancellation and update $M(K)$ if possible. Intuitively, this is to simplify and remove "not-important" critical points.

- (Step 3): Output the graph $G_\delta = \bigcup_{e \in K, \text{pers}(e) > \delta} \{$ stable 1-manifold of $e\}$.

  In particular, we only consider critical edges that are "important" (i.e., pers $> \delta$). Then we trace the valley ridges (stable 1-manifolds) connecting them to minima. These minima - which have persistence greater than δ - are the topographically prominent peaks of ρ.

Simplified algorithm.

It turns out that algorithm firstDM-graph() can be significantly simplified [40]. In particular, one does not need to explicitly maintain any discrete Morse gradient vector field at all. See Algorithm DM-graph() below.

Algorithm 1: DM-graph(K, ρ, δ)

Input: Triangulation K, density function $\rho : V(K) \to \mathbb{R}$, persistence threshold δ
Output: a graph skeleton G_δ
(Step 1) Compute persistence pairing $\mathscr{P}$ induced by the lower star filtration w.r.t. -ρ,
(Step 2) Set $\mathscr{T}_\delta := \{e \in E \mid e \text{ is negative and } \text{pers}(e) \leq \delta\}$
For each component (tree) T in $\mathscr{T}_\delta$, set its root to be $r(T) := \text{argmin}_{v \in T}$ -$\rho(v)$.
(Step 3) Let $\pi_T(x, y)$ be the tree path from x to y in a tree T. Output:

$$G_\delta = \bigcup_{e=(u,v), \text{pers}(e) > \delta} \{e \cup \pi_{T_{i_1}}(u, r(T_{i_1})) \cup \pi_{T_{i_2}}(v, r(T_{i_2})) \mid u \in T_{i_1}, v \in T_{i_2} \text{ in } \mathscr{T}_\delta\}. \tag{2.3}$$

In particular, in (Step 3) above, we only consider critical edges with pers $> \delta$, and their stable 1-manifolds turn out to be the union of tree paths as specified in Eqn (2.3). Note that (Step 2, 3) can be implemented in time linear to the number of vertices and edges in K.

2.4 Reprint

Chapter 2, in full, is a reprint of the Preliminaries section as it appears in Graph skeletonization of high-dimensional point cloud data via topological method in Journal of Computational Geometry. Magee, Lucas; Wang, Yusu. 2022. The author of the dissertation was the primary investigator and the author of this article.

Chapter 3

Extracting Neuronal Trees from Mouse Brain Imaging Datasets

In this chapter, we will apply the already developed DM graph reconstruction algorithm (Algorithm 1) to extract meaningful graph structure from neuroscientific datasets. More specifically, we will develop pipelines for extracting Morse graphs from both 2D and 3D mouse brain imaging datasets. These pipelines are then used to develop several pipelines for neuronal process segmentation, neuron reconstruction, and tracer injection summarization with collaborators at Cold Spring Harbor Laboratory.

3.1 Introduction

Analyzing the structure and connections of neurons is vital for comprehending brain circuitry. This requires accurately labeling neuronal process in whole brains. Historically, this was manually done with a microscope - a labor intensive process prone to human error. More recently, image visualization techniques have advanced to enable visualization of full brain imaging datasets at a subcellular resolution [13, 45, 119]. However, these imaging techniques have enabled the quick generation of large-scale, high-resolution datasets [109, 115, 92], making manual annotation no longer viable due to the amount of data that needs to be annotated.

The analysis of neuroanatomical imaging datasets specifically benefited from machine learning based methodologies. Previous research has focused on reconstructing neuronal struc-

tures from electron microscopy data and image stacks [64, 72, 111, 112, 121, 29, 54, 152, 65]. Although such methodologies are quite effective, there are still weaknesses inherent in machine learning approaches that are very apparent in the analysis of neuroanatomical imaging datasets. Such approaches do not factor in the inherent structure and connectivity of individual neurons, which are tree-like in nature. Considering that these same approaches rely heavily on local information to make decisions and outputs produced by such approaches do not respect the tree structure of individual neurons and the connectivity of their branches, there is a need for methodologies that respect neuronal structure.

In contrast to traditional machine learning based approaches, topological methods capture global connectivity and will naturally respect the true structure of neurons. In particular, DM graph reconstruction has already been applied successfully to different domains, such as road network reconstruction [147], where respecting the connectivity of the input data is imperative. An example of where DM graph reconstruction can be utilized in mouse brain imaging is shown in Figure 1.4. Figure 1.4(A) shows a 2D mouse brain image, circling a clear neuron branch that happens to have lower pixel values than other neuron branches in the image. UNET [124], a machine learning image segmentation network, fails to identify the neuronal branch in this low intensity region (Figure 1.4 (B)), because the architecture relies on local information. In contrast, the DM graph reconstruction output (Figure 1.4(C)) , which is computed with global information, accurately captures the neuronal branch.

In this chapter, we apply DM graph reconstruction to mouse brain imaging datasets. In particular, we collaborated with Cold Spring Harbor research scientists to produce several pipelines for analyzing mouse brain imaging datasets with DM graph reconstruction.

The first pipeline is a Siamese neural network architecture for neuronal process segmentation of 2D mouse brain images that takes gray-scale masks of DM graph reconstruction outputs as input. Additionally, bouton detection is performed by intersecting the highly persistent nodes on the Morse graph with the final process detection output. Our contribution to this pipeline is strictly the DM graph reconstruction output masks and the bouton detection. The complete

pipeline and results have been published in [4].

The second pipeline reconstructs individual neurons from 3D mouse brain imaging. The starting point of the pipeline is the DM graph reconstruction output of the 3D input mouse brain image, which is our contribution to this pipeline. The complete pipeline and results have been published in [146].

The third pipeline constructs trees to summarize tracer injection datasets, which labels thousands of neurons are collectively instead of individually and thus has a more complicated topological structure than individual trees. The starting point of this pipeline is also the DM graph reconstruction output of the 3D input mouse brain image, which is our contribution to this pipeline. The complete pipeline and results have been published in [146].

The final pipelines are full brain skeletonization pipelines for 2D and 3D imaging datasets. These pipelines are publicly available on Github.

3.2 DM Graph Reconstruction for 2D Neuronal Process Detection

3.2.1 Neuronal Process Segmentation

We develop a pipeline that produces gray-scale masks of DM graph reconstruction outputs of 2D mouse brain imaging. We first apply a Gaussian filter to the input image to smooth pixel values. Next, we perform DM graph reconstruction on the Gaussian smoothed input image, taking pixel values to be the density function. A mask is created by assigned each pixel that is part of the DM graph reconstruction output the average pixel of the path that it is part of (maximum value assigned if a pixel corresponds to a junction node on the graph). This pipeline is shown in Figure 3.1, which was originally published in [4].

The DM graph reconstruction output is a poor image segmentation output on its own for a few reasons. First and most importantly, it contains many false positives. Secondarily, there is also no thickness to the graph, and thus thickness of the branches is also not captured. Although

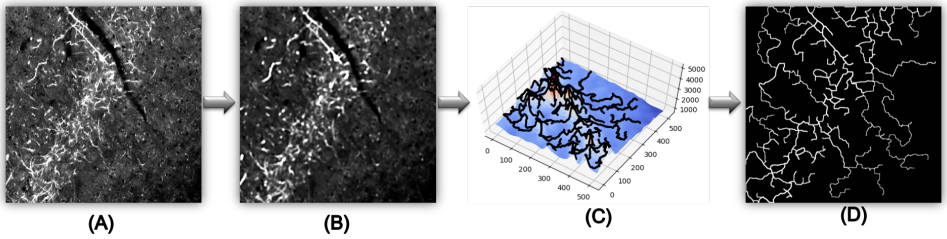


Figure 3.1. (A) 2D mouse brain image on which we wish to capture neuronal process. (B) Input image after Gaussian smoothing. (C) Pixel values from Gaussian smoothed image are projected into the third dimension. We observe that the neuronal process corresponds to the mountain ridges of this terrain. DM graph reconstruction extracts these mountain ridges. (D) The final gray-scale mask of the DM graph reconstruction output. This diagram was originally published in [4].

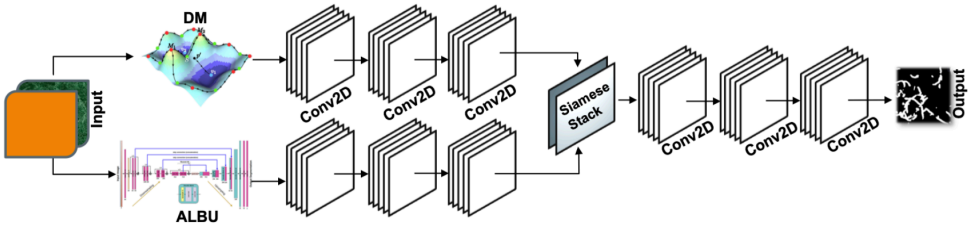


Figure 3.2. The DM++ architecture contains a Siamese network that takes process detection output from traditional machine learning network, such as ALBU, and DM graph reconstruction masks as inputs to produce a final process detection output. This diagram, and the architecture it is conceptualizing, were originally published in [4].

machine learning methodologies do not capture the underlying connectivity of neuronal branches, do not have these downsides. Thus, researchers at Cold Spring Harbor Laboratory developed a Siamese network architecture, named DM++, that takes our gray-scale masks as one input, and the process detection output of ALBU [17] as another input. The architecture, which has been published in [4], is shown in Figure 3.2.

In [4], neuronal process detection is performed on three mouse brain imaging datasets - monochrome Serial Two Photon Tomographic images (STP), 3-color fluorescent WSI images from the Mouse Brain Architecture project (MBA), and Bright Field Images (BFI), in [4]. Precision, recall, and F1 scores of DM++, ALBU [17], UNET [124], and an unsupervised methodology are shown in Table 3.1. DM++ outperformed all other methodologies on all

datasets in precision, recall, and F1. An example of where DM++ improves the connectivity of the neuronal process segmentation output is shown in Figure 3.3. While ALBU produces a segmentation that disconnects a branch with lower intensity than other branches, the DM graph reconstruction mask perfectly captures the branch. This enables DM++ to produces a final segmentation that perfectly capture the branch.

Table 3.1. Precision, Recall, and F1 values for neuronal process segmentation on three different mouse brain imagining datasets (STP, MBA, and BFI) using UNET, ALBU, DM++, and an unsupervised baseline technique. The unsupervised technique for Process Detection involves intensity-based thresholding with hard-coded parameters for threshold and morphological operations. These results were originally published in [4].

Model	**Precision**	**Recall**	**F1**
Process Detection - STP			
Unsupervised	0.61	0.63	0.62
UNET	0.85	0.88	0.86
ALBU	0.90	0.88	0.89
DM++	**0.92**	**0.93**	**0.92**
Process Detection - MBA			
Unsupervised	0.41	0.51	0.46
UNET	0.67	0.73	0.70
ALBU	0.79	0.82	0.81
DM++	**0.83**	**0.84**	**0.84**
Process Detection - BFI			
Unsupervised	0.59	0.61	0.60
UNET	0.73	0.75	0.74
ALBU	0.79	0.80	0.80
DM++	**0.81**	**0.83**	**0.82**

3.2.2 Bouton Detection

Boutons are swellings that occur along neuronal axon branches. In mouse brain imaging datasets, they appear as balls with significantly higher pixel values than the axon branches they lie on (see Figure 3.4 for an example image with many boutons). Thus, boutons are actually captured by the DM graph reconstructions outputs as vertices on the graph that have high persistence values.

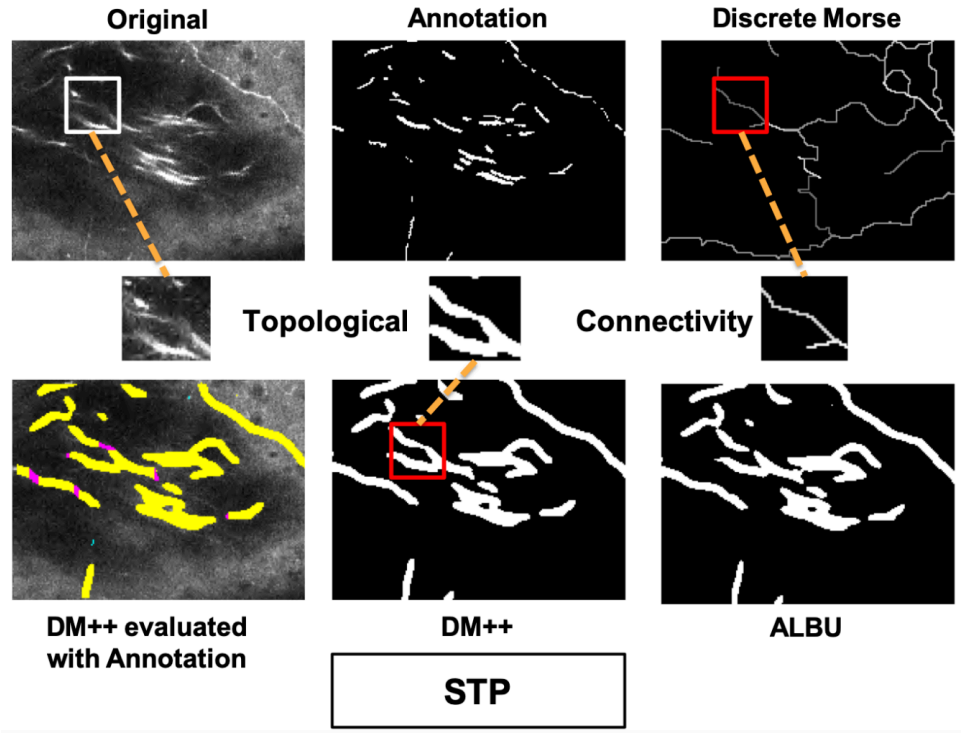


Figure 3.3. An example of where DM++ improves the connectivity of neuronal process segmentation of an STP mouse brain image. This diagram was originally published in [4].

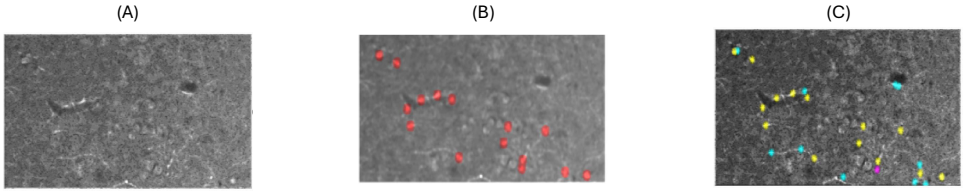


Figure 3.4. (A) A tile from a mouse brain image containing several neuronal branches with boutons. (B) The same image in (A) with detected boutons marked in red. (C) The same image in (A) with true positives detected in (B) marked in yellow, false positives in (B) marked in magenta, and false negatives missed in (B) marked in cyan. These pictures were originally published in [4].

We detect boutons by intersecting the highly persistent vertices of the DM graph reconstruction output with the final process detection output produced by DM++. An example from [4] is shown in Figure 3.4. We report modified precision, recall, and F-1 scores of .76, .31, and .44 respectively. The modified metrics used a radius of 2.5 microns, meaning a detection would be marked as a true positive if the closest true bouton was at most 2.5 microns away from the detection. This result was previously published in [4].

3.3 DM Graph Reconstruction for 3D Mouse Brain Images

3.3.1 Single Neuron Skeletonization

We develop a pipeline that produces a graph skeleton containing all neuronal process in a 3D mouse brain image. We run DM graph reconstruction on the 3D image, taking voxel values to be the density function. DIPHA [6] is used for the persistence computation inside of the DM graph reconstruction algorithm. The output produced is a list of vertices (voxels) and edges that make up the output graph that captures all neuronal process in the image. The pipeline is shown in Figure 3.5 (Pre-processing and Skeletonization steps).

The DM graph reconstruction output is great at skeletonizing all of the neuronal process in a 3D image (Figure 3.5). However, it is a poor single neuron skeletonization methodology on its own. While of the process of a single neuron may be skeletonized by DM graph reconstruction, it is impossible for DM graph reconstruction to not capture neuron branches in the image that

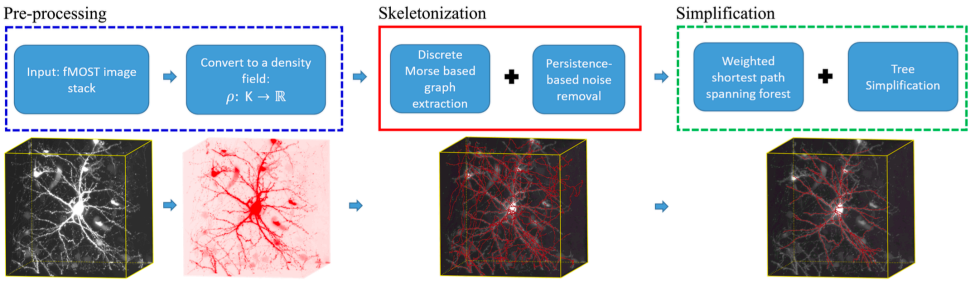
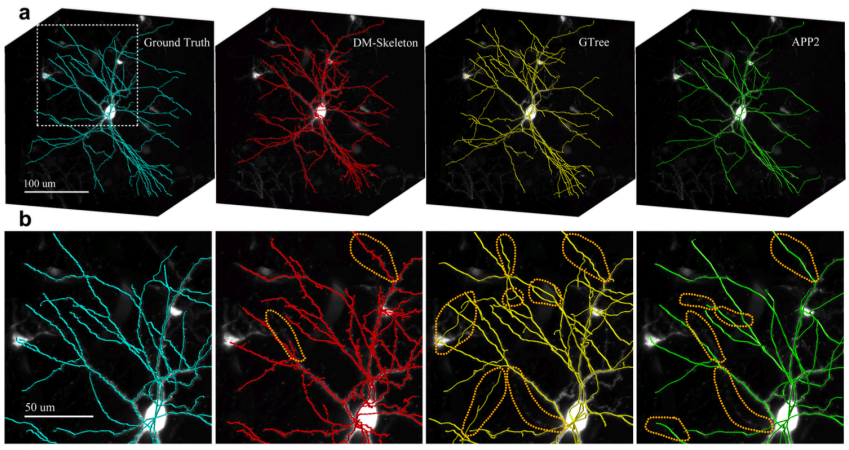


Figure 3.5. Workflow for single neuron skeletonization pipeline developed with our Cold Spring Harbor collaborators in [146]. The methodology and the figure were previously published in [146].

belong to multiple neurons. Thus, researchers at Cold Spring Harbor laboratory developed a single neuron skeletonization pipeline for fluorescent micro-optical sectioning tomograph (fMOST) 3D mouse brain images using the DM graph reconstruction output as the starting point [146]. Afterwards, the output graph is postprocessed by first computing a weighted shortest path spanning forest with trees rooted at each soma in the input image and the simplifying each tree. The full single neuron skeletonization pipeline is shown in Figure 3.5. Refer to [146] for more details on the simplification.

In [146], single neuron skeletonization is performed on 224 micron by 224 micron by 251 micron 3D fMOST mouse brain images centered at the somas of the neurons we wish to reconstruct. Results of one such single neuron reconstruction, as well as the ground truth annotation and results of APP2 [152] and GTree [65] are shown in Figure 3.6(A). Zoom-ins of the reconstructions (Figure 3.6(B)) reveal that APP2 misses several branches in the reconstruction and GTree has many false positive branches. The DM graph reconstruction based approach is observably closest to the ground truth for this particular neuron. Precision, recall, and F1 scores for all three methods on three neuron reconstructions are reported in Figure 3.6(C). For all three neurons, the DM graph reconstruction approach yields a significantly higher F1 score than APP2 and GTree. Figure 3.6 was previously published in [146].



	Method	Precision	Recall	F1
Neuron1	APP2	**0.926**	0.726	0.814
	GTree	0.846	0.906	0.875
	DM-Skeleton	0.900	**0.940**	**0.920**
Neuron2	APP2	0.654	0.568	0.608
	GTree	0.749	**0.750**	0.750
	DM-Skeleton	**0.865**	0.749	**0.803**
Neuron3	APP2	0.815	0.602	0.692
	GTree	0.621	0.729	0.670
	DM-Skeleton	**0.890**	**0.789**	**0.836**

Figure 3.6. (A) A zoom-in of a fMOST 3D mouse brain image centered around the soma of a neuron. From left to right, we see the ground truth annotation, the DM graph reconstruction approach, GTree, and APP2 skeletonization outputs. (B) Further zoom-ins on the ground truth and reconstruction outputs highlighting the differences in each output. (C) Precision, recall, and F1 scores for each of the three methodologies on three single neurons from 3D fMOST mouse brain imaging. This figure was previously published in [146].

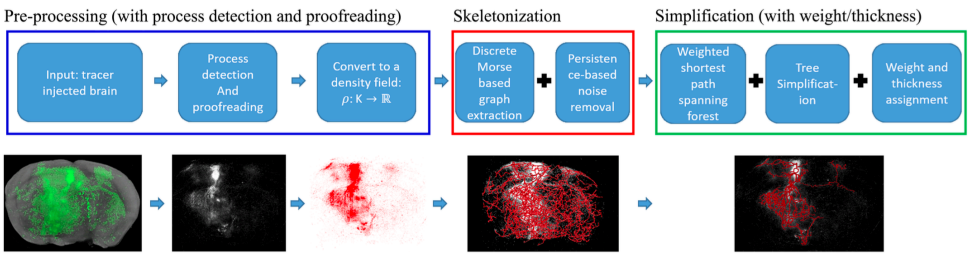


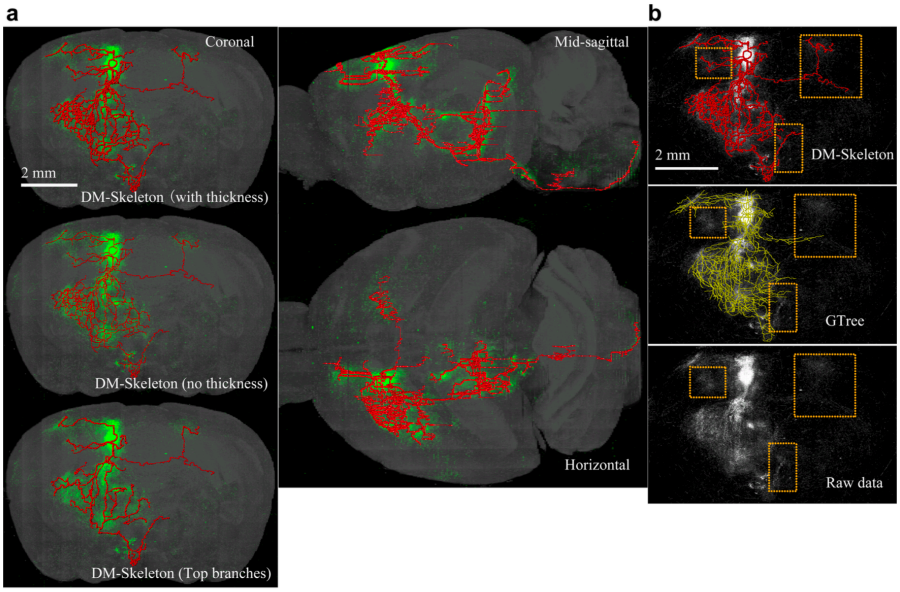
Figure 3.7. Workflow for tracer injection summaraization pipeline developed with our Cold Spring Harbor collaborators in [146]. The methodology and the figure were previously published in [146].

3.3.2 Tracer Injection Summarization

We develop a pipeline to produce a graph skeleton containing all potential neuronal trajectories from 3D multiple neuron tracer injection imaging datasets. We run the DM graph reconstruction algorithm on a (preprocessed (Figure 3.7 Pre-processing)) 3D mouse brain neuron tracer injection image, taking voxel values to be the density function. DIPHA [6] is used for the persistence computation inside of the DM graph reconstruction algorithm. The output is produced is a list of vertices (voxels) and edges that make up the output graph that captures all potential neuronal trajectories in the dataset. The pipeline is shown in 3.7 (Skeletonization).

The DM graph reconstruction captures all possible neuron trajectories, but contains many false positives. Thus, we collaborated with researchers at Cold Spring Harbor Laboratory to develop a tracer injection summarization pipeline that uses our DM graph reconstruction pipeline [146]. As previously stated, the raw tracer injection dataset is preprocessed with process detection [4] and proofreading. Our DM graph reconstruction pipeline is run on this preprecossed image. Finally, the output of DM graph reconstruction is postprocessed by first computing a weighted shortest path spanning forest, and then simplifying each tree. Postprocessing is concluded by assigning a thickness to each node along the graph. The full tracer injection summarization pipeline is shown in Figure 3.7. Refer to [146] for more details on the simplification.

In [146], tracjer injection summarization is performed on a full STP 3D mouse brain image. Multiple views (Figure 3.8(A)) show that the summarization pipeline developed in



Method	Top 5 non-injection regions in proofread brain (rank)					Other selected regions		Contralateral coverage	Ipsilateral coverage	Overall coverage
	CP(i)	MOs(i)	SSp(i)	SC(i)	APN(i)	MOp(c)	MOs(c)			
Proofread	13.44% (1)	9.56% (2)	4.83% (3)	3.76% (4)	3.05% (5)	1.34%	0.63%	-	-	-
DM	15.08% (1)	8.08% (2)	4.24% (3)	3.75% (5)	3.98% (4)	1.48%	0.26%	**57.99%**	**96.24%**	**93.34%**
GTree	19.68% (1)	10.21% (2)	6.25% (3)	1.51% (13)	2.21% (8)	0.21%	0.09%	15.44%	93.23%	87.33%
APP2	20.00% (1)	7.08% (2)	5.83% (3)	3.49% (7)	5.64% (4)	0	0	0	91.97%	85.00%

Figure 3.8. (A) Several views of the DM graph reconstruction based summarization on an 3D STP mouse brain image. (B) Fixed view of the raw data, DM graph reconstruction based summarization pipeline output, and GTree output. (C) . This figure was previously published in [146].

[146] faithfully captures all neuronal signal in the image. Comparing to other methodologies (Figure 3.8(B)), GTree struggles to signal through low intensity regions, a strength of the DM graph reconstruction approach. APP2 results for this image were excluded due to poor quality. Figure 3.8(C) contains coverage results for all three approaches, highlighting the DM graph reconstructions superiority in faithfully covering the full brain. Figure 3.8 and its results were previously published in [146].

3.4 Full Brain Skeletonization

Our final work in this Chapter concerns using DM graph reconstruction to construct skeletonizations for all neuronal process in high-resolution full brain imaging datasets. For both 2D and 3D pipelines, DIPHA [6] is used for the persistence computation in the DM graph

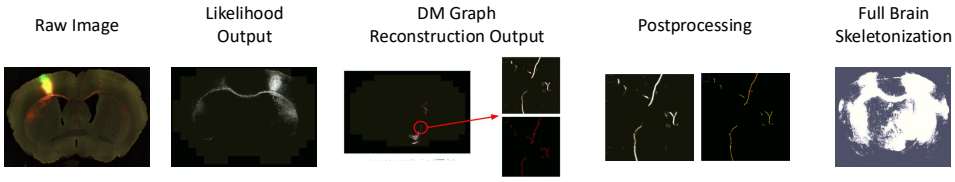


Figure 3.9. Workflow for our 2D mouse brain skeletonization pipeline. This pipeline is publicly available at www.github.com/lucasjmagee/Neuron-Fragment-DM-Skeletonization. The raw image is download from www.brainimagelibrary.org/. Likelihood and binary DM++ outputs were provided by our collaborators at Cold Spring Harbor Laboratory.

reconstruction algorithm.

3.4.1 2D Full Brain Skeletonization Pipeline

Given a full stack of high-resolution 2D mouse brain images, we build a pipeline to skeletonize all neuronal process in each image. Our pipeline is shown in Figure 3.9. Starting with raw images, DM++ [4] is run to obtain a likelihood masks. DM graph reconstruction is run on the likelihood images, and the output is intersection with the binary DM++ segmentation output to avoid capture of non-neuronal processes in the images. DM++ outputs were provided to us by our collaborators at Cold Spring Harbor Laboratory. We postprocess our graphs with a so-called ”haircut” process. The raw DM graph reconstruction outputs contain many false positives by nature of the algorithm. While the intersection with the binary DM++ process segmentation output removes most false positives, this leaves several short paths off of the true neuronal branches (Figure 3.9). We remove these branches by removing branches from our path that contain a single node of degree 1 and have a single change in direction. The skeletonizations for each image in the stack can be considered simultaneously to view all neuronal process in the full mouse brain.

3.4.2 3D Full Brain Skeletonization Pipeline

Given a high-resolution full brain 3D mouse brain imaging dataset, such as fMOST datasets, we build a pipeline to skeletonize all neuronal process in the image. Such datasets

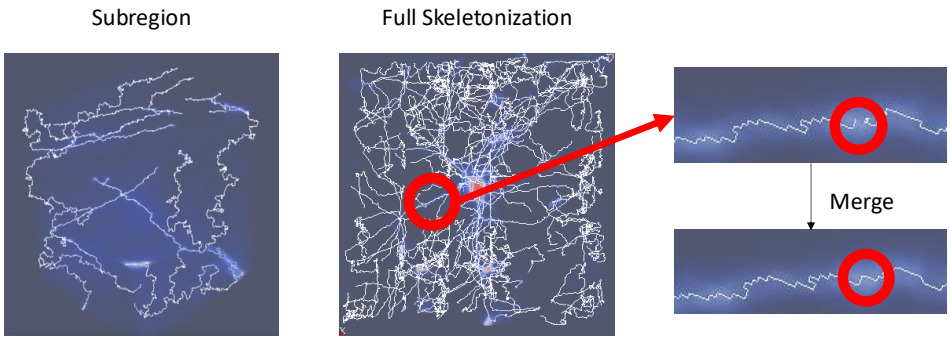


Figure 3.10. Workflow for our 3D mouse brain skeletonization pipeline. The input 3D mouse brain image is first divided into sub-images. Then DM graph reconstruction is performed on the each sub-images. Resulting graphs are tied together via the stitching process described in [148]. This pipeline is publicly available at www.github.com/lucasjmagee/3D-Discrete-Morse-Graph-Reconstruction.

are far too large for the DM graph reconstruction algorithm (and specifically, the persistence computation within the algorithm). Thus, we opt to run DM graph reconstruction on overlapping subsets of the input image, and stitch together the graph reconstructions via the methodology described in [148]. Briefly, we first divide the input image into overlapping sub-images. Next, DM graph reconstruction is performed on each sub-image. Then, a new graph is made starting with the union of all sub-image graphs. Unwanted disconnections of neuronal branches may occur in overlapping areas of our sub-images (see Figure 3.10). We build a new complex which builds triangulation in the overlap regions around components of our graph that lie within them and assign new vertices function values via diffusing of function values of vertices in the original sub-image graphs by Gaussian kernel. Finally, we run DM graph reconstruction on this new complex, which will correctly output a graph with connected branches that were previously disconnected in the overlapping regions (see Figure 3.10). Please refer to [148] for more detail on the stitching process.

3.5 Reprint

Chapter 3, in part, is a reprint of the material as it appears in Semantic segmentation of microscopic neuroanatomical data by combining topological priors with encoder-decoder deep networks in Nature Machine Intelligence. Banerjee, Samik; Magee, Lucas; Dingkang, Wang; Li, Xu; Huo, Bingxing; Jayakumar, Jaik-ishan; Matho, Katie; Lin, Adam; Ram, Keerthi; Sivaprakasam, Mohanasankar; Huang, Josh; Wang, Yusu; Mitra, Partha. 2020. The author of the dissertation was the second author of this article.

Chapter 3, in part, is a reprint of the material as it appears in Detection and skeletonization of single neurons and tracer injections using topological methods in bioRxiv. Wang, Dingkang; Magee, Lucas; Huo, Bingxing; Banerjee, Samik; Li, Xu; Jayakumar, Jaikishan; Lin, Meng Kuan; Ram, Keerthi; Wang, Suyi; Wang, Yusu; Mitra, Partha. 2020. The author of the dissertation was the second author of this article.

Chapter 4

Minimum Monotone Tree Decomposition of Density Graphs

We concluded the previous chapter (Chapter 3) with the development of full brain neuronal process skeletonization pipelines. Given that we can now extract all the neuronal process in a mouse brain imaging dataset into a single graph, it is natural to question how one might decompose that graph into the original neurons. In this chapter, we generalize the decomposition problem to constructing so-called minimum M-Tree Sets of density graphs. We will cover several hardness results for computing minimum M-Tree Sets and other variations of the problem. We will also provided several approximation algorithms, highlighted by a 3-approximation algorithm for computing the minimum SM-Tree Set for density cactus graph [97].

4.1 Introduction

A common problem in modern data analysis is taking large, complex datasets and extracting simpler objects that capture the true nature and underlying structure. In this chapter, we are interested in the case when the input data is the aggregation of a collection of trees. In fact, each tree also has attributes over nodes (e.g., the strength of certain signal) which decreases monotonically from its root – we call such a tree a monotone tree. Such trees come naturally in modeling a process that dissipates as it moves away from the root. One such example is in the

construction of neuronal cells: a single neuron has tree morphology, with the cell body (soma) serving as the root. In (tracer-injection based) imaging of brains, the signal often tails off as it moves away from the cell body and out of the injection region, naturally giving rise to a rooted monotone tree. Figure 4.1 (D) is centered around the soma of a single neuron within a full mouse brain imaging dataset with branches that get weaker as they get further from the soma.

Generally, we are interested in the following: given input data that is the aggregation of a collection of monotone trees, we aim to reconstruct the individual monotone trees. The specific version of the problem we consider in this chapter is where the input data is a graph $G = (V,E)$ with a density function $f : V \to \mathbb{R}^{\geq 0}$ defined on its vertices. Our goal is to decompose (G, f) into a collection of monotone trees $(T_1, f_1), \ldots, (T_k, f_k)$ whose union sums to the original (G, f) at each $v \in V$. See Section 4.2 for precise definitions. A primary motivation for considering graphs to be the input is because graphs are flexible and versatile, and recently, a range of methods have been proposed to extract the hidden graph structure from a wide variety of datasets; see e.g., [69, 82, 110, 1, 134, 58, 89, 27, 147, 40, 96]. In the aforementioned example of neurons, the discrete Morse-based algorithm of [40] has been applied successfully to extract a graph representing the summary of a collection of neurons [146, 4]. To extract the individual neurons from such a summary would be a significant achievement for the neuroscience community - which has developed many techniques to extract individual neuron skeletonizations from imaging datasets; see e.g., [65, 116, 29]. However, going from a graph to a collection of trees poses algorithmic challenges.

The monotone-tree decomposition problem has been studied in the work of [5], which develops a polynomial-time algorithm for computing the minimum cardinality set of monotone trees (M-Tree Set) of a density function defined on **a tree** (instead of a graph). However many applications for such a decomposition have graphs that may contain cycles, with the authors of [5] explicitly mentioning a need for algorithms that can handle such input domains.

4.1.1 Contributions

We consider density functions defined on graphs, which we refer to as *density graphs*. Our goal is to decompose an input density graph (G, f) into as few monotone trees as possible, which we call the *minimum M-Tree Decomposition problem*. See Section 4.2 for formal definitions and problem setup. Unfortunately, while the minimum M-Tree Decomposition problem can be solved efficiently in polynomial time via an elegant greedy approach when the density graph is itself a tree [5], we show in Section 4.3 that the problem for graphs in general is NP-Complete. In fact, no polynomial time constant factor approximation algorithm exists for this problem under reasonable assumptions (see Section 4.3). Additionally, we show NP-Completeness for several variations of the problem (Section 4.3). We therefore focus on developing approximation algorithms for this problem. In Section 4.4, we first provide two natural approximation algorithms but with additive error. For the case of multiplicative error, we provide a polynomial time 3-approximation algorithm for computing the so called minimum SM-Tree Set of a density cactus graph.

4.2 Preliminaries

4.2.1 Problem Definition

We will now introduce definitions and notions in order to formally define what we wish to compute. Given a graph $G(V,E)$, a *density function* defined on G is a function $f : V \to \mathbb{R}^{\geq 0}$. A *density graph* (G, f) is a graph G paired with a density function f defined on its vertices. A *monotone tree* is a density tree with a *root* $v \in V$ such that the path from the root to every node $u \in V$ is non-increasing in density values. See Figure 4.1 for explicit examples of density trees and monotone trees. While multiple nodes may have the global maximum value on the monotone tree, exactly one node is the root. For example, in Figure 4.1 (B), either node with the global maximum value may be its root, but only one of them is the root.

Given a density graph $(G(V,E), f)$, we wish to build a collection of monotone subtrees

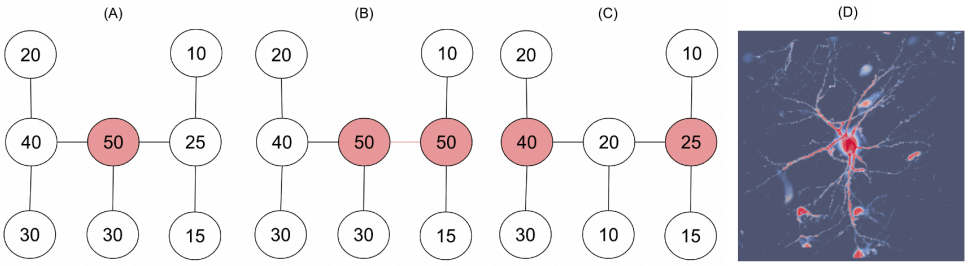


Figure 4.1. (A) - (C) Contain examples of density trees with relative maxima colored red. (A) shows a monotone tree. (B) shows a monotone tree with multiple nodes having the global maximum density value. (C) shows an example of a density tree that is not a monotone tree. (D) A zoom in an individual neuron within a full mouse brain imaging dataset. The dataset is an fMOST imaging dataset that was created as part of the Brain Initiative Cell Census Network and is publicly available for download.

$(T_1, f_1), (T_2, f_2), \ldots, (T_n, f_n)$ such that $T_i \subseteq G$ for all i and $\sum_{i=1}^{n} f_i(v) = f(v)$ for all $v \in V$. Note that if a node $v \in V$ is not in a tree T_i then we say that $f_i(v) = 0$ and vice versa. We will refer to such a decomposition as a *monotone tree (M-tree) decomposition* of the density graph, and refer to the set as an *M-Tree Set* throughout the remainder of the chapter. An M-Tree Set is a *minimum M-Tree Set* for a density graph if there does not exist an M-Tree Set of the density graph with smaller cardinality. An example of a density graph and a minimum M-Tree Set is shown in Figure 4.2. Note that a density graph may have many different minimum M-Tree Sets. We abbreviate the cardinality of a minimum M-Tree Set for a density graph (G, f) as $|\mathsf{minMset}((G, f))|$.

There are different types of M-Tree Sets that may be relevant for different applications. A *complete M-Tree (CM-Tree) Set* is an M-Tree Set with the additional restriction that every edge in the density graph G must be in at least one tree in the set. A *strong M-Tree (SM-Tree) Set* is an M-Tree Set such that the intersection between any two trees in the set must be either empty or contractible. We similarly abbreviate the cardinality of a minimum SM-Tree Set of (G, f) as $|\mathsf{minSMset}((G, f))|$. A *full M-Tree (FM-Tree) Set* is an M-Tree Set such that for each element $(T_i(V_i, E_i), f_i)$, $f_i(v) = f(v)$ for the root node $v \in V_i$ of (T_i, f_i). The (minimum) M-Tree Set in Figure 4.2 is also a (minimum) CM-Tree Set but is neither a SM-Tree Set nor a FM-Tree Set.

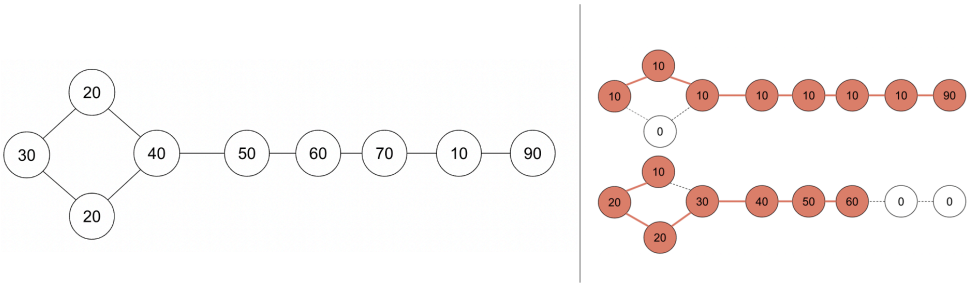


Figure 4.2. A density graph (left) together with a minimum M-Tree Set (right). Note that a minimum M-Tree Set is not necessarily unique for a density graph.

4.2.2 Greedy Algorithm for Density Trees [5]

We will now briefly describe the algorithm for computing minimum M-Tree Sets for density trees developed in [5], as some of the ideas will be useful in our work. Please refer to [5] for more details. The approach of [5] relies on a so-called *monotone sweeping operation* to build individual elements of a minimum M-Tree Sets of density trees. Algorithm 2 explicitly defines a generalized version of this operation that we will need in a later proof.

Algorithm 2: monotone-sweep($(T(V,E),f), v \in V, \alpha$)

Input: A density tree $(T(V,E),f)$, a starting node $v \in V$, and a staring value α such that $0 < \alpha \leq f(v)$

Output: A monotone subtree $(T',h_{f,v,\alpha})$ and a remainder $(T,R_{v,\alpha}f)$

(Step 1) Initialize output density subtree T' to only contain the input vertex v, with corresponding density function $h_{f,v,\alpha}(v) = \alpha$

(Step 2) Perform DFS starting from v. For each edge $(u \to w)$ traversed:

$$h_{f,v,\alpha}(w) = \begin{cases} h_{f,v,\alpha}(u) & f(w) \geq f(u) \\ max(0, h_{f,v,\alpha}(u) - (f(u) - f(w))) & otherwise \end{cases}$$

Return monotone tree $(T',h_{f,v,\alpha})$ and remainder density tree $(T,R_{v,\alpha}f)$.

The operation takes a density tree $(T(V,E),f)$, a node $v \in V$, and a starting function value α such that $0 < \alpha \leq f(v)$ as input. A monotone subtree $(T',h_{f,v,\alpha})$ and the remainder density tree $(T,R_{v,\alpha}f)$ where $R_{v,\alpha}f(u) = f(u) - h_{f,v,\alpha}(u)$ for all $u \in V$ is returned.

Algorithm 3, which outputs a minimum M-Tree Set of density trees, performs the monotone sweeping operation iteratively from certain nodes, called the *mode-forced* nodes of the

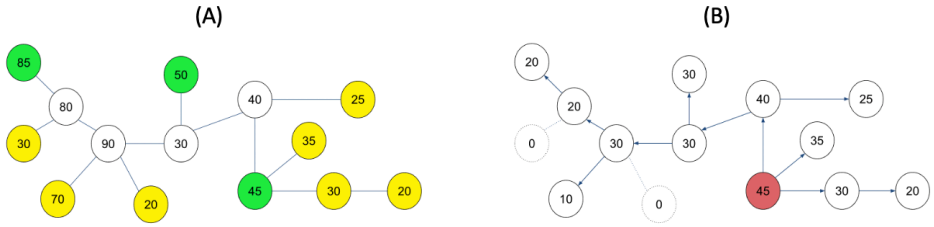


Figure 4.3. (A) A density graph with mode-forced nodes colored green and insignificant vertices colored yellow. (B) A single element built by the monotone sweep operation from a mode forced node as performed in Algorithm 3.

density tree. To compute these mode-forced nodes, one iteratively remove leaves from the tree if their parent has greater or equal density. Such leaves are referred to as *insignificant vertices*. Once it is no longer possible to remove any additional nodes, the leaves of the remaining graph are the mode-forced nodes of the original density graph.

Algorithm 3: tree-algo($(T(V,E),f)$)

Input: A density tree $(T(V,E),f)$
Output: A minimum M-Tree set of $(T(V,E),f)$
(Step 1) Find a mode forced vertex $v \in V$
(Step 2) Perform monotone-sweep($(T(V,E),f),v,f(v)$) to build a single element of a minimum M-Tree Set.
(Step 3) Repeat Steps 1 and 2 on remainder $(T,R_{v,f(v)}f)$ until no density remains.

An example of a single iteration of the tree algorithm is shown in Figure 4.3. The running time complexity of Algorithm 3 is $O(n*|\text{minMset}((T,f))|)$ where n is the number of nodes in T. We note that all M-Tree Sets of a density tree are also SM-Tree Sets, so Algorithm 3 also outputs a minimum SM-Tree Set.

4.2.3 Additional Property of Monotone Sweeping Operation

Unfortunately, neither Algorithm 2 nor Algorithm 3 can be directly used to compute minimum M-Tree Sets of density graphs with cycles. Nevertheless, we can show Claim 4.2.1 which will later be of use in developing approximation algorithms in Section 4.4.

Claim 4.2.1. *Given a density tree $(T(V,E),f)$, let $v \in V$. Let $a,b \in \mathbb{R}^+$ such that, without loss of generality, $0 < a < b \leq f(v)$. Let $(T,R_{v,a}f)$ be the remainder of* monotone-sweep$((T,f),v,a)$. *We can define a similar remainder $(T,R_{v,b}f)$. Then we have* $|\mathsf{minMset}((T,R_{v,b}f))| \leq |\mathsf{minMset}((T,R_{v,a}f))|$.

Proof. We will prove the claim by contradiction. Assume that $|\mathsf{minMset}((T,R_{v,b}f))| > |\mathsf{minMset}((T,R_{v,a}f))|$.

In particular, we will construct two new density trees, (T_a,f_a) and (T_b,f_b), as follows: T_a is equal to our starting tree with the addition of two nodes v_a and v_∞, with two additional edges connecting to v_a to both v_∞ and v. Set $f_a(v_a) = a$ and $f_a(v_\infty) = \infty$. Similarly define T_b and f_b.

Now imagine we run Algorithm 3 on (T_a,f_a). v_∞ is a mode-forced node, and thus we can perform the first iteraiton in Algorithm 3 on v_∞. Sweeping from v_∞ will leave remainder with a minimum M-Tree Set of size $|\mathsf{minMset}((T_a,f_a))| - 1$. The remainder is exactly the same as $(T,R_{v,a}f)$ at all nodes $v \in V$, and is zero at our newly added nodes. Hence, $|\mathsf{minMset}((T_a,f_a))| = |\mathsf{minMset}((T,R_{v,a}f))| + 1$. Similarly, by performing Algorithm 3 on (T_b,f_b), we have $|\mathsf{minMset}((T_b,f_b))| = |\mathsf{minMset}(T,R_{v,b}f)| + 1$

Now if our initial assumption is true, then by the above argument we have that

$$|\text{minMSet}((T_b,f_b))| > |\text{minMSet}((T_a,f_a))|. \tag{4.1}$$

However, we could construct an M-tree set of (T_b,f_b) as follows: First construct one monotone tree rooted at v_∞ that leaves no remainder at both v_∞ and v_b, then perform the monotone sweep operation starting at v with starting value a to build the rest of the component. Note that the remainder after removing this tree is in fact $(T_a,R_{v,a}f)$, which we can then decompose using the minimum M-tree set of $(T_a,R_{v,a}f)$. In other words, we can find a M-tree set for (T_b,f_b) with $|\mathsf{minMset}(T,R_{v,a}f)| + 1 = |\mathsf{minMset}(T_a,f_a)|$. This however contradicts with Eqn (4.1) (and the correctness of Algorithm 3). Hence our assumption cannot hold, and we must have that $|\mathsf{minMset}(T,R_{v,b}f)| \leq |\mathsf{minMset}(T,R_{v,a}f)|$. This proves the claim.

We note that while this proof is for M-Tree Sets specifically, the proof for SM-Tree Sets follows identical arguments. □

4.3 Hardness Results

Given that there exists a polynomial time algorithm for computing minimum M-Tree Sets of density trees, it is natural to ask whether or not such an algorithm exists for density graphs. We prove Theorem 4.3.1, stating that the problem is NP-Complete.

Theorem 4.3.1. *Given a density graph $(G(V,E),f)$ and a parameter k, determining whether or not there exists an M-Tree set of size $\leq k$ is NP-Complete.*

Proof. It is easy to see that this problem is in NP, so we will now show it is also in NP-Hard. First we consider a variation of the Set Cover problem where the intersection between any two sets is at most 1. We refer to this problem as Set Cover Intersect 1 (SC-1). SC-1 is a generalization of the NP-Complete problem of covering points in a plane with as few lines as possible [100], and approximation bounds of SC-1 are well studied in [86]. Given an instance of SC-1 (m sets $S_1, S_2, \ldots, S_m$ covering a universe of n elements $e_1, e_2, \ldots, e_n$, and a number k), we reduce to an instance of the M-Tree Set decision problem as follows:

- Create a bipartite graph $G(V = A \cup B, E)$ equipped with a density function $f : V \to \mathbb{R}^{\geq 0}$ based on the input (SC-1) instance.
- In particular, for each set S_i, add a node a_{S_i} to A and set $f(a_{S_i}) = |S_i|$.
- For each element e_j, add a node b_{e_j} to B and set $f(b_{e_j}) = 1$
- For each set S_i, add edge between a_{S_i} and b_{e_j} for each element $e_j \in S_i$.

An example of this reduction is illustrated in Figure 4.4.

First Direction: If there is a Set Cover of size $\leq k$, then there is an M-Tree Set of density graph (G,f) whose cardinality is $\leq k$.

Let S_{cover} be a set cover of size $n \leq k$. For each $S_i \in S_{cover}$, we will construct a monotone tree (T_i, f_i) rooted at a_{S_i}. In particular, $f_i(a_{S_i}) = f(a_{S_i})$. Then, for each element $e_j \in S_i$, T_i will include b_{e_j} and the edge (a_{S_i}, b_{e_j}), with $f_i(b_{e_j}) = 1$. Note that if e_j is an element in multiple sets in S_{cover}, simply pick one $S_i \in S_{cover}$ such that $e_j \in S_i$ to be the representative set of e_j. Finally, for each set $S_l \notin S_{cover}$, for each element $e_j \in S_l$, add the node a_{S_l} and the edge (b_{e_j}, a_{S_l}) to T_i with $f_i(a_{S_l}) = 1$, where (T_i, f_i) is the monotone tree rooted at the node a_{S_i} where $S_i \in S_{cover}$ is the representative set containing e_j.

Firstly, each element in the M-Tree Set is connected by construction. The only nodes in an element (T_i, f_i) are the root node a_{S_i}, where $S_i \in S_{cover}$, nodes of the form b_{e_j}, where $e_j \in S_i$, and nodes of the form a_{S_l}, where $S_l \notin S_{cover}$ and there exists e_j in both S_i and S_l. Edges of the form (a_{S_i}, b_{e_j}) are part of the domain by construction and are included in T_i. Similarly, edges of the form (a_{S_l}, b_{e_j}) are also part of the domain by construction and are included in T_i. For each edge $(a_{S_l}, b_{e_j}) \in T_i$, there must also exist an edge (a_{S_i}, b_{e_j}). Thus all nodes in T_i are connected to a_{S_i} - and in particular at most 2 edges away.

Secondly, each element in the M-Tree Set is a tree. Consider element (T_i, f_i). By construction, if a cycle were to exist in T_i it would have to be of the form $a_{S_i}, b_{e_p}, a_{S_l}, b_{e_q}, a_{S_i}$, where both e_p and e_q are in both S_i and S_l. However, such a cycle would imply that two sets have at least two elements in their intersection, which is not possible given we reduced from SC-1.

Next, each element in the M-Tree Set is a monotone tree. $f_i(v) = 1$ for all $v \in T_i$ that are not the root a_{S_i} of (T_i, f_i) and $f_i(a_{S_i}) \geq 1$.

Finally, $f(v) = \sum_{a=1}^{n} f_i(v)$ for all $v \in G$. Each node a_{S_i} such that $S_i \in S_{cover}$ is part of one monotone tree (T_i, f_i) and $f_i(a_{S_i}) = f(a_{S_i})$. Each node $b_{e_j} \in B$ is also part of only one monotone tree (T_i, f_i) and $f_i(e_j) = 1 = f(e_j)$. Finally, for a set $S_l \notin S_{cover}$, a_{S_l} is included in $m = |S_l|$ monotone trees. For each such monotone tree (T_i, f_i), $f_i(a_{S_l}) = 1$, thus $\sum_{a=1}^{n} f_i(a_{S_l}) = m = f(a_{S_l})$.

Thus, we have proven that there exists a M-Tree Set of (G, f) of size $\leq k$.

Second Direction: If there is an M-Tree Set of density graph (G, f) of size $\leq k$, then there is

a Set Cover of size $\leq k$.

Let $\{(T_i, f_i)\}$ be an M-Tree set of density graph (G, f) of size k. Each monotone tree (T_i, f_i) in the set has a root node m_i. If multiple vertices in T_i have the maximum value of f_i (as seen in Figure 4.1(B)) simply set one of them to be m_i. Each edge in T_i has implicit direction oriented away from m_i. First we prove Lemma 4.3.2.

Lemma 4.3.2. *Let $b_{e_j} \in B$. Either b_{e_j} is the root of a monotone tree in the M-Tree Set or at least one of its neighbors is the root of a monotone tree in the M-Tree Set.*

Proof. Assume b_{e_j} is not a root of any monotone tree. Consider a monotone tree (T_i, f_i) of the M-Tree Set containing b_{e_j}. This means that $f_i(b_{e_j}) > 0$. Consider node a_{S_l} that is the parent of b_{e_j} in (T_i, f_i). Assume a_{S_l} is not the root node of (T_i, f_i). Because a_{S_l} is not the root of the component, it must have a parent b_{e_d}. Consider the remaining density graph $(G, g = f - f_i)$. By definition of monotone tree, $0 < f_i(b_{e_j}) \leq f_i(a_{S_l}) \leq f_i(b_{e_d})$. By construction, we also know $f(a_{S_l}) = \sum_{e_j \in S_l} f(b_{e_j})$. Therefore, $g(a_{S_l}) > \sum_{e_j \in S_l} g(b_{e_j})$. Because a_{S_l} has more density than the sum of all of its neighbors in (G, g), it is impossible for a_{S_l} to not be the root of at least one monotone tree in any M-Tree Set of (G, g). Thus if b_{e_j} is not the root of any monotone tree in the M-Tree Set, a_{S_l} must be the root of a monotone tree in the M-Tree Set. □

We now construct a set cover from the M-Tree Set with the help of Lemma 4.3.2. Initialize S_{cover} to be an empty set. For each $a_{S_i} \in A$ that is a root of a monotone tree in the M-Tree Set, add S_i to S_{cover}. Next for each $b_{e_j} \in B$ that is the root of a monotone tree in the M-Tree Set, if there is not already a set $S_i \in S_{cover}$ such that $e_j \in S_i$, choose a set S_l such that $e_j \in S_l$ to add to the Set Cover. Every element must now be covered by S_{cover}. A node e_j that is not the root in any monotone tree in the M-Tree Set must have a neighbor a_{S_l} that is a root in some monotone tree by Lemma 4.3.2. The corresponding set S_l was added to S_{cover} - thus e_j is covered. A node e_m such that b_{e_m} is the root of a monotone tree in the M-Tree Set must also be covered by S_{cover} - as a set was added explicitly to cover e_m if it was not already covered. We've added at most one set to the cover for every monotone tree in the M-Tree Set, therefore $|S_{cover}| \leq k$.

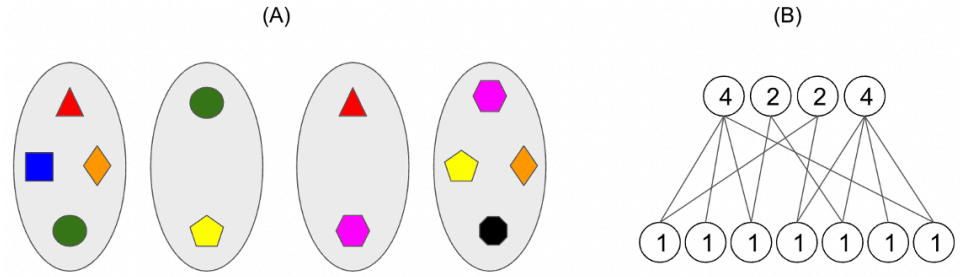


Figure 4.4. (A) SC-1 instance with 4 sets and seven elements. (B) M-Tree decision problem instance created by following reduction outlined in proof of Theorem 4.3.1. The top row consists of nodes in $A \subset V$ in the bipartite graph, which are nodes representing sets, while the bottom row consists of nodes in $B \subset V$ in the bipartite graph, which are nodes representing elements.

Combining both directions, we prove that, given a SC-1 instance, we can construct a density graph (G, f) such that there exists a set cover of size $\leq k$ if and only if the density graph has a M-tree Set of size $\leq k$. This proves the problem is NP-Hard, and thus the problem is NP-Complete. □

4.3.1 Approximation Hardness

From the proof of Theorem 4.3.1, it is easy to see that given an instance of SC-1, the size of its optimal set cover is equivalent to the cardinality of the minimum M-Tree Set of the density graph constructed in the reduction. Hence the hardness of approximation results for SC-1 translate to the minimum M-Tree Set problem too. We therefore obtain the following result, stated in Corollary 4.3.3 which easily follows from a similar result for SC-1. The SC-1 result from [86] is stated in Appendix A.1. We note that while Corollary 4.3.3 states the bound in terms of $n =$ of number of relative maxima, a similar bound can be obtained where $n =$ number of vertices.

Corollary 4.3.3. *There exists a constant $c > 0$ such that approximating the minimum M-Tree Decomposition problem within a factor of $c\frac{log(n)}{log(log(n))}$, where n is the number of relative maxima on the given density graph, in deterministic polynomial time is possible only if $NP \subset DTIME(2^{n^{1-\varepsilon}})$ where ε is any positive constant less than $\frac{1}{2}$.*

Proof. Under the same assumptions mentioned above, there exists a $c > 0$ such that SC-1 cannot be approximated within a factor of $c\frac{log(n)}{log(log(n))}$, where n is the number of elements in the universe [86]. We note that for a given SC-1 instance, performing the reduction to the M-Tree Set decision problem seen in the proof of Theorem 4.3.1 results in a density graph with at most $\frac{n(n-1)}{2} + n$ relative maxima - the upper bound on the number of sets in the SC-1 instance. Thus, the number of relative maxima on the density graph is $O(n^2)$.

For sufficiently large n, we have the following:

$c\frac{log(n^2)}{log(log(n^2))} = 2c\frac{log(n)}{log(2log(n))} = 2c\frac{log(n)}{log(log(n))+1} < 2c\frac{log(n)}{log(log(n))}$

Thus there exists a $c > 0$ such that minimum M-Tree Decomposition problem cannot be approximated within a factor of $c\frac{log(n^2)}{log(log(n^2))}$ under the same assumptions mentioned previously. Because the number of relative maxima on the density graph is $O(n^2)$, we can substitute the number of relative maxima for n^2 to establish our final bound.

□

4.3.2 Variations of minimum M-Tree Sets are also NP-Complete

In addition to proving that computing minimum M-Tree Sets of density graphs is NP-Complete, we have also proven Theorem 4.3.4 in Appendix A.2. The theorem states that computing the minimum CM-Tree Sets, minimum SM-Tree Sets, and minimum FM-Tree Sets of density graphs is also NP-Complete.

Theorem 4.3.4. *Given a density graph $(G(V,E), f)$ and a parameter k, determining whether or not there exists a CM-Tree Set, SM-Tree Set, or FM-Tree Set of size $\leq$ k are all NP-Complete.*

It should be noted that Corollary 4.3.3 can be extended to CM-Tree Sets and FM-Tree Sets. In contrast, SC-1 is not used in the NP-Complete proof for SM-Tree Sets. Thus, Corollary 4.3.3 does not apply to minimum SM-Tree Set and there is hope we can develop tighter bounded approximation algorithms for this problem than for the other variations.

4.4 Algorithms

4.4.1 Additive Error Approximation Algorithms

Now that we have shown that computing minimum M-Tree Sets of density graphs, as well as several additional variations, is NP-Complete, we focus on developing approximation algorithms. We define two algorithms with different additive error terms. Firstly, we note that a naive upper bound for a given density graph is the number of relative maxima on the graph. We include Algorithm 9 in Appendix A.3 to establish this naive upper bound.

Shifting focus to nontrivial approaches, Algorithm 4 computes the minimum M-Tree Set of a density graph restricted to a spanning tree $T \subseteq G$. We prove that $|\mathsf{minMset}((T,f))| \leq |\mathsf{minMset}((G,f))| + 2g$, where g the *genus* of G. For a connected graph, $G(V,E)$, its genus is equal to $|E| - |V| + 1$, which is the number of independent cycles on the graph. This approximation error bound for Algorithm 4 is stated in Theorem 4.4.1.

Algorithm 4: additive-error-algo((G(V, E), f))

Input: A density graph $(G(V,E),f)$ such that $\beta_1 G = g$
Output: An (S)M-Tree set of G, f
(Step 1) Compute g edges that if removed leave a spanning tree T of G
(Step 2) Compute minimum (S)M-Tree set of density tree (T,f) via Algorithm 3

Theorem 4.4.1. *Let $(G(V,E),f)$ be a density graph with $\beta_1 G = g$. Let k^* be the size of a minimum (S)M-Tree Set of (G,f). Algorithm 4 outputs an (S)M-Tree Set of size at most $k^* + 2g$.*

Proof. We need to prove Lemma 4.4.2 to provide an upper bound on $|\mathsf{minMset}(T,f)|$ for any spanning tree $T \subseteq G$. Algorithm 3 will then output an M-Tree Set of size at most equal to the upper bound, thus completing our proof. The proof is identical for SM-Tree Sets.

Lemma 4.4.2. *Let $(G(V,E),f)$ be a density graph with $\beta_1 G = g$ and $|\mathsf{minMset}(G,f)| = k^*$. For any spanning tree $T \subseteq G$, $|\mathsf{minMset}(T,f)| \leq k^* + 2g$*

Let $M = \{(T_i, f_i)\}$ be a minimum M-Tree set of (G, f). Let E_{cut} be the set of g edges that if removed from G leave spanning tree T. Firstly, we note that $|\mathsf{minMset}(G, f)| \leq |\mathsf{minMset}(T, f)|$, as any M-Tree Set of (T, f) is also an M-Tree Set of (G, f).

We will construct an M-Tree Set of (T, f) from a minimum M-Tree Set of G. For each monotone tree $(T_i, f_i) \in M$, consider an edge $e_j = (u, v) \in E_{cut}$ that is in T_i. There is implicit direction to e_j with respect to the root of (T_i, f_i), meaning either (1) $(u \to v)$ or (2) $(v \to u)$. If (1) is the case, we can cut the branch rooted at v off of (T_i, f_i) to create two non-intersecting monotone trees. See Figure 4.5 for an example. We perform a similar operation if (2) is the case, but instead cut the branch rooted at u. Perform this cut for each edge in E_{cut} to divide (T_i, f_i) into, at most, $|E_{cut}| + 1$ non-intersecting monotone trees. After dividing each tree into at most $|E_{cut}| + 1$ non-intersecting monotone trees, we make 2 key observations - (1) we still have a M-Tree Set of (G, f) and (2) no edge in E_{cut} is in any monotone tree in the M-Tree Set. Thus the M-Tree Set is also an M-Tree Set of (T, f).

We can shrink the size of this M-Tree Set by summing the components that share the same root. In particular, consider an edge $e_j = (u, v) \in E_{cut}$. We have created as many as k^* additional monotone trees rooted at u and as many as k^* additional monotone trees rooted at v. Sum the monotone trees rooted at u to create a single monotone tree rooted at u. The sum would clearly still be a monotone tree because all monotone trees are subtrees of tree T, so no cycle or non-non-increasing path from u will be created. We can similarly do the same for v, and for all edges in E_{cut}. This we have a new M-Tree Set of (T, f), with (at most) an additional monotone tree rooted at each node of each edge in E_{cut} when compared to the original M-Tree Set of G. Thus $|\mathsf{minMset}(T, f)|$ is bounded above by $k^* + 2g$. □

4.4.2 Approximation Algorithm for Minimum SM-Tree Sets of Density Cactus Graphs.

A cactus graph is a graph such that no edge is part of more than one simple cycle [67]. See Figure 4.6 (A) for an example. Many problems that are NP-hard on graphs belong to P when

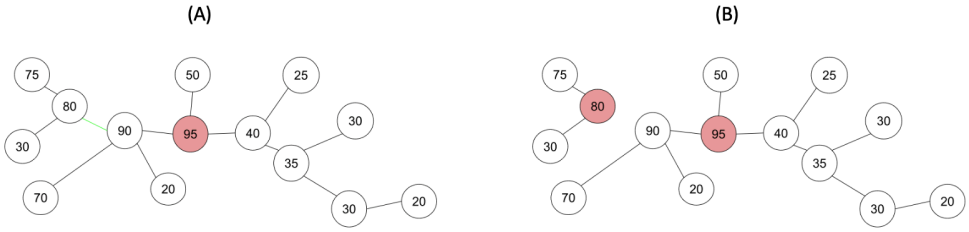


Figure 4.5. (A) shows a single monotone tree with its root colored red and an edge colored green. Cutting the green edge leaves us with two non-intersecting monotone trees shown in (B).

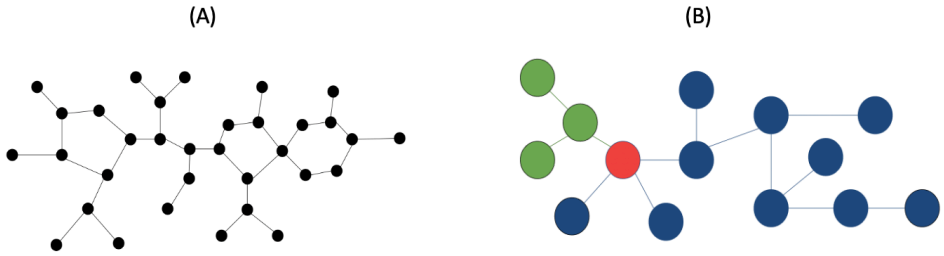


Figure 4.6. (A) An example of a cactus graph - which is a graph such that no edge is part of more than a single simple cycle. It is essentially a tree of cycles graph - which is a graph such that no vertex is part of more than one simple cycle - with the exception that two simple cycles may share a single vertex. Tree of cycles graphs are cactus graphs but cactus graphs (such as this one) are not necessarily tree of cycles graphs. (B) An example of an input for Algorithm 5. The density tree is broken into two subtrees (green and blue) that have a single node as intersection (red). Monotone sweeping is performed iteratively at mode-forced nodes only in one of the subtrees. Once the only remaining mode-forced nodes lie on the other tree, the output tuple containing the number of monotone sweeps performed and the remaining density at the intersection node is returned.

restricted to cacti - such as vertex cover and independent set [68]. While we do not yet know whether or not computing a minimum M-Tree Set (or any variations) of density cactus graphs is NP-hard, we have developed a 3-approximation algorithm for computing the minimum SM-Tree Set of a density cactus graph.

We first prove Theorem 4.4.3, which states that for any density cactus graph (G, f), there exists a spanning tree $T \subseteq G$ such that $|\mathsf{minSMset}(T, f)|$ is at most 3 times $|\mathsf{minSMset}(G, f)|$.

Theorem 4.4.3. *Let $(G(V,E), f)$ be a density cactus graph. There exists a spanning tree T of G such that $|\mathsf{minSMset}(T, f)| \leq 3|\mathsf{minSMset}(G, f)|$.*

Algorithm 5: split-tree-algo((T(V, E), f), T_1, T_2)

Input: A density tree $(T(V,E),f)$ and two subtrees T_1, T_2 of T that share a single node v as intersection

Output: A tuple (a,b) representing the number of monotone sweeps a from mode-forced nodes on T_1 to make all mode-forced nodes on T be part of T_2, and the remaining function value b at v after the monotone sweeps.

While there exists mode-forced node $u \in V$ off of T_2:
- monotone-sweep$((T,f),u,f(u))$
Set $a =$ number of monotone sweeps performed
Set $b =$ remaining density on v
Return (a,b)

Proof. Let $M = \{(T_i, f_i)\}$ be a minimum SM-Tree Set of (G,f), $k = |M|$, and $\beta_1 G = g$. Consider graph $G' = \bigcup_{i=1}^{k} T_i$. Let $\beta_1 G' = g'$. We note that $g' \leq g$. and that M is also a minimum SM-Tree Set of G'. We will use G' to help construct a spanning tree T of G with an SM-Tree Set with the desired cardinality. Note that if G' has no cycles then there obviously exists a spanning tree T of G such that $|\mathsf{minSMset}(T,f)| = |\mathsf{minSMset}(G,f)|$. Additionally, if $g' = 1$, then creating T by removing any edge from the simple cycle $|\mathsf{minSMset}(T,f)| \leq |\mathsf{minSMset}(G,f)| + 2$ (similar arguments to Lemma 4.4.2 and Theorem 4.4.1). Therefore assume $g' \geq 2$. Construct spanning tree T as follows:

- Add each edge in G that is not part of a simple cycle.
- For each simple cycle in G that is not in G', add all edges of cycle to T except for one missing in G' (it does not matter which if multiple such edges exist).
- For each simple cycle in G that is in G', add all edges of that cycle to T except for one (does not matter which).

Let k' be the $|\mathsf{minSMset}(T,f)|$. k' is bounded above by $k + 2g'$, because removing edges that aren't used in the any monotone tree in M from the domain will not change $|\mathsf{minSMset}(G',f)|$. Additionally, removing an edge from a simple cycle in g' will increase the $|\mathsf{minSMset}(G',f)|$ by at most 2 (again by Lemma 4.4.2).

k' is also bounded below by $2+g'$. For each cycle in G', the number of monotone trees in M that contain nodes in a simple cycle must be at least 3 - otherwise the set cannot be an SM-Tree Set. So consider a leaf cycle C_0 in G'. We know that there are at least 3 monotones trees in M that cover C_0 . For a cycle C_1 adjacent to C_0 in G' that there is a single path between the two cycles, and the monotone trees that cover C_0 cannot completely cover C_1, otherwise M would not be an SM-Tree Set. There must be at least one monotone tree with nodes on C_1 and no nodes on C_0. Continuing traversing the graph to all cycles and it is clear that for each cycle there must be an additional monotone tree added to the SM-Tree Set. Thus, we cannot have an SM-Tree Set of size less than $2+g'$.

From above, we have $\frac{k'}{k} \leq \frac{k+2g'}{k} \leq \frac{2+g'+2g'}{2+g'} \leq \frac{3g'+2}{g'+2} < 3$. □

With Theorem 4.4.3 proven, we aim to compute the optimal density spanning tree of a density cactus graph. To help compute such an optimal density spanning tree, we first define Algorithm 5. Given a density tree $(T(V,E),f)$ divided into two subtrees T_1 and T_2 that share a single node $v \in V$ as intersection, Algorithm 5 performs monotone sweeping operations on the mode-forced nodes of T_1 until all mode-forced nodes of (T,f) are on T_2. An example of a valid input is seen in Figure 4.6 (B). The output is a tuple (a,b), where a is the number of monotone sweeps performed and b is the remaining density on v after performing the monotone sweeps. The tuple will essentially capture how helpful monotone sweeping from T_1 is for building a minimum (S)M-Tree Set on T_2. Algorithm 5 can be used to help compute the desired density spanning tree. In particular, it is used in Algorithm 6, to cut the optimal edge from each cycle, one cycle at a time. We prove Theorem 4.4.4 which states that Algorithm 6 returns an SM-Tree Set at most 3 times larger than a minimum SM-Tree Set of a density cactus graph. The running time of Algorithm 6 is $O(n^3)$ where n is the number of nodes in the input cactus graph. Algorithm 2 ($O(n^2)$) is performed once for each edge ($O(n)$) that is part of a simple cycle.

Theorem 4.4.4. *Algorithm 6 outputs an SM-Tree Set that is at most 3 times the size of a minimum SM-Tree Set of the input density cactus graph.*

Algorithm 6: opt-spanning-tree-algo((G(V, E), f))

Input: Density cactus graph $(G(V,E),f)$
Output: SM-Tree Set of (G,f)
If G is a tree
- Compute optimal (S)M-Tree Set of (G,f) using Algorithm 3.

Else If G has only a single cycle
- compute optimal sized (S)M-Tree Set of each density spanning tree of G and return smallest cardinality (S)M-Tree Set.

Else (G has multiple simple cycles)
- Compute a leaf cycle $C = c_1,\ldots,c_m$ connected to rest of cycles at c_i
- Let $G_C =$ the simple cycle C with all branches off of each node in the cycle - not including the branches off of c_i that do not lead to other cycles in the graph. Let $G_{\bar{C}} = T - G_C + c_i$.
- Fix a spanning tree $T_{G_{\bar{C}}}$ of $G_{\bar{C}}$.
- For each spanning tree T_i of G_C computing split-tree-algo$((G_C \cup G_{\bar{C}}, f), T_i, T_{G_{\bar{C}}})$
- Set $G = G(V, E - e^*)$ such that e^* is edge removed from C that results in spanning tree with smallest output of split-tree-algo.
- Iterate until basecase (single cycle graph) is achieved

Proof. Clearly, the algorithm outputs a minimum SM-Tree Set when the input domain is a tree. When G contains a single cycle, by Lemma 4.4.2 the outputted SM-Tree Set will have at most 2 more monotone trees than a minimum SM-Tree Set of G. Therefore, we only need to prove Theorem 4.4.4 holds when G contains multiple simple cycles. Because G is a cactus a leaf cycle $C = c_1,\ldots,c_m$ exists. Let G_C be the graph of all nodes in the cycle and branches off of those nodes, excluding branches off of c_i that do not lead to other cycles. Let $G_{\bar{C}}$ be the graph of G excluding C and all branches off of C, except for the node c_i itself. G_C has m spanning trees, $T_1,\ldots,T_m$ corresponding to the m edges of C. Fix a spanning tree $T_{G_{\bar{C}}}$ of $G_{\bar{C}}$. We next introduce Lemma 4.4.5.

Lemma 4.4.5. *Let $T^* =$ spanning tree of G_C such that the output of* split-tree-algo*$(T^* \cup T_{G_{\bar{C}}}, T^*, T_{G_{\bar{C}}})$ is minimized.* $|\mathsf{minSMset}((T^* \cup T_{G_{\bar{C}}}, f))| \leq$ $|\mathsf{minSMset}((T_k \cup T_{G_{\bar{C}}}, f))|$ *for any spanning tree $T_k \subseteq G_C$.*

Proof. Lemma 4.4.5 is proven by proving Claim 4.4.6 and Claim 4.4.7.

Claim 4.4.6. *If* split-tree-algo$(T_j \cup T_{G_{\bar{C}}}, T_j, T_{G_{\bar{C}}})[0] <$ split-tree-algo$(T_k \cup T_{G_{\bar{C}}}, T_k, T_{G_{\bar{C}}})[0]$ *then* $|\mathsf{minSMset}((T_j \cup G_{\bar{C}}, f))| \leq |\mathsf{minSMset}((T_k \cup G_{\bar{C}}, f))|$.

Proof. Let T_j, T_k be spanning trees of G_C such that $a_j < a_k$, where $a_j =$ split-tree-algo$(T_j \cup T_{G_{\bar{C}}}, T_j, T_{G_{\bar{C}}})[0]$ and $a_k =$ split-tree-algo$(T_k \cup T_{G_{\bar{C}}}, T_k, T_{G_{\bar{C}}})[0]$. Let $s^* = |\mathsf{minSMset}((T_{G_{\bar{C}}}, f))|$.

Algorithm 5 performs Algorithm 3 sweeping from mode-forced nodes on T_j, but stops once mode-forced nodes only remain on $T_{G_{\bar{C}}}$. Thus it is still constructing minimum SM-Tree Sets but stopping short of completion. The first element of the output of Algorithm 5 indicates the number of iterations required to have only mode-forced nodes on $T_{G_{\bar{C}}}$. $|\mathsf{minSMset}((T_j \cup T_{G_{\bar{C}}}, f))| \leq a_j + s^*$. Similarly, $|\mathsf{minSMset}((T_k \cup T_{G_{\bar{C}}}, f))| \geq a_k + s^* - 1$. These bounds prove the claim.

□

Claim 4.4.7. *If* split-tree-algo$(T_j \cup T_{G_{\bar{C}}}, T_j, T_{G_{\bar{C}}})[0] =$ split-tree-algo$(T_k \cup T_{G_{\bar{C}}}, T_k, T_{G_{\bar{C}}})[0]$ *and* split-tree-algo$(T_j \cup T_{G_{\bar{C}}}, T_j, T_{G_{\bar{C}}})[1] <$ split-tree-algo$(T_k \cup T_{G_{\bar{C}}}, T_k, T_{G_{\bar{C}}})[1]$ *then* $|\mathsf{minMset}((T_j \cup G_{\bar{C}}, f))| \leq |\mathsf{minMset}((T_k \cup G_{\bar{C}}, f))|$.

Proof. Let T_j, T_k be spanning trees of G_C such that $a_j = a_k$ and $b_j < b_k$ where $(a_j, b_j) =$ split-tree-algo$(T_j \cup T_{G_{\bar{C}}}, T_j, T_{G_{\bar{C}}})$ and $(a_k, b_k) =$ split-tree-algo$(T_k \cup T_{G_{\bar{C}}}, T_k, T_{G_{\bar{C}}})$.

$a_j = a_k$ indicates that both T_j and T_k require the same number of iteration of monotone sweeps to leave mode-forced nodes on $T_{G_{\bar{C}}}$. However, $b_j < b_k$ means that T_j is more helpful than T_k for reducing the minimum SM-Tree Set size on the remainder in the same number of monotone sweeps by Claim 4.2.1. This proves the claim. □

This completes the proof of Lemma 4.4.5.

□

It follows from Lemma 4.4.5 that Algorithm 6 outputs a minimum SM-Tree Set on the density spanning tree that has the smallest sized minimum SM-Tree Set of all density spanning trees. Combining this with Theorem 4.4.3 finishes the proof.

□

4.5 Conclusion

Decomposing density graphs into a minimum M-Tree Set becomes NP-Complete when the input graph is not restricted to trees. We proved that computing the minimum M-Tree, CM-Tree, SM-Tree, and FM-Tree Set of density graphs is NP-Complete. We provided additive error approximations algorithms for the minimum M-Tree Set problem, as well as developed a 3-approximation algorithm for minimum SM-Tree Sets for density cactus graphs. Future work will be to close the gap between the bounds of approximation we have established with the error bounds of the algorithms we have developed.

4.6 Reprint

Chapter 4, in full, is a reprint of the material as it appears in Minimum Monotone Tree Decomposition of Density Functions Defined on Graphs in Lecture Notes in Computer Science. Magee, Lucas; Wang, Yusu. 2024. The dissertation author was the primary investigator and author of this paper.

Chapter 5

High Dimensional PCD Algorithm

In previous chapters (Chapter 3 and Chapter 4), we focused first on extracting graph structure from 2D and 3D mouse brain imaging datasets. Now, in this chapter, we will shift focus to the second part of this dissertation - extracting graph structure from high-dimensional PCDs. We will first change our perspective of DM graph reconstruction from a density-based perspective to a filtration-based perspective, allowing us to define a generalized DM graph reconstruction algorithm. After providing theoretical justification for the new generalized algorithm, we design a DM graph reconstruction algorithm for high dimensional PCDs and include several empirical results to demonstrate both the effectiveness and efficiency of the methodology.

5.1 Introduction

Modern complex data, or the space where data is sampled from, often has a simpler underlying structure. A key step in modern data analysis is to model and extract such hidden structures. A particularly interesting type of non-linear structure is a (geometric) graph skeleton, which can be thought of as a 1-D *singular manifold*, consisting of pieces of 1-manifolds (curves) glued together. Graph structures are common in practice, such as river networks and dark matter filament structures in cosmology. Graphs can also be natural models for the evolution of trends behind data (e.g, the evolution of topics in twitter data).

While there has been beautiful work on manifold learning [125, 135, 9, 46], recovering

singular manifolds is more challenging [10]. Nevertheless, recovering a hidden graph skeleton (singular 1-manifolds) from data has attracted much attention; e.g, in [69, 82, 110]. In general, one of the main challenges involved is to identify graph nodes and connections among them. Local information is often used to make inference or decisions, making it hard to handle noise, non-uniform sampling and gaps in data. To this end, topological methods become useful, as they offer ways to capture the global structure behind data and thus can be robust in detecting junction nodes and their global connectivity. Indeed, there are several algorithms that extract a graph skeleton behind point cloud datasets (PCDs) based on topological ideas; e.g. [1, 58, 27, 89]. Unfortunately, while such approaches work well when the input points are sampled within a tubular neighborhood of the hidden graph (called tubular or Hausdorff noise), they do not effectively handle more general noise, such as outliers and background noise. The locally-defined principal curve approach [110] can handle noisy data with non-tubular noise via a ridge-finding strategy using a constraint mean-shift-like procedure. However, the procedure only moves points closer to a graph skeleton without outputting an actual graph.

Recently, there has been a line of work using a persistence-guided discrete Morse theory based approach to reconstruct a graph (or even a 2D) skeleton from *density field* [38, 63, 123, 134, 147]. In particular, assume that the input is a density field defined on a discretized domain. Such methods use the discrete Morse (DM) theory to compute the so-called stable 1-manifolds to capture the mountain ridges of the density field and returns these mountain ridges as the extracted graph skeleton; see Figure 1.2 for a 2D example. Persistent homology is used to simplify the resulting stable 1-manifolds. The algorithm based on this idea has been significantly simplified in [40] together with theoretical analysis. The resulting method (which we will refer to as DM-graph) can recover a hidden graph from noisy and non-homogeneous density fields, and has already been applied to several applications in 2D/3D [4, 41, 43]. These graphs have also been used as input for Graph Neural Networks (GNNs) to generate effective predictive models for rock data [20]. However, this method currently assumes that one has *a discretization of the ambient space where data is embedded in*, which becomes prohibitively expensive for

high dimensional data, and also cannot be directly applied to metric data that is not embedded.

5.1.1 Contribution

We consider the general setting where the input is just a set of points P embedded in a metric space, say the Euclidean space $\mathbb{R}^d$, or with pairwise distances (or correlations) given. The previous DM-graph does not work in this setting, and as we will explain later, the straightforward extension is not effective for high-dimensional PCDs. In this chapter, we extend the idea behind the discrete-Morse based approach beyond density field, and combine it with the so-called sparsified weighted Rips filtration of [16] to develop an effective and efficient algorithm to infer graph skeletons of high-dimensional PCDs.

More specifically, in Section 5.2, we view the DM-graph reconstruction method from a filtration perspective instead of a density perspective, and thus generalize the DM-graph algorithm to work with an arbitrary filtration (which intuitively is a sequence of growing spaces spanned by our input points in our setting). We then prove (Theorem 5.2.5) that the output of the generalized method contains a so-called *lex-optimal persistent cycle basis* of the given filtration, thereby showing that the output captures meaningful information w.r.t. the filtration. This result is of independent interest.

We next show how this simple change of view can help us reconstruct the graph skeleton of a set of points P more efficiently and effectively. In particular, the filtration perspective now allows us to combine the DM-based graph reconstruction algorithm with a sparsified weighted Rips filtration scheme proposed by [16], which both improves the quality of the reconstruction and significantly reduces the time complexity. This new graph reconstruction algorithm for PCDs, called DM-PCD, is our second main contribution and presented in Section 5.4.

Finally, we show experimental results on a range of datasets, and compare with previous methods to demonstrate the effectiveness of our new DM-PCD algorithm. More results are shown in the Appendix.

5.2 Generalized Algorithm and Optimality

Now suppose instead of a triangulation of a d-D domain Ω, we have an arbitrary simplicial complex K – our algorithm only needs its 2-skeleton $K = (V, E, T)$. Suppose further that there is a simplex-wise function $\widehat{\rho} : K \to \mathbb{R}$. Let $\Pi_{\widehat{\rho}} := \langle \sigma_1, \ldots, \sigma_N \rangle$ be an ordered sequence of simplicies of K that is *consistent with* $\widehat{\rho}$ (see the end of Section 2.1), and let $\mathscr{F}_{\widehat{\rho}}$ be the simplex-wise filtration of K induced by this order $\Pi_{\widehat{\rho}}$. (We will describe in Section 5.4 how to set up this filtration for graph skeletonization from PCDs.) We now generalize algorithm DM-graph() to the following extDM-graph(), where essentially, only (Step 1) differs by taking an arbitrary simplex-wise filtration $\mathscr{F}_{\widehat{\rho}}$, which we state in Algorithm 7 for clarity.

Algorithm 7: extDM-graph($K, \mathscr{F}_{\widehat{\rho}}, \delta$)

Input: Arbitrary simplex-wise filtration $\mathscr{F}_{\widehat{\rho}}$ of a simplicial complex $K = (V, E, T)$, threshold δ
Output: A reconstructed graph G_δ
(Step 1) Compute persistence pairings w.r.t. $\mathscr{F}_{\widehat{\rho}}$
(Step 2) + (Step 3): same as in alg. DM-graph

It is easy to verify that the original DM-graph(K, ρ, δ) algorithm is a special case of the above algorithm, where we set $\mathscr{F}_{\widehat{\rho}}$ in extDM-graph($K, \mathscr{F}_{\widehat{\rho}}, \delta$) to be the lower-star filtration induced by the vertexwise function $\rho' = -\rho : V(K) \to \mathbb{R}$; specifically, for any simplex $\sigma \in K$, set $\widehat{\rho}(\sigma) := \max_{v \in \sigma} -\rho(v)$. The difference between our extDM-graph() algorithm and the original algorithm is rather minor. However, we will see that this change of perspective (from density-function based view to arbitrary filtration-based view) significantly broadens the applicability of this algorithm. In particular, in Section 5.4 we will show how this generalized algorithm can be combined with weighted Rips sparsification strategy to reconstruct a hidden graph skeleton of high-dimensional points data. But first, in what follows, we provide some characterization of the graph skeleton output by extDM-graph. Specifically, we show that the output of extDM-graph() contains the so-called *lex-optimal* cycle basis of K w.r.t. important 1D homological features in $\mathrm{dgm}_1 \mathscr{F}_{\widehat{\rho}}$. To make this statement more precise, we first introduce some notations, following

[34, 42, 150]. Intuitively, a 1-cycle is a collection of edges forming one or multiple closed loops; and a d-cycle is a d-D analog of it.

Definition 5.2.1 (Persistent cycles [42]). *Let $\mathscr{F}$ be a simplexwise filtration of K induced by the ordered sequence of simplices $\sigma_1,\ldots,\sigma_N$, and $\mathrm{dgm}_q\mathscr{F}$ its resulting q-th persistence diagram. Given a point $\mathsf{p}=[b,d]\in\mathrm{dgm}_q\mathscr{F}$, a q-cycle γ is* a persistent q-cycle w.r.t. p *if (i) if $d\neq\infty$, γ is a cycle in K_b containing σ_b, and γ is not a boundary in K_{d-1} but becomes a boundary in K_d; and (ii) otherwise if $d=\infty$, then γ is a cycle in K_b containing σ_b.*

Given a subset $D=\{\mathsf{p}_1,\ldots,\mathsf{p}_r\}\subseteq\mathrm{dgm}_q\mathscr{F}$ with $r=|D|$, we say that a set of cycles $\{\gamma_1,\ldots,\gamma_r\}$ form a persistent cycle-basis *for D if γ_i is a persistence cycle w.r.t. p_i for all $i\in[1,r]$.*

Roughly speaking, a persistent cycle γ w.r.t. a persistence point $\mathsf{p}=[b,d]$ is created at b and killed at d, and can be thought of a representative of the homological feature captured by point $\mathsf{p}\in\mathrm{dgm}\mathscr{F}$. A persistent cycle basis w.r.t. $D\subset\mathrm{dgm}\mathscr{F}$ corresponds to representative cycles captured by points in D. More specifically, given a cycle γ, let $[\gamma]_{K_i}$ denote the homology class of γ in complex K_i. The following result from [42] intuitively says that a persistence cycle-basis for $\mathrm{dgm}_q\mathscr{F}$ essentially generates the interval decomposition of persistence module $\mathbb{P}\mathscr{F}$.

Claim 5.2.2 ([42]). *Let $\{\gamma_1,\ldots,\gamma_g\}$ be a persistence cycle-basis for $\mathrm{dgm}_q\mathscr{F}=\{\mathsf{p}_1,\ldots,\mathsf{p}_g\}$ and $g=|\mathrm{dgm}_q\mathscr{F}|$. Then $\mathbb{P}\mathscr{F}=\bigoplus_{\mathsf{p}_\ell\in\mathrm{dgm}_q\mathscr{F}}\mathbb{I}^{\mathsf{p}_\ell}$, where the interval module $\mathbb{I}^{\mathsf{p}_\ell}=\{I_i\xrightarrow{\nu_{i,j}}I_j\}_{i\leq j}$ is generated by γ_ℓ in the sense that $I_i=[\gamma_\ell]_{K_i}$.*

Lexicographic optimal cycles are introduced in [33, 34]. We will extend them to the persistence version. Given a simplex-wise filtration $\mathscr{F}$ of K induced by an ordering of simplices $\sigma_1,\ldots,\sigma_N$, we set $\mathsf{ind}(\sigma)$ as the order it appears in $\mathscr{F}$; i.e, $\mathsf{ind}(\sigma_i)=i$.

Definition 5.2.3 (Lexicographic order [34]). *Given two q-cycles $C_1,C_2\in\mathsf{C}_q(K)$, we say that $C_1\preceq C_2$ if either (i) $C_1+C_2=0$ or (ii) otherwise, the simplex $\sigma_{max}:=\mathrm{argmax}_{\sigma\in C_1+C_2}\mathsf{ind}(\sigma)$ is from C_2. If (ii) holds, we say that $C_1\prec C_2$, i.e., C_1 is* smaller than C_2 in lexicographic order. *Intuitively, $C_1\preceq C_2$ if simplices in C_1 comes "earlier" than C_2 in the filtration order.*

Definition 5.2.4 (Lex-optimal persistent cycles)**.** *Given a persistence point* $\mathsf{p} = [b,d] \in \mathrm{dgm}_q\mathscr{F}$*, a q-cycle* γ *is a* lexicographic-optimal (lex-opt for short) persistent cycle w.r.t. p *if (i)* γ *is a persistent cycle w.r.t.* p*; and (ii) among all persistence cycles w.r.t.* p*,* γ *has the smallest lexicographic order. We say that* $\Gamma = \{\gamma_1, \ldots, \gamma_r\}$ *forms a* lex-optimal persistent cycle basis *for a multiset* $D = \{\mathsf{p}_1, \ldots, \mathsf{p}_r\} \subseteq \mathrm{dgm}_q\mathscr{F}$ *if* γ_i *is a lex-optimal persistence cycle w.r.t* p_i *for all* $i \in [1,r]$.

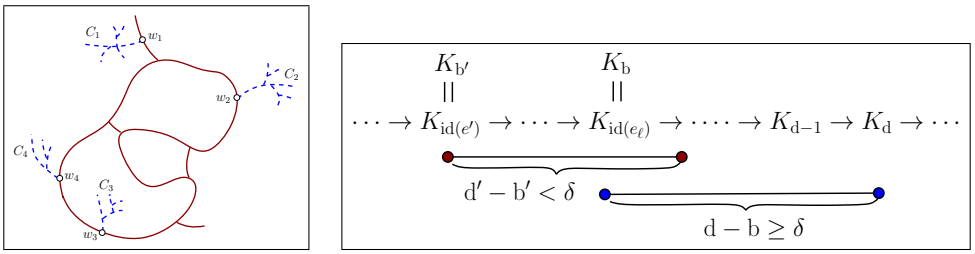
Given a q-th persistence diagram $\mathrm{dgm}_q\mathscr{F}$ and a threshold δ, let $\mathrm{dgm}_q\mathscr{F}(\delta) \subseteq \mathrm{dgm}_q\mathscr{F}$ denote the subset of points in $\mathrm{dgm}_q\mathscr{F}$ whose persistence is larger than δ (intuitively, these corresond to important features). Our first main result is the following theorem.

Theorem 5.2.5. *(i)* G_δ *as constructed w.r.t. a simplex-wise filtration* $\mathscr{F}_\rho$ *contains a lex-optimal persistence cycle basis for* $\mathrm{dgm}_1\mathscr{F}_\rho(\delta)$*, and (ii) the first Betti number of* G_δ *equals* $|\mathrm{dgm}_1\mathscr{F}_\rho(\delta)|$.

The above theorem suggests that the output graph G_δ by our algorithm extDM-graph() contains the "best" loops whose homology classes have large persistence and whose edges come *as early as possible in the filtration*. In particular, imagine that important edges or more faithful edges come early in the filtration, then the output graph contains those loops with large persistence ($> \delta$) and formed by more faithful edges whenever possible. In the graph reconstruction from PCDs application in the next section, intuitively, if edges from high-density region come into the filtration first, then the resulting output graph will use such edges whenever possible. See Figure 5.3 (A) to (D).

5.3 Proof of Theorem 3.5

We assume that K is connected. If it is not, then we will perform the following arguments to each connected component of K. Now recall that $\mathscr{T}_\delta := \{e \in E \mid e \text{ is negative and } \mathrm{pers}(e) \leq \delta\}$ consists of all negative edges with persistence at most δ (from (Step 2) of Algorithm 7). It is shown in [40] that $\mathscr{T}_\delta$ consists of a set of trees. Set



(a) (b)

Figure 5.1. (a) The solid red curve is G_δ, while dashed trees are components in $\widehat{G}_\delta \setminus G_\delta$. The closure of each component C_i connects to G_δ at one point w_i, and thus its closure can deformation retracts to $w_i \in G_\delta$. (b) As $\mathrm{pers}(e') = \mathsf{d}' - \mathsf{b}' \leq \delta$, and $\mathrm{pers}(\mathrm{e}_\ell) = \mathsf{d} - \mathsf{b} > \delta$, and $\mathsf{b}'(= \mathsf{ind}(e')) \leq \mathsf{b}$, it then follows that the persistent cycle $C' = \pi(u,v) + e'$ must become a boundary in the simplicial complex $K_{\mathsf{d}-1}$, which in turn leads to that $[\gamma'] = [\gamma^*]$ in $K_{\mathsf{d}-1}$.

$$E_\delta^+ := \{e \in E \mid e \text{ is positive and } \mathrm{pers}(e) > \delta\}, \text{ and}$$

$$E_\delta^- := \{e \in E \mid e \text{ is negative and } \mathrm{pers}(e) > \delta\}.$$

Set $\widehat{G}_\delta = \mathscr{T}_\delta \bigcup G_\delta$, where G_δ is the output of algorithm extDM-graph. Furthermore, by construction, G_δ consists of edges in $E_\delta^- \cup E_\delta^+$ together with a set of tree paths in $\mathscr{T}$ (recall Eqn (2.3) in Algorithm 1, which is the same as the construction for algorithm extDM-graph). It follows that

$$\widehat{G}_\delta = \mathscr{T}_\delta \bigcup G_\delta = \mathscr{T}_\delta \bigcup E_\delta^- \bigcup E_\delta^+. \tag{5.1}$$

We prove Theorem 3.5 in two steps, laid out in the following two lemmas.

Lemma 5.3.1. *Statements (i) and (ii) in Theorem 3.5 holds for $\widehat{G}_\delta$. That is: (i') $\widehat{G}_\delta$ constructed w.r.t. a simplex-wise filtration $\mathscr{F}_\rho$ contains a lex-optimal persistence cycle basis for $\mathrm{dgm}_1 \mathscr{F}_\rho(\delta)$, and (ii') the first Betti number of $\widehat{G}_\delta$ equals $|\mathrm{dgm}_1 \mathscr{F}_\rho(\delta)|$.*

Lemma 5.3.2. *$\widehat{G}_\delta$ deformation contracts to G_δ.*

Our theorem then follows from these two lemmas. Specifically, we will use the graph $\widehat{G}_\delta$

as a proxy: Lemma 5.3.1 states that the desired results hold for $\widehat{G}_\delta$. Lemma 5.3.2 then relates $\widehat{G}_\delta$ to G_δ. In particular, as both $\widehat{G}_\delta$ and G_δ are graphs, this lemma implies that any simple cycle in $\widehat{G}_\delta$ must be present in G_δ as well. Theorem 3.5 then follows. What remains is to prove these two lemmas, which we present in the two subsections that follow.

5.3.1 Proof of Lemma 5.3.1

Let $\mathrm{dgm}_1\mathscr{F}_\rho(\delta) = \{\mathsf{p}_1, \ldots, \mathsf{p}_g\}$. By the definition of positive and negative edges, we know:

- (C1). $\widehat{\mathscr{T}} = \mathscr{T}_\delta \cup E^-$ is a spanning tree of K.
- (C2). By the definition of positive edges, E^+_δ contains exactly those edges whose addition create the persistence points in $\mathrm{dgm}_1\mathscr{F}_\rho(\delta)$. In other words, $g = |E^+_\delta|$ and we can order edges in $E^+_\delta = \{\mathsf{e}_1, \ldots, \mathsf{e}_g\}$ so that for any $\ell \in [1, g]$, $\mathsf{p}_\ell = [\mathsf{ind}(\mathsf{e}_\ell), d_\ell]$: i.e, the birth-time of p_ℓ corresponds to the insertion of edge e_ℓ in the simplicial complex $K_{\mathsf{ind}(\mathsf{e}_\ell)}$.

Furthermore, the addition of each positive edge $\mathsf{e}_\ell \in E^+_\delta$ creates a cycle in the spanning tree $\widehat{\mathscr{T}}$ (as e_ℓ is not a tree edge), As $\widehat{G}_\delta = \widehat{\mathscr{T}} \cup E^+$, we thus have $\beta_1(\widehat{G}_\delta) := \mathit{rank}(\mathsf{H}_1(\widehat{G}_\delta))$ is the same as $g = |\mathrm{dgm}_1\mathscr{F}_\rho(\delta)|$. This proves part (ii') in Lemma 5.3.1 for the graph $\widehat{G}_\delta$.

We now prove part (i') of Lemma 5.3.1. Consider any $\mathsf{e}_\ell \in E^+_\delta$, and let γ^* denote a lex-opt persistent cycle of the corresponding persistence point $\mathsf{p}_\ell = [\mathsf{b}, \mathsf{d}]$ (where $\mathsf{b} = \mathsf{ind}(\mathsf{e}_\ell)$). By the definitions of persistent cycles and lexicographic order, γ^* necessarily contains e_ℓ, and all other edges in γ^* have an index smaller than $\mathsf{ind}(\mathsf{e}_\ell)$. We will next prove that γ^* is in $\widehat{G}_\delta$, that is, treating a cycle (under $\mathbb{Z}_2$ coefficients) as a set, $\gamma^* \subseteq \widehat{G}_\delta$.

In particular, take any edge $e' \in \gamma^*$ with $e' \neq \mathsf{e}_\ell$, we will show that $e' \in \widehat{G}_\delta$.

- If e' is negative, then this is trivially true as $e' \in \widehat{\mathscr{T}} \subseteq \widehat{G}_\delta$.
- If e' is positive but with persistence $\mathrm{pers}(e') > \delta$, then it is also true as $e' \in E^+_\delta \subseteq \widehat{G}_\delta$.

- So what remains is the case when $e' = (u, v)$ is positive but with $\text{pers}(e') \leq \delta$. However, we will show that this case cannot happen, which implies that $e' \in \widehat{G}_\delta$.

  Assume this case happens for edge e'. Then let $C_{e'}(= \pi(u,v) + e') \subset K_{\mathsf{ind}(e')}$ be a persistent cycle w.r.t. the persistence point $[\mathsf{b}' = \mathsf{ind}(e'), \mathsf{d}']$ generated by e'. First, as the path (1-chain) $\pi(u,v)$ is contained in $K_{\mathsf{ind}(e')}$, all edges in $\pi(u,v)$ have an index less than that of e'. This means that the cycle $\gamma' = \gamma^* - e' + \pi(u,v)$ is necessarily smaller than γ^* in lexicographic order. We now claim that γ' is also a persistent cycle w.r.t. the persistence point $\mathsf{p}_\ell = [\mathsf{b}, \mathsf{d}]$ corresponding to the positive edge e_ℓ.

  Indeed, as γ^* is a persistent cycle w.r.t. p_ℓ, we know that $\mathsf{ind}(e') < \mathsf{ind}(\mathsf{e}_\ell) = \mathsf{b}$. Recall that the persistence point corresponds to the positive edge e' is $[\mathsf{b}' = \mathsf{ind}(e'), \mathsf{d}']$. As $\text{pers}(\mathsf{e}_\ell) = \rho(\mathsf{d}) - \rho(\mathsf{b}) > \delta$ while $\text{pers}(e') = \rho(\mathsf{d}') - \rho(\mathsf{b}') \leq \delta$, it then follows that $\mathsf{d}' < \mathsf{d}$. (See Figure 5.1 (b) for illustrations of these notations.) Hence we know that it is necessary that the cycle $\pi(u,v) + e'$ becomes boundary in $K_{\mathsf{d}-1}$. In other words, in $K_{\mathsf{d}-1}$, the two cycles γ^* and γ' are homologous. It is then easy to verify that γ' must be a persistent cycle for p_ℓ as well.

  Since γ' is also a persistent cycle for p_ℓ and is lexicographically smaller than γ^*, this contradicts our assumption that γ^* is a lex-opt persistent cycle for p_ℓ. Hence no positive edge $e' \in \gamma^*$ with $\text{pers}(e') < \delta$ can be in γ^*.

By the above case analysis, any edge $e' \in \gamma^*$ must be in $\widehat{G}_\delta$. It then follows that $\gamma^* \subseteq \widehat{G}_\delta$. As this argument holds for any edge in E^+_δ, we thus have proven (i'). This finishes the proof of Lemma 5.3.1.

5.3.2 Proof of Lemma 5.3.2

First, by construction of $\widehat{G}_\delta$ (Eqn (5.1)), we have that $G_\delta \subseteq \widehat{G}_\delta$, and all edges in $\widehat{G}_\delta \setminus G_\delta$ must come from $\mathscr{T}_\delta$. Now recall $\widehat{\mathscr{T}} = \mathscr{T}_\delta \bigcup E^-$, which is a spanning tree of K. Given an arbitrary tree T and two nodes $u, v \in T$, let $\pi_T(u,v)$ denote the unique tree path from u to v in T. We have

the following simple claim.

Claim 5.3.3. *Given any rooted tree T with root $r(T)$ and two nodes $u, v \in T$, we have that $\pi_T(u,v) \subseteq \pi_T(u, r(T)) \cup \pi_T(v, r(T))$.*

Proof. If u and v have ancestor / descendent relation, say u is ancestor of v, then it is clear that $\pi_T(u,v) \subseteq \pi_T(v, r(T))$, and the claim then follows. Otherwise, let w be the common ancestor of u and v. It can again be verified that in this case, $\pi_T(u,w) \subseteq \pi_T(u, r(T))$, $\pi_T(v,w) \subseteq \pi_T(v, r(T))$, while $\pi_T(u,v) = \pi_T(u,w) \circ \pi_T(w,v)$. The claim thus follows. □

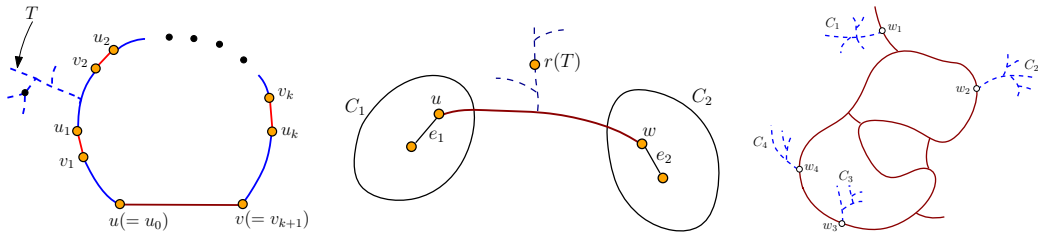
$\mathscr{T}_\delta$ is a spanning forest of vertex set V. Given any vertex $v \in V$, suppose it is in the tree $T \in \mathscr{T}_\delta$. We denote $path_{\mathscr{T}_\delta}(v) := \pi_T(v, r(T))$ to be the path from v to the root $r(T)$ of T. Recall that G_δ is constructed by, for any edge $e = (u,v) \in E_\delta^- \cup E_\delta^+$, adding $e \cup path_{\mathscr{T}_\delta}(u) \cup path_{\mathscr{T}_\delta}(v)$ into G_δ.

Lemma 5.3.4. *For each edge $\mathrm{e} = (u,v) \in E_\delta^+$, set $\gamma = e \cup \pi_{\mathscr{T}}(u,v)$. Then the cycle γ must be contained in G_δ.*

Proof. Consider the path $\pi = \pi_{\mathscr{T}}(u,v)$: it will be broken into $k \geq 0$ maximally connected pieces from $\mathscr{T}_\delta$, connected by edges in $E^- \cup E^+$. If $k = 0$, we are done, because this means that u, v are contained in the same tree T in $\mathscr{T}$, and it then follows from Claim 5.3.3 that

$$\pi = \pi_T(u,v) \subseteq path_{\mathscr{T}_\delta}(u) \cup path_{\mathscr{T}_\delta}(v) \subseteq G_\delta.$$

So assume that $k > 0$, and the edges connecting these pieces are $e_1 = (v_1, u_1), \ldots, e_k = (v_k, u_k)$ from u to v along π; see Figure 5.2 (a). Obviously, for each $i \in [1,k]$, $e_i \notin \mathscr{T}_\delta$ and $e_i \in E_\delta^- \cup E_\delta^+$. Set $u_0 = u$ and $v_{k+1} = v$. It then follows that for any $i \in [0,k]$, u_i is connected to v_{i+1} within some tree, say $T \in \mathscr{T}_\delta$. By Claim 5.3.3 that the portion of π from u_i to v_{i+1} must be



(a) (b) (c)

Figure 5.2. (a) The path $\pi = \pi(u,v)$ is broken into $k+1$ pieces, each of which (blue subcurves) is a maximal connected component in $\pi \cap \mathscr{T}_\delta$, while the connecting edges (red edges (v_i,u_i)'s) must come from $E_\delta^- \cup E_\delta^+$. (b) The solid red path is the tree path $\pi = \pi_T(u,w)$ connecting u and w in the tree T from the spanning forest $\mathscr{T}_\delta$. The root of T is $r(T)$. (c) The solid red curve is G_δ, while dashed trees are components in $\widehat{G}_\delta \setminus G_\delta$. The closure of each C_i connects to G_δ at one point w_i, and thus its closure can deformation retract to w_i.

contained in $path(u_i) \cup path(v_{i+1})$. Applying this for all $i \in [0,k]$, it follows that

$$\begin{aligned}\pi &\subseteq \Big(\bigcup_{i\in[0,k]} \big(path(u_i) \cup path(v_{i+1})\big)\Big) \bigcup \big(e_1 \cup e_2 \cdots \cup e_k\big) \\ &= \big(path(u) \cup path(v)\big) \bigcup \big(e_1 \cup path(v_1) \cup path(u_1)\big) \bigcup \cdots \bigcup \big(e_k \cup path(v_k) \cup path(u_k)\big).\end{aligned}$$

As all edges $e_1, \ldots e_k$ and e are all in $E^- \cup E^+$, it then follows that $\pi \subseteq G_\delta$ and thus $\gamma = e \cup \pi \subseteq G_\delta$. □

Lemma 5.3.5. *$\beta_0(G_\delta) = \beta_0(\widehat{G}_\delta)$, and $\beta_1(G_\delta) = \beta_1(\widehat{G}_\delta)$.*

Proof. That $\beta_1(G_\delta) = \beta_1(\widehat{G}_\delta)$ follows immediately from Lemma 5.3.4. We now prove that G_δ and $\widehat{G}_\delta$ also has the same number of connected components. Note that we have already assumed that K is connected, and thus $\widehat{G}_\delta$ is connected as it contains a spanning tree $\widehat{\mathscr{T}}$ of K. So what remains is to show that G_δ is connected.

Assume G_δ is not connected, and let C_1, C_2 be two components of G_δ. Recall that G_δ is constructed by the union of paths $\bigcup_{e=(u,v)\in E_\delta^- \cup E_\delta^+} \big(e \cup path_{\mathscr{T}_\delta}(u) \cup path_{\mathscr{T}_\delta}(v)\big)$. Let $e_1 \in C_1$ be an arbitrary edge from $C_1 \cap (E_\delta^- \cup E_\delta^+)$: Note that such an edge must exist, as otherwise C_1 will not be in G_δ. Similarly, let $e_2 \in C_2 \cap (E_\delta^- \cup E_\delta^+)$. Let u be an endpoint of e_1 while w be an edge point of e_2. We know that u and w are connected in $\mathscr{T}_\delta$ by path $\pi = \pi_{\mathscr{T}_\delta}(u,w)$. We now

claim that this path must be in G_δ; which contradicts with our assumption that C_1 and C_2 are two connected components of G_δ. Hence our assumption is wrong, and G_δ must be connected as well, which finishes the proof of the claim.

What remains is to show that the path $\pi = \pi_{\mathscr{T}_\delta}(u,w)$ as described above must be in G_δ. Let $T \in \mathscr{T}_\delta$ be the tree in $\mathscr{T}_\delta$ containing path π, and let r_T be its root. We now perform a case analysis based on the location of r_T w.r.t. u and w. (Case 1): u is an ancestor of w in T; (case 2): w is an ancestor of u in T; and (case 3): otherwise. See Figure 5.2 (b) for an illustration of (case 3). We first prove that $\pi \subseteq G_\delta$ for (case 3). In this case, we have that $path_{\mathscr{T}_\delta}(u) \cup path_{\mathscr{T}_\delta}(w)$ is a superset of π, that is, $\pi \subseteq path_{\mathscr{T}_\delta}(u) \cup path_{\mathscr{T}_\delta}(w)$. Furthermore, since both $e_1, e_2 \in E_\delta^- \cup E_\delta^+$, by construction of G_δ, $path_{\mathscr{T}_\delta}(u) \subseteq G_\delta$ and $path_{\mathscr{T}_\delta}(w) \subseteq G_\delta$. It then follows that $\pi \subseteq G_\delta$. Using a similar argument, one can show that $\pi \subseteq G_\delta$ for (case 1) and (case 2) as well.

Putting everything together, we have that G_δ is connected and thus $\beta_0(G) = \beta_0(\widehat{G}_\delta)$. This finishes the proof of the lemma. □

Now let $C_1, \ldots, C_s$ be the components of $\widehat{G}_\delta \setminus G_\delta$, and for each $i \in [1,s]$, let $\overline{C}_i$ be the closure of C_i. We claim that $\overline{C}_i \setminus C_i$ can contain only one vertex, say w_i. See Figure 5.2 (c). Indeed, as $\widehat{G}_\delta \setminus G_\delta \subseteq \mathscr{T}_\delta$, each $\overline{C}_i$ is simply connected (i.e, it is a subtree of some tree in $\mathscr{T}_\delta$). Suppose $\overline{C}_i \setminus C_i$ contains at least two vertices, say w and w'. As $\overline{C}_i$ is connected, there is a path $\pi_{\overline{C}_i}(w,w')$ connecting w to w' in $\overline{C}_i$. On the other hand, as G_δ is connected (Lemma 5.3.5), there is another path $\pi_{G_\delta}(w,w')$ connecting w and w'. This gives rise to a cycle $\gamma = \pi_{\overline{C}_i}(w,w') \cup \pi_{G_\delta}(w,w')$ in $\widehat{G}_\delta$, and this cycle is not in G_δ. This however contradicts to what we just proved that $\beta_1(G_\delta) = \beta_1(\widehat{G}_\delta)$. Hence this cannot happen.

Hence $\overline{C}_i$ can only connect to G_δ via one point w_i as illustrated in Figure 5.2 (c). It then follows that $\widehat{G}_\delta$ deformation retracts to G_δ by contracting each subtree $\overline{C}_i$ to the point w_i. This finishes the proof of Lemma 5.3.2.

5.4 PCD Algorithm via Sparse Weighted-Rips

Given a PCD $P \subset \mathbb{R}^d$, we now wish to compute a graph skeleton of P. Our algorithm can be easily extended to the case where these points P are not embedded but with only pairwise distances (or similarity) given.

A baseline approach.

A natural approach is to (i) build a simplicial complex K from P to "approximate" the space behind P, (ii) estimate a density function ρ at $P = V(K)$, and (iii) then perform algorithm DM-graph. A reasonable choice for K is the so-called Rips complex $\text{rips}^r(P) := \{(p_{i_0}, \ldots, p_{i_k}) \mid \|p_{i_j} - p_{i_{j'}}\| \leq r\}$: Intuitively, an edge $(p,q) \in \text{rips}^r(P)$ if the distance between points $p, q \in P$ is at most r. A triangle is in $\text{rips}^r(P)$ if all three edges are in, and similarly for higher-dimensional simplices. However, we only need 2-skeleton of $\text{rips}^r(P)$, which we still denote by $\text{rips}^r(P)$. The estimated density of a point is determined by summing the distances under a Gaussian kernel to each of its KNN for some k. We refer to this algorithm as baseline where we use the $\text{rips}^r(P)$ as choice of complex K, that is, we perform DM-graph($\text{rips}^r(P), \rho, \delta$).

Challenges with baseline**.**

This baseline approach faces several challenges. (C-1) It is usually hard to choose the right radius r and the topology of $\text{rips}^r(P)$ crucially decides the final output graph: see Figure 5.3, where if r is too small, the shape is not yet captured by $\text{rips}^r(P)$; for larger r, there can be spurious topological features (extra loops) in K which cannot be simplified by persistence (as these loops are generated by edges with infinity persistence). There is also the issue that even if one has found a radius r value such that $\text{rips}^r(P)$ can provide the correct topology, the geometry of the graph skeleton computed by this baseline algorithm may lose resolution (e.g, Figure 5.3 (H)).

(C-2) Points may be sampled at non-uniform resolution, hence there may not exist a single good r that can capture all features; see Figure 5.4. (C-3) Even for a moderate radius r,

the size of Rips complex becomes large, making persistence computation very costly. (C-4) The Rips complex can be a poor approximation of the hidden space when there is background noise; see Figure 5.4 (F) and (H), where even though the hidden space consists of 5 independent cycles (see Section 5.5), with much background noise, even a small radius r makes the Rips complex connect these noisy points and lose the hidden structure. Removing low-density points can help; however in general that can be challenging when the density distribution is non-uniform.

A DTM-Rips based approach.

The Rips complex is defined based on the Euclidean distance between input points, and does not handle noise or non-uniform point samples well. The *distance-to-measure (DTM)* distance is introduced in [26] to provide a more robust way to produce distance field for noisy points. We use the work of [16] to induce a weighted Rips complex from DTM distances, which we now describe briefly. In particular, given a set of points (P, d_P) equipped with metric d_P (for points $P \subset \mathbb{R}^d$, d_P is the Euclidean distance in $\mathbb{R}^d$). For a fixed integer parameter $k > 0$, let $\mathsf{kNN}(p)$ denote the set of k-nearest neighbor of p in P under metric d_P. For each $p \in P$, we set *(DTM-induced) weight* w_p as $w_p = \sqrt{\frac{1}{k}\sum_{q \in \mathsf{kNN}(p)} \mathsf{d}_P^2(p,q)}$, and the *weighted radius of p at scale* α as $r_p(\alpha) = \sqrt{\alpha^2 - w_p^2}$. Now given a simplex $\sigma = \{p_{i_0}, \ldots, p_{i_s}\}$, we define $\rho_w(\sigma)$ to be

$$\rho_w(\sigma) = \min\{\alpha' \mid w_{p_{i_j}} \leq \alpha', \text{ and } \mathsf{d}_P(p_{i_j}, p_{i_{j'}}) \leq r_{p_{i_j}}(\alpha') + r_{p_{i_{j'}}}(\alpha'), \forall j \neq j' \in [0,s]\}.$$

This gives an ordering of all possible simplices formed by points in P (again, edges and triangles are needed), and the resulting filtration is called *DTM-Rips filtration* $\mathscr{F}_{\rho_w}$. Equivalently, consider the DTM-weighted Rips complex $\mathsf{wRip}^{\alpha}(P)$ at scale r defined as: $\mathsf{wRip}^{r}(P) = \{\sigma = \{p_{i_0}, \ldots, p_{i_s}\} \mid \rho_w(\sigma) \leq r\}$. The sequence of $\mathsf{wRip}^{r}(P)$ with increasing scales $r = [0, \infty)$ gives rise to the filtration $\mathscr{F}_{\rho_w}$. The weight ω_p is a certain average distance to the kNN of p and thus intuitively an inverse density estimator (high density points have low weight). Given two points $p, q \in P$, the edge $\sigma = (p,q)$ has **smaller** $\rho_w(\sigma)$ if p and q has lower weight (thus **higher**

density). Simplices spanned by **higher** density points will enter **earlier** into the filtration $\mathscr{F}_{\rho_w}$.

Incorporating data sparsification.

However, the size of weighted Rips can still be large. To this end, we deploy the sparsified version of DTM-Rips developed in [16]. The resulting filtration is denoted by *sparse DTM-Rips* $\widehat{\mathscr{F}}_{\rho_w}(\varepsilon)$ which uses a sparsification parameter $\varepsilon > 0$. See [16] for details of its construction. Our final graph skeletonization algorithm for PCDs, denoted by DM-PCD(P,k,ε,δ), consists of only two steps:

(Step 1). Compute the sparse DTM-Rips filtration $\widehat{\mathscr{F}}_{\rho_w}(\varepsilon)$ using parameters k (to compute DTM-weights of points) and ε (for sparsification).

(Step 2). Apply extDM-graph$(K, \widehat{\mathscr{F}}_{\rho_w}(\varepsilon), \delta)$ to compute the graph skeleton of P, where K is given implicitly as all simplices in $\widehat{\mathscr{F}}_{\rho_w}(\varepsilon)$.

Intuitively, using the DTM-weight alleviates the problem of noisy points (challenge (C-4)), using sparsification addresses the issue of size (challenge (C-3)), while using the entire sparse DTM-Rips filtration allows us to use all radii/scales (instead of a Rips complex at a fixed radius r as in baseline), thereby addressing challenges (C-1) and (C-2). Also, while at a larger radius, the filtration will include edges and triangles spanned by far-away points. Theorem 5.2.5 guarantees that we will output those important loop features using edges that come in as early as possible, i.e., those spanned by higher density points (with smaller ρ_w values) whenever possible. This allows DM-PCD to capture hidden graphs across different scales. See Figure 5.4.

5.5 Experimental Results

We compare our DM-PCD algorithm with the baseline algorithm introduced in Section 5.4, and with SOA graph skeletonization algorithms based on Reeb graph [58] and Mapper [130] (referred to as ReebRecon and Mapper below). (Mapper can produce higher dimensional structures beyond graph skeleton, although often 1D structures are used in practice.) We test on two synthetic point sets and three real datasets. Unless otherwise specified, we use $k = 15$

and $\varepsilon = .99$ in our DM-PCD(P,k,ε,δ); while the persistence simplification threshold δ depends on the point set at hand. For baseline, ReebRecon, and Mapper we report the results of the best parameters we find for them. In particular, a key input for the Mapper algorithm is an appropriate filter function. We tested several standard choices, including distance to base point, eccentricity, density, graph Laplacian eigenfunction and so on, and report the best results found. *For all experiments, all methodologies are run on the original point cloud data, and the figures showing results of higher dimensional data display projections of the results into a lower dimensional space.* Significantly more results and details are in the Appendix.

Overview.

Methods are run on five total datasets - two lower-dimensional (2-D) synthetic datasets, image patches dataset [22] (8-D), time-delay embedding of traffic sensor datasets [21] (6-D), and Coil-20 [106] (16384-D). Our experiments show that DM-PCD is able to extract the true underlying structure of all of these datasets while the other methodologies struggle with noise (image patches and traffic datasets), capturing features at different scales (synthetic and Coil-20 datasets), and having geometrically faithful outputs (synthetic datasets). Additionally, the size of the filtration used by DM-PCD is consistently smaller than that used by baseline.

Synthetic datasets.

We create two synthetic PCDs to illustrate the behavior of our DM-PCD algorithm. Circle dataset contains a noisy and non-uniform sample around a hidden circle with 2050 points. DM-PCD *is able to recover a geometrically faithful hidden circle. The other methodologies, which require more parameters, also recover a hidden circle, but with a less geometrically faithful structure. Additionally,* baseline *requires far greater running time for comparable results.* See Figure 5.3: the output of our method (in (C)) recovers the hidden circle. In comparison, outputs of baseline algorithm over the Rips complex $\text{rips}^r(P)$ at different radius values are shown in (E) – (H). The total number of simplices involved in our sparsified DTM-Rips filtration is $368,276$. The successful baseline result (shown in Figure 5.3 (G)) however requires $7,708,243$ simplices,

which is about **20 fold** increase in size. This results in a drastic run-time difference (2.6 seconds vs. 44.8 seconds) between DM-PCD and baseline. In general, DM-PCD is more efficient than baseline because persistence is computed on a much smaller filtration (see Appendix for fully detailed timing results on all datasets). Also, in general, it is not clear which r to choose for baseline, and if r is too large (e.g., Figure 5.3 (H)), then the output graph is geometrically not faithful any more – this is because long edges are now present in the Rips complex and can appear early in the lower-star filtration in the DM-graph algorithm. In contrast, our output (in (C)) takes advantage of the lex-optimality of the algorithm (Theorem 5.2.5) and thus always uses "good" edges (small edges from high density regions that enter the filtration early) first. The ReebRecon approach also uses Rips complex at a fixed scale r and thus has similar issues with baseline. The Mapper approach (Figure 5.3 (J)) correctly captures topology of the space, but misses some geometric details.

The top row of Figure 5.4 shows the reconstruction from a set of 300 points non-uniformly sampled from two circles (of different sizes) with background noise. DM-PCD *successfully captures both circles, while other methods either fail to capture both circles, or have a topologically correct output that is less geometrically faithful than our method's output.* Our algorithm scans through all scales in the filtration and captures both loop features. In contrast, both baseline and ReebRecon can capture only one loop. Using a small radius r, they can capture the small loop but not the big one. To capture the large loop, they need to use a large radius r (as in Figure 5.4 (C) and (D)), at which point the small loop is destroyed in the Rips complex. Mapper is able to capture both loops, but again some geometric details are lost (Figure 5.4 (E)). See more results in the Appendix.

Image patches dataset.

The image patches dataset from [22] contains $50K$ points in $\mathbb{S}^7 \subset \mathbb{R}^8$, each of which corresponds to a 3x3 image patch [90]. We subsample $10K$ points randomly so computationally we can experiment with Rips complexes at different radii for the baseline. DM-PCD *is the only*

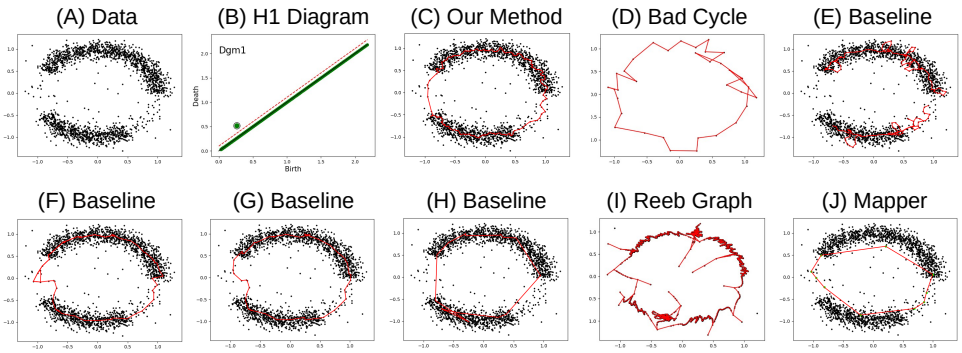


Figure 5.3. (A) Noisy sample of a circle. (B) 1-D persistence diagram w.r.t. our later sparse DTM-Rips filtration. Persistence points are green. Only one point (big point) p has persistence larger than some threshold δ (above the red dotted line). (C) Output of our DM-PCD method with persistence threshold $\delta = .25$, which is a lex-optimal persistent cycle w.r.t. the only high persistence point p in (B). (D) shows a "bad" persistent cycle w.r.t. the same high persistence point p. In contrast, our output in (C) uses good (high density) edges whenever possible. (E) – (H) are outputs from the baseline algorithm using different radius r. (E) $r = 0.1$: The underlying shape (circle) is not yet captured. (F) $r = 0.2$: There are spurious loops that cannot be simplified via persistence. (G) $r = 0.25$: The circle is recovered; however, the size of $\text{rips}^r(P)$ is now 20 times that of our sparse DTM-Rips filtration. (H) $r = 1.05$: The output loses geometric details. (I) is output from ReebRecon using $\text{rips}^r(P)$ with radius $r = .25$ and has much noise. (J) is output from Mapper using graph Laplacian filter ($k = 15$).

method that can extract the true underlying structure from the dataset. All other methods fail to extract any meaningful structure. The projection of points in 3D (Figure 5.4 (F)) is very noisy. However, the analysis of [22] shows that the underlying space has a "three-circle model", with two circles intersecting the third circle twice but not intersecting each other, thus the first Betti number of the underlying space is 5. Our DM-PCD (shown in Figure 5.4 (G)) successfully recovered the same "three-circle model" (with correct $\beta_1 = 5$) directly from raw data **without** preprocessing, and the locations of these (outer, horizontal, and vertical) circles match those shown in [22]. Both baseline and Mapper (in Figure 5.4 (H) and (I)) fail to capture it. (More details in the Appendix.) Results by ReebRecon are omitted for this data set, as the algorithm does not handle background noise well and results are poor.

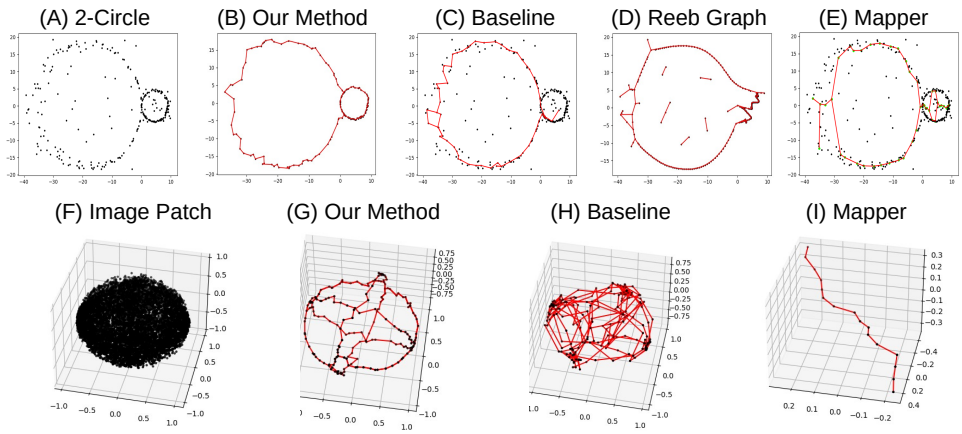


Figure 5.4. Top row: 2-circle data. Output of our DM-PCD method with persistence threshold $\delta = 1.2$ in (B); of baseline in (C), of ReebRecon in (D), and of Mapper in (E). Using a smaller radius r for baseline and ReebRecon will lose the large circle. Mapper output misses geometric details. Bottom row: image patch dataset with projection in $\mathbb{R}^3$ shown in (F). Our output with persistence threshold $\delta = .146$ in (G) captures the 3-circle model (with $\beta_1 = 5$) perfectly. For baseline in (H), further simplification will remove the main circle from the "3-circle model" while keeping all the noisy ones. Mapper (I) (base point filter) is unable to capture any of the loops.

Traffic flow dataset.

We extract two time-series from [21], which are the traffic flows at detector #409529 from the time-range 10/1/2017 to 10/14/2017 and from the time-range 11/19/2017 to 12/2/2017 (including Thanksgiving). Each time-series is mapped to a PCD in $\mathbb{R}^6$ via time-delay embedding as proposed by [113], who also propose that loops in the resulting PCD can be used to detect quasi-periodic behavior in the original time-series data. We note that a normal time range has one major loop, indicating one major periodicity; while the Thanksgiving period has two: a normal one and one that indicates the traffic pattern over the holiday weekend. DM-PCD *recovers these loops much better than* baseline *and* Mapper. Results by ReebRecon are again omitted due to low quality.

Coil-20 dataset.

In our final experiment, we use the Coil-20 dataset provided by [106]. More specifically, we take a subset of 17 objects - removing objects 5, 6, and 19. Objects 5 and 9 are both medicine

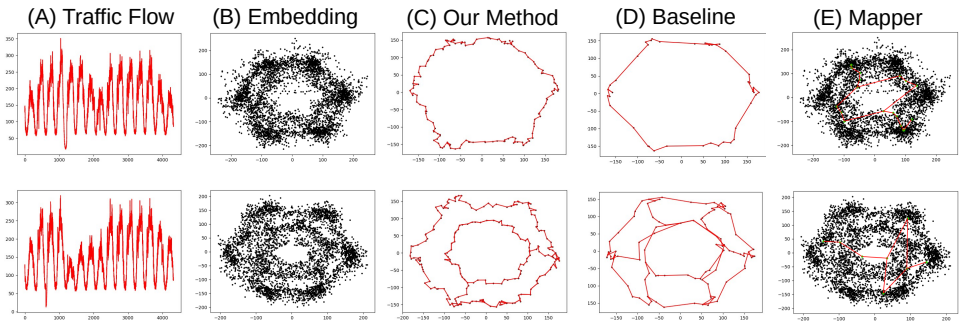


Figure 5.5. Top row: traffic flow for range (10/1/2017 - 10/14/2017) and bottom row is for range (11/19/2017 - 12/2/2017). (A) input time series, and (B) 2d projections of the time delay embeddings of the time-series. (C) Outputs of our DM-PCD algorithm with persistence thresholds of $\delta = 50$ (top row) and $\delta = 12.5$ (bottom row). (D) Outputs of the baseline approach. For the Thanksgiving period, any further simplification will destroy the outer-loop but not the cross connections. (E) Outputs of Mapper with graph Laplacian filter ($k = 15$).

boxes, and objects 3, 6, and 19 are toy cars, and we wanted to evaluate our method's performance on a dataset containing unique objects. We refer to this subset as Coil-17. Following the process used by [99] to convert images to point clouds, we convert each 128 x 128 gray scale image into a 16384 dimensional vector. Hence the input is a set of 1224 points in $\mathbb{R}^{16384}$. Outputs of other methods can be found in the Appendix.

We visualize the data in two dimensions using UMAP dimensionality reduction with L1 metric. Presumably, each class forms a high-dimensional loopy shape. We run DM-PCD using L1 metric with $k = 5$. DM-PCD *is able to capture most of the individiual coils UMAP does, while providing a more correct representation of some classes than UMAP.* Shown in Figure 5.6 is the UMAP reduction with objects uniquely colored (A), the output of our DM-PCD algorithm with a persistence threshold of 0 (B), the output of our DM-PCD algorithm with a persistence threshold of 56 (C), and the output in (C) after removing the critical edges with L1 length above a threshold of 1700 in the 16384 dimensional original space (D). The raw output contains the loops that we would expect to see based on our understanding of the data and the shapes formed in the UMAP projection. The raw output also contains many other edges, revealing more relationships both within individual classes and across multiple classes in the feature space. We remove the longer edges in order to better highlight the features of the output that capture individual objects.

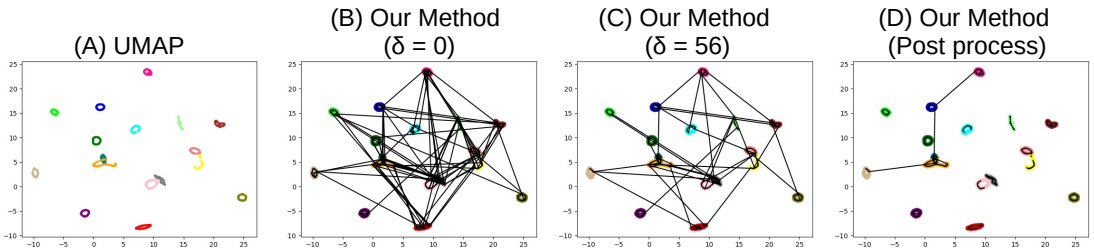


Figure 5.6. Coil: (A) The UMAP projection (using L1 metric) of Coil-17. (B) Output of our DM-PCD algorithm with persistence threshold $\delta = 0$. (C) Output of our DM-PCD algorithm with persistence threshold $\delta = 56$. (D) The output shown in (C) with critical edges of L1 length greater than 1700 removed from the output.

Taking a closer look at Figure 5.6 (D), there are eight objects that the DM-PCD captures in the same exact manner that the UMAP projection does. Close up pictures of these eight objects are shown in Figure 5.7.

There are also four objects that both the DM-PCD output and the UMAP projection capture as loops, but the loops differ between the two methods. Close up pictures of these four objects are shown in Figure 5.8. Object 11 (Figure 5.8 (A)) is a single loop in the UMAP projection, but is actually two full loops in the DM-PCD output. A closer look (Figure 5.8 (B)) at images 16, 17, 54, and 55 shows two separate loops in the output. The L1 distances between these images in the 16384 dimensional original space (1069.2157046029981, 1038.6980409049952, 693.1607849029981, and 566.901973108997) for pairs (16,54), (17,55), (16,17), and (54,55) respectively) do not match the distances between the pairs in the UMAP projections. This indicates that the UMAP projection does not preserve the underlying structure of this object, and that the DM-PCD output containing two loops is correct.

A similar result is obtained for Object 14 (Figure 5.8 (C)), where UMAP projects a single loop and the DM-PCD output contains multiple loops. Objects 11 and 14 are symmetrical, adding further justification that multiple loops is a better skeletonization.

Object 2 (Figure 5.8 (D)) makes a complete loop in the DM-PCD output, but the loop looks incomplete in the UMAP projection. We ran UMAP projections on smaller subsets of

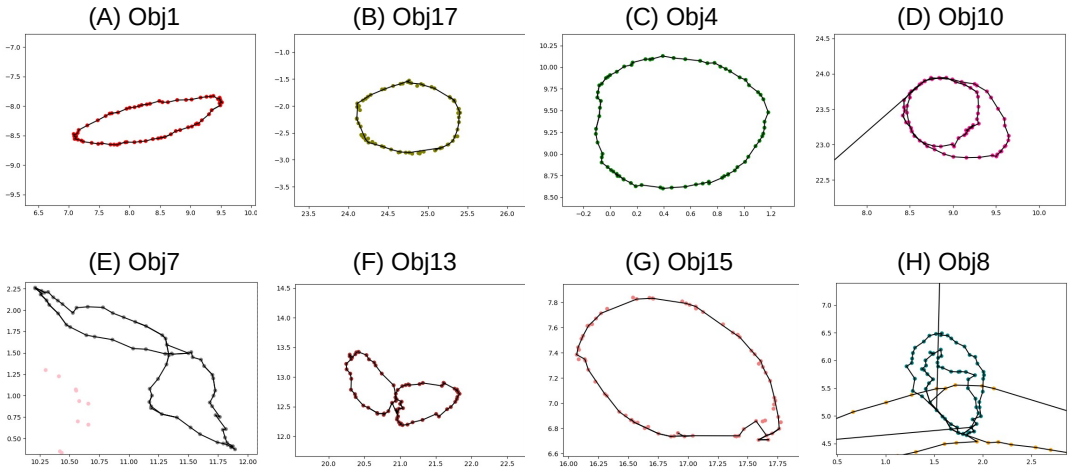


Figure 5.7. Coil: Zoom ins of Figure 5.6 (D) on the eight objects for which our DM-PCD post processed output matches the UMAP projection.

Coil-20, some of which project Object 2 as a clear loop (Figure 5.8 (E)), whereas the DM-PCD output consistently captures Object 2 as a loop.

Object 20 (Figure 5.8 (F)) is captured as the same loop in both the UMAP projection and the DM-PCD output, but the DM-PCD output has an additional edge dividing the loop. The edge connects images 44 and 70, which have a L1 distance of 1329.8000237339966 in the original space. While the other edges adjacent to these nodes are much shorter, other edges that would similarly divide the loop into two are much longer. For example, the L1 distance between images 19 and 58 is 1822.1608110429997. The dividing edge in the DM-PCD output captures this difference, whereas the UMAP projection has no indication of such a difference.

For the remaining five objects, it is not as clear whether or not the DM-PCD output is correct. Close ups of all five objects are shown in Figure 5.9. For each object, the figure shows the output at persistence thresholds $\delta = 0$ (first row), $\delta = 56$ (second row), and $\delta = 56$ with critical edges longer than 1700 removed (third row). Object 3 (Figure 5.9 (A)) and Object 18 (Figure 5.9 (E)) are not captured as a loop in either the UMAP projection or in any DM-PCD

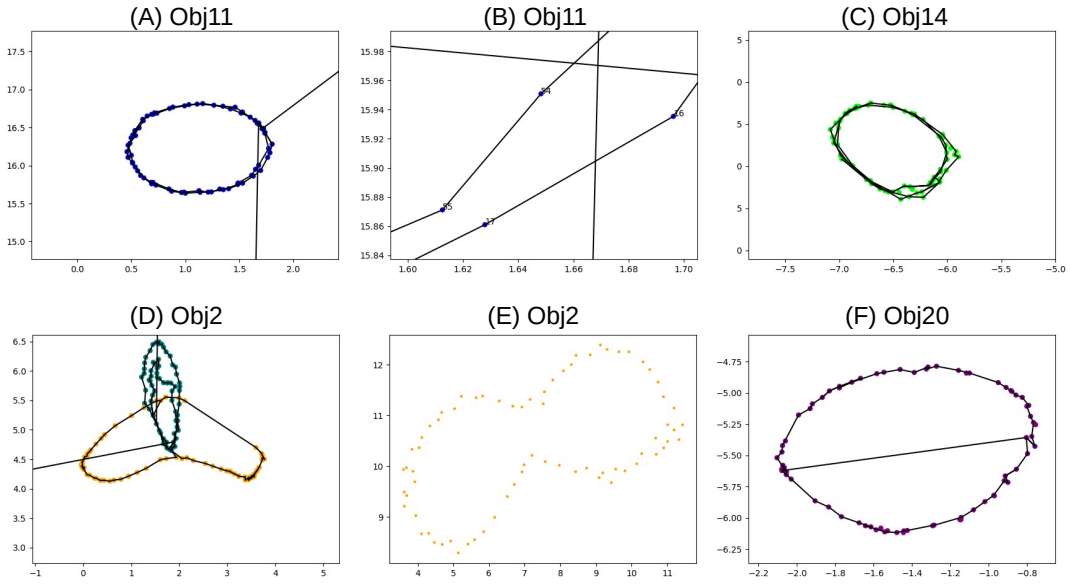


Figure 5.8. Coil: Zoom ins of Figure 5.6 (D) on the four objects for which our DM-PCD post processed output captures a different loop than the UMAP projection. (A) Object 11 - While it appears at first glance that the DM-PCD output matches the loop the UMAP projection contains, a closer look reveals (B) that the DM-PCD output actually captures two loops. (C) Object 14 - Similarly to Object 11, the DM-PCD output contains multiple loops and UMAP projection only captures one. (D) Object 2 - UMAP projection is not a complete loop, but DM-PCD produces a complete loop. (E) Object 2 - UMAP projection of only Object 2's images - a complete loop is visible. (F) Object 20 - UMAP projection shows a single loop, while the DM-PCD output captures the same loop, but has an additional edge dividing the loop.

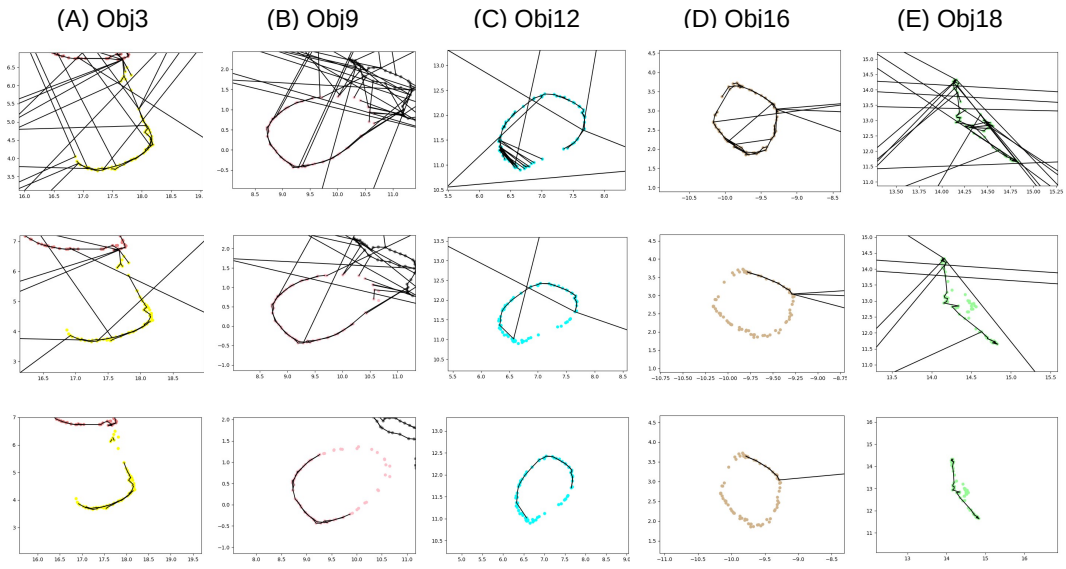


Figure 5.9. Coil: Zoom ins of DM-PCD outputs with persistence thresholds $\delta = 0$ (first row), $\delta = 56$ (second row), and $\delta = 56$ with critical edges longer than 1700 removed (third row). The objects of focus are (A) Object 3, (B) Object 9, (C) Object 12, (D) Object 16, and (E) Object 18.

output. The arcs appear to follow the arcs embedded in the UMAP projection. Object 9 (Figure 5.9 (B)) does appear as a loop in the UMAP projection, but is captured as a (double) arc by DM-PCD. Object 12 (Figure 5.9 (C)) is captured as a loop in UMAP, but is not in any DM-PCD output. However an arc spanning most of the loop is clearly captured. Under different parameters, DM-PCD was able to extract a loop. Object 16 (Figure 5.9 (D)) is captured as a loop in UMAP, and a loop is only captured by DM-PCD with persistence threshold $\delta = 0$.

A final note on comparing DM-PCD to UMAP projections - the metric distortion of UMAP became apparent when viewing the DM-PCD outputs. There is metric distortion within classes - such as Object 20, where there appears to be an extra edge in the DM-PCD output because the UMAP embedding does not preserve the distances in the original space. There is also clear metric distortion in the UMAP embedding with respect to object relationships. For example, before removing critical edges with length greater than 1700, both Objects 4 and 16 have an edge that connects to Object 8. However, once the thresholding is applied, the edge connecting Objects 4 and 8 is removed and the edge connecting Objects 16 and 8 remains. This

would indicate that Object 16 is closer to Object 8 than Object 4 is, but Object 4 appears closer in the UMAP embedding.

5.6 Conclusion

We generalized the DM-graph reconstruction algorithm to arbitrary filtrations, proved that the output of this generalized algorithm is meaningful, and developed a method for graph reconstruction from high-dimensional PCDs. Empirical results demonstrate the effectiveness of our DM-PCD approach.

Time complexity is the main limitation of our approach. The time to compute persistence is a function of the size of the input filtration. While the theoretical worst case running time is cubic in this size, in practice modern implementations (such as PHAT) perform in subquadratic time (we observe near-linear growth of time w.r.t. the size of simplicial complex in our experiments). Hence reducing the size of filtration is crucial in practice. While the sparsification strategy we used in this chapter helps to bring down the size of filtration, the reduction might not be significant enough for very large or more challenging datasets than what we experiment with in this chapter.

In addition to running time causing potential limitations, the raw output of our DM-PCD method must be connected. As shown in Coil-20, with some post-processing we were able to capture individual objects quite easily - but the proper post-processing approach will depend on individual datasets and may not be so straight forward.

5.7 Data and Code Availability

Code for both our new methodology and the baseline approach is publicly available at https://github.com/lucasjmagee/PCD-Graph-Recon-DM. The repository also contains all datasets used in this manuscript.

5.8 Reprint

Chapter 5, in full, is a reprint of the material as it appears in Graph skeletonization of high-dimensional point cloud data via topological method in Journal of Computational Geometry. Magee, Lucas; Wang, Yusu. 2022. The author of the dissertation was the primary investigator and the author of this article.

Chapter 6

scRNA-seq Graph Extraction Algorithm

In the previous chapter (Chapter 5), we developed a generalized DM graph reconstruction algorithm and combined that with the sparsified weights Rips filtration of [16] to develop a graph extraction algorithm for high-dimensional PCDs. In this chapter, we develop an efficient and effective algorithm to construct the graph skeleton of high-dimensional scRNA-seq datasets. We include several experimental results that empirically demonstrate the outputs of our algorithm capture meaningful structure. We then use the output graphs of our algorithm to define cell type identity, study gene expression gradients, construct cell type taxonomies, and analyze gene expression changes along Alzheimer's disease progression.

6.1 Introduction

The development and application of single-cell and spatial genomics approaches in neuroscience has led to the generation of large high-dimensional datasets and atlases of the brain in multiple species [156, 157, 155, 120, 74, 95]. Single cell and nucleus RNA-seq (sc/snRNA-seq) are routinely used to study anatomic and transcriptomic structure [156, 155] developmental stages [138], evolutionary trajectories [108], disease states [101, 95], and a variety of other experimental conditions. To understand these data requires quantification of transcriptomic and anatomic relationships across structures and biological conditions [36].

Extracting meaningful structure from high-dimensional datasets however is challeng-

ing, and there are challenging issues in analyzing, representing, and visualizing single cell or nucleus RNA-seq to deduce interpretable and reproducible structure (Armand et al. 2021). Representation learning methods are routinely used to analyze these complex, high-dimensional data by projecting them into lower-dimensional embeddings [73] that facilitate interpretation of the structure and dynamics of cell heterogeneity [73, 154]. Routinely applied methods seek to dramatically reduce the dimension of the ambient space including popular techniques such as t-distributed stochastic neighbor embedding (t-SNE) [35], isomap [135], locally linear embedding (LLE) [125], Uniform manifold approximation and projection (UMAP) [8], Laplacian eigenmaps [140], and several others [73, 154].

There are limitations in our intuition about the properties and structure of high dimensional data [79]. Interpreting and visualizing high-dimensionality datasets require methods to remove noise and facilitate exploratory data analysis [32], and analysis of single cell omics data shares challenges common to all data analysis in high dimensions. The curse of dimensionality, whereby as the number of dimensions and features increases, data becomes increasingly sparse and less representative, leads to finding redundant or non-relevant features, overfitting, and intensive computational demands [36]. A common genomics workflow is to apply multivariate linear or other non-linear machine learning techniques to reduce dimensionality to tens or hundreds of dimensions, derive analyses or associations, and then finally to apply t-SNE and UMAP to arrive at a visual depiction of clusters in 2 or 3 dimensions. Low dimensional embeddings are ubiquitously used in the literature to directly generate cluster or cell type assignments [83] and to infer properties of clusters (their heterogeneity, separation, or similarity) [114, 156, 3]. Unfortunately, extreme dimension reduction introduces significant distortion of high-dimensional datasets [25], and while these methods claim faithful representation of local and global structure in low dimensions, there is evidence they also fail in this regard [35].

Modeling and visualizing major differences in transcriptomic structure in low dimensions has interpretive value and has led to advances in our understanding of numbers of cell types in the brain and their taxonomic structure [156, 157, 120, 3, 74, 95, 11]. While progress in

deriving these taxonomies has been immense, studies have shown that use of low dimensional embeddings creates extreme local metric distortion that may obscure fine structure relationships [25]. For more precise quantification of cell type properties, such as the nature of cell type gradients and their relationship to cell type, it is preferable to work directly in the ambient space where the original distance metric holds. Although extracting meaningful structure directly from high-dimensional transcriptomic data without preliminary dimensionality reduction is appealing it is not entirely straightforward.

Several novel approaches have been proposed to capture the underlying structure of high-dimensional transcriptomic datasets [95, 158, 18]. A promising approach is the application of topological data analysis (TDA) [48, 23, 44]. TDA proposes to analyze real world datasets based on concepts from topology, a field that describes geometric properties preserved under continuous transformations. In the discrete setting, modeling using simplicial complexes, generated by triangulations of the data based on neighborhood relationships, is used to define and represent the underlying structure. Persistent homology, an essential concept in TDA, is used to quantify meaningful topological features that are preserved by varying a threshold with respect to some desired metric. In single cell genomics we are interested in quantitative differences in gene expression between cells and cell types and understanding how robust these features are. Here, TDA enables prioritization of features retained through persistence, while reducing noise of individual cell measurements.

We present single cell Discrete Morse Graph Reconstruction (scDMGR) combining persistent homology with discrete Morse theory, with key modifications for single cell omics data, for extracting meaningful graph structure and cell type relationships from scRNA-seq datasets. Recent theoretical advances [96] and code optimizations, now make scDMGR a viable approach for problems in single cell genomics. scDMGR constructs a compact 1-D spanning representation, a Morse graph, of the underlying data in the ambient space, as well as a gradient field for the entire dataset such that every cell flows via an underlying gradient path into a unique cell on the graph. Local maxima of the Morse graph itself represent putative cell types,

and transitions between types are optimal paths of individual cells connecting peaks through saddles. In this manner, scDMGR models high dimensional scRNA-seq data without need for intermediate dimension reduction (e.g. PCA, t-SNE, UMAP). Although the Morse graph lies in the ambient gene expression space, the graphs are one dimensional and can be visualized, in the same appealing way as a UMAP. However, local transcriptomic structure is preserved, as every edge on the graph is part of the original neighborhood graph in the high-dimensional gene expression space.

The scDMGR method is useful for studying the basic structure of cell type identity, observed transcriptomic gradients and continuous variation between cell types, and other relevant aspects of cell type differentiation such as perturbations in disease. After first overviewing this approach, we illustrate an application to cell type identity using whole cortex single cell in the mouse [157]. A particular strength of scDMGR is the analysis of transition states or gradients between potential cell types. To demonstrate this, we analyze data from GABAergic hippocampal CA1 interneurons [81] where a transcriptomic gradient in express was found in Cck expressing cells. scDMGR finds a single cell path representation of this gradient that enables more detailed characterization. The inherent graphical structure of Morse graphs provides a graph-based taxonomy that captures complex cell type relationships, as well as a method for evaluating annotation from other studies. We illustrate this by deriving a novel whole brain GABAergic taxonomy of the mouse brain [157], and evaluate the reproducibility of cell types labeled through this study. In an application of scDMGR to translational studies of Alzheimer's disease [133, 101, 94, 104], we use single nucleus data from a recent study [133, 101, 94] to help distinguish between cell type loss and modified transcriptional state.

6.2 Methods

6.2.1 Topological Data Analysis

We use several concepts from topology and discrete Morse theory, references include [44, 118, 127]. We construct a simplicial complex spanned by input points (cells) in the input ambient space based on k-NN. A simplicial complex is an object consisting of smaller building blocks called simplices. A 0-simplex is a vertex, a 1-simplex is an edge, and a 2-simplex is a triangle, a 3-simplex is a tetrahedron, etc. In this work, we only need to build 2D simplicial complexes. Specifically, we use the 2D k-NN complex, which is the set of vertices, edges, and triangles induced by the k-NN graph (Euclidean distance).

6.2.2 Lower-Star Filtration

A filtration is a finite sequence of simplicial complexes connected by inclusion. A simplex-wise filtration is a filtration in which for every two consecutive simplicial complexes K_i, K_{i+1} in the sequence there is only a single additional simplex in K_{i+1}. The lower-star filtration is a filtration in which the simplices are added in increasing value with respect to some descriptor function. We use the lower-star simplex-wise filtration with respect to (inverse) Jaccard index on the 2D k-NN complex. The Jaccard index is defined on edges (u,v), value at vertex u is the maximum of incident edges, the value on a triangle is the minimum of its three edges. For more information on filtrations, and persistence computation used within these algorithms, see [44, 118, 127].

6.2.3 scRNA-seq DM Graph Reconstruction Algorithm

The original discrete Morse graph reconstruction algorithm was developed to extract the true underlying graph structure of density fields [38, 63, 147, 134, 123]. Within this algorithm, the lower-star filtration of the discretized input domain with respect to the (negated) input density function is computed. The final output is the mountain ridges, also known as the unstable 1-

manifolds, of the input density function. [96] observe that the output of the original algorithm is entirely dependent on the lower-star filtration computed at the beginning. The major contribution is a generalized DMGR algorithm (Algorithm 7 of Chapter 5) that takes any filtration as input, allowing for a wider variety of input domains and descriptor functions to be utilized in graph extraction. Lex-optimality of output graphs is proven, showing that the output of the algorithm contains meaningful information with respect to the input. A PCD algorithm is provided, but the weighted Rips filtration used is not so informative for scRNA-seq datasets. Thus we developed Algorithm 8. Here, δ is a noise parameter, with higher values resulting in more restricted graphs.

Algorithm 8: scDMGR(X, k, δ)

Input: Normalized gene by cell matrix X, neighborhood value k to build k-NN complex, persistence threshold δ

Output: A reconstructed graph G_δ

(Step 1) Compute 2D k-NN complex K on cells of X (euclidean metric)

(Step 2) Build lower-star filtration $\mathscr{F}_K$ of K w.r.t. (inverse) Jaccard index of edges in K.

(Step 3) extDM-graph($K, \mathscr{F}_K, \delta$)

6.2.4 Jaccard Index

Jaccard index is a meaningful descriptor function for scRNA-seq datasets, and capturing the mountain ridges of such function provides meaningful interpretation with respect to cell type identity. If a cell type is truly a cell type, there should be a set of cells with similar gene expression corresponding to that type. Because these cells have similar gene expression, they should be k-NN with each other. Thus, individual cell types should correspond to relative maxima of Jaccard index on the k-NN graph. Relative maxima of Jaccard index are captured by the scDMGR output graphs.

Additionally, if two cell types are truly distinct, then these cell clusters will have a meaningful drop in Jaccard index separating the two, i.e. "peak-saddle-peak." The Morse graphs of scDMGR capture relative maxima and drops in Jaccard index(saddles) through optimal gradient paths that measure the fastest changes in gene expression separating types. Depth of

the gradient path to the saddle can be interpreted as measuring the strength of evidence for distinction between the two related types.

Although for interpretive purposes, we want to compute the unstable 1-manifolds (mountain ridges) of the Jaccard index, it is computationally easier to compute stable 1-manifolds (valley ridges). Stable 1-manifolds of the inverted Jaccard index are what is explicitly computed in the Algorithm 7, but is equivalent to the unstable 1-manifolds of the original function.

6.2.5 Wasserstein Distance

We use the Wasserstein distance originally proposed in [142] to compare the distributions of cells. We use the implementation from the Python Optimal Transport library [50].

6.2.6 W-L Graph Distance

To compare these Morse graphs, we use the Weisfeiler-Lehman (W-L) distance [31], an optimal transport-inspired metric for graph data with node features, combined with the classical Weisfeiler-Lehman graph isomorphism test [149].

6.2.7 Soft Margin Support Vector Machines (SVMs)

We use a soft margin SVM to distinguish excitatory and inhibitory classes.

6.2.8 k-Label Cohesiveness Index (k-LCI)

Given a graph $G(V,E)$, and a label function $L : V \to \{l_1, l_2, ..., l_n\}$ defined on its vertices, the k-LCI of label l_i is defined as k-LCI$(l_i) = \frac{\sum_{v \in V_{L_i}} |\{u \in N_k(v) | L(u) = l_i\}|}{\sum_{v \in V_{L_i}} |N_k(v)|}$, where V_{L_i} is the set of vertices in the graph with label l_i, and $N_k(v)$ reachable from v in k edges excluding v itself. k-LCI(l_i) therefore measures the fraction of neighbors of each $v \in V_{l_i}$ reachable within k edges in the graph that have the same label l_i.

6.2.9 Morse Graph Separability

To measure separability of anatomic regions using Morse graphs we define separability as the percentage of triplets of the form (a,b,c), where a and b represent elements of the same class, and c represents an element from a different class, such that the distance $d(a,b) < min\{d(a,c),d(b,c)\}$.

6.2.10 Morse Persistence Taxonomy (MPT)

We represent our proposed taxonomy as a N x N, where each row and column correspond to a single type. The value of the i-th row and j-th column of MPT is equal to the highest such persistence threshold such that the Morse graph contains at least three non-critical edges between the i-th and j-th cell type. For full GABAergic MPT is computed for each of the 6 subclasses (Lamp5, Pvalb, Sncg, Sst, Sst Chodl, and Vip) individually, as we explicitly show for Sst, as well as for all pairs of the 6 subclasses. MPT matrix entries corresponding to two cluster types belonging to the same subclass is equal to the corresponding value from the appropriate subclass MPT. The values for entries in the matrix corresponding to two cluster types belonging to two different subclasses is equal to the corresponding value from the appropriate subclass pair MPT.

6.2.11 Constellation Map Embedding

From the MPT cluster types that are outliers for both 1-LCI score and for highest lasting persistence threshold are removed. MPT is input to multidimensional scaling to achieve the coordinates for each type in the constellation. The size of the vertex for a cell type is scaled as max 1-LCI score across Morse graphs of all persistence thresholds. Louvain clustering to determine the optimal number of supertypes and the cluster types that belong to each. Finally, an edge is drawn between two cluster types if the corresponding entry in MPT is $\geq .05$ and at least 75% of the max value of either cell type's row in MPT, and where thickness of the line scales by corresponding MPT entry.

6.3 Datasets

6.3.1 Mouse Hippocampus CA1 Cell Dataset

We took the 465 Cck expressing cells of the hippocampal CA1 inhibitory neurons from [81]. We preprocessed the dataset by taking log(CPM + 1) of the raw gene expression of 27,998 genes. k = 15 was used for all k-NN complexes. For more information, see [81].

6.3.2 Mouse Cortex and Hippocampus Dataset

We took the 1.3 million from the adult mouse isocortex and HPF from [157]. We preprocessed subsets of interest by taking log(CPM + 1) of the raw gene expression. For experiments restricted to a single subclass, we restricted to the top 2,000 most differentially expressing genes by ANOVA. For experiments run on subsets of cells covering multiple subclasses, we restricted to the top 4,000 most varying genes. k = 25 was used for all k-NN complexes. For more information, see [157].

6.3.3 Human Sea-AD Dataset

We took 992,281 cells belonging to 109 out of 139 supertypes from the Sea-AD dataset in [101], selected from GABAergic and glutamatergic classes. We preprocessed the set of cells belonging to each supertype by taking log(CPM+1) of the raw gene expression. We restricted to the top 2,000 most varying genes. k = 15 was used for all k-NN complexes. See [101] for more information.

6.4 Discrete Morse Graph Reconstruction for Omics Data

The scDMGR algorithm proposed here is based on techniques of topological data analysis (TDA) and Morse theory, summarized in Methods [48, 44]. Briefly, Morse theory allows us to study the critical points of functions on manifolds, and a combinatorial adaptation of Morse Theory, called discrete Morse theory (DMT), can be applied to any simplicial complex defined on

data points [51, 127]. Recent work in DMT has developed an approach to extract the underlying data graph skeleton from a given density field [38, 63]. Here, DMT is used to extract the one-dimensional structure (unstable 1-manifolds) that captures the ridges of the density field, which are returned as a characteristic description of the salient features of the data. A continuous threshold technique (persistent homology) is used to denoise the output, returning only the most significant peaks (Methods). The work of [96] derives a DMT algorithm for efficient graph reconstruction to infer a graph skeleton for high-dimensional point cloud datasets.

While implementations of this algorithm work well on lower dimensional datasets, performance is degraded on datasets of scale encountered in single cell genomics. To run DMT on omics data we introduce a data structure (called a lower star filtration (Methods)) having two properties: 1) a simplicial complex of points, edges, and triangles that capture k nearest neighbor relations, and 2) an ordering of the simplices subject to a metric to be described below. The scDMGR algorithm constructs a (discrete) gradient vector field for the entire dataset, and every cell flows via the underlying gradient path into a unique cell on the Morse graph, called the Morse representative of that cell.

Figure 6.1A-E summarizes the proposed scDMGR workflow for our applications. Starting with a cell by gene matrix with log normalized CPM data, we select genes with significant differential expression (Figure 6.1A) (Methods) which determine the dimension of the ambient space. We then construct the *k*-nearest neighbor complex of vertices (cells), edges, triangles ordered by inclusion using Euclidean distance in ambient space, (shown in (Figure 6.1B) for Sst cortical interneurons [157]. In the *k*-NN complex, edges (pairs of cells u,v) are assigned a similarity score, the Jaccard index J(u,v), defined as the fraction of shared neighbors of u and v, and the Jaccard index of a cell is defined to be the maximum J of all incident edges. Intuitively, two cells are close if they share a large fraction of nearest neighbors in gene expression space (Figure 6.1C). The filtration is ordered by Jaccard index, where a relative maximum of the Jaccard index on the *k*-NN complex corresponds to a peak in density of cells sharing a common transcriptional profile. The scDMGR algorithm then computes the persistence pairings with

respect to the filtration and outputs the Morse graph (stable 1-manifold) consisting of saddles and optimal gradient paths to the peaks they separate (Figure 6.1D). Thus, instead of reducing the dimensionality of the data, scDMGR selects a subset of samples that best capture the underlying topology of the data. The Morse graph can be visualized in other representations such as UMAP representation (Figure 6.1E), and has an intuitively appealing interpretation conceptualized as a landscape of cells with unique expression (types) and ridges as gradient transitions between these types (Figure 6.1F).

There are many results on the inaccuracies of local and global structure (Figure 6.1E) and limitations on metric distortions in the application of nonlinear dimension reduction of high dimensional data [35, 83, 25]. We illustrate this local distortion in Figure C.1 which shows Cck expressing cells in mouse hippocampus CA1 [81]. Overall, t-SNE distance correlates with ambient Euclidean distance ($\rho = 0.6, p < 2.2x10^{-16}$), while local 15-nearest neighbors of a given point correlate only at ($\rho = 0.1$, $p < 2.2x10^{-16}$). More than half of cells have 2 or fewer nearest neighbors preserved in the embedding.

In contrast, the scDMGR representation better maintains the original space distance structure and preserves neighborhood relationships. The underlying Morse graph is a subgraph of the k-NN complex and is a compact and faithful representation of the data, and not very sensitive to the number of nearest neighbors k chosen (Figure C.1). Figure 6.1G plots the fraction of cells from the full Sst dataset that live on the Morse graph as a function of persistence threshold (green curve). There is a high fraction of all cells at lower persistence threshold on the graph with decreasing number as the threshold increases. Compactness of the Morse representation is indicated by a large fraction of cells that have a 25-NN nearest neighbor on the Morse graph (blue) at each persistence threshold. This is significantly higher than the fraction of cells with a 25-NN representation of random sets of cells of equivalent size (mean, red).

Although GABAergic interneuron cell types have a common developmental origin, their expression patterns are less correlated with spatial location than for glutamatergic neurons [158]. Nevertheless, cortical regions still have identifiable and differentiating transcriptomic

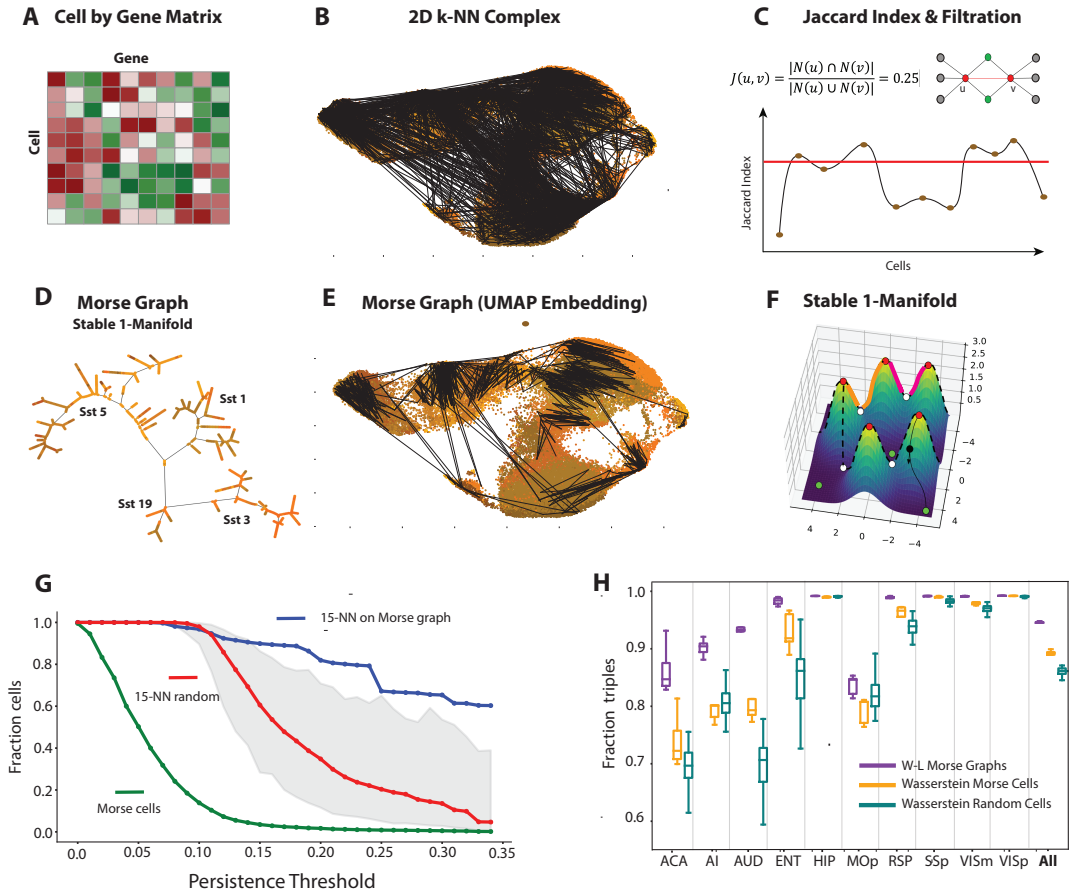


Figure 6.1. (A) Log normalized cell by gene expression matrix, (B) 25-NN complex including cells (vertices), edges, and triangles using Euclidean distance, sparser for viewing. (C) Jaccard index for each edge computing the fraction of shared neighbors in k-NN graph. Jaccard index provides a filtration input to scDMGR algorithm. (D) scDMGR constructs the Morse graph in the ambient gene expression space consisting of saddles and optimal gradient paths on J(u,v) to the peaks they separate (unstable 1-manifold) and providing the structure for cell type interpretation. (E) Morse graph superimposed on UMAP embedding of Sst cells shows preservation of longer-range connections with significant noise in short range distances. (F) Intuitive conceptualization of Morse structure as traversing peaks and saddles through gradient paths. Two peak saddle peak paths are shown (orange, red). (G) Fraction of cells from the full Sst dataset as a function of persistence threshold (green curve). Fraction of cells that have a 25-nearest neighbor on the Morse graph (blue) at each persistence threshold, 1000 random sets of cells of equivalent size (mean, red) capture the entire dataset poorly. (H) Weisfeiler-Lehman (W-L) as a measure of graph distance, and Wasserstein (W) distance (Methods) between sets of cells, confirm transcriptomic identity of regions. Fraction separability using these metrics captures regional transcriptomic identity of 10 cortical regions,and all 18 regions [157]. Wasserstein distance on Morse cells (yellow), and Wasserstein distance on random subsets of cells (green) show that the Morse graph structure (W-L metric) of cells captures regional identity significantly better than density-based cell type estimates (W metric), as well as the W metric on random cell sets.

signatures [70]. To illustrate that the Morse graph differentiates the identity of cortical regions we performed the following experiment using Vip positive cells from 18 cortical regions including hippocampus [157]. Ten subsets of 500 cells from each region were used to construct 180 Morse graphs representing the cell type structure in the 18 regions. To compare anatomic identity based on gene expression, we compare distances between Morse graphs using Weisfeiler-Lehman (W-L) [31] and use the Wasserstein (W) from optimal transport [142] to compute distances between sets of cells.

For each cortex region R, we choose two subsets (A,B) from R and the third C from a different region. Percent separability is defined as the fraction for triplets for which W(A,B)¡ min W(A,C), W(B,C) and indicates that A and B are more similar in transcriptomic structure than either is to C. Figure 6.1H shows results for 10 regions (ACA, AI, AUD, ENT, HIP, MOp, RSP, SSp, VISm, VISp) and for all 18 regions [157]. Scoring using the W-L graph distance to compare Morse graphs, Wasserstein distance on Morse cells (i.e., cells in the Morse graphs), and Wasserstein distance on random subsets of cells, shows that the Morse graph structure (W-L metric) of cells measures regional transcriptomic identity significantly better than density based cell type estimates (W-metric), as well as the W metric on random cell sets. Over all regions, W-L graph distance attains 0.95 mean accuracy, Wasserstein distance 0.88, and random sets 0.85, illustrating that Morse graphs capture the regional identity of cell type architecture via cells on the graph, which is further improved using the edges that define their relationships.

6.5 Discrete Morse Graph Reconstruction: Cell Type Identity

Effective methods for profiling thousands of cells or nuclei at a time and simultaneous computational advances make it is now possible to systematically classify and characterize the transcriptomic diversity of neural cells [117, 56]. A transcriptomic cell type can be defined by a (potentially small) group of cells with similar gene expression profile [14]. A collection of cells

sharing similar transcription will have common neighbors in the k-NN graph and therefore pairs with higher Jaccard index. A key feature of scDMGR is that that Morse graph paths respect the gradient structure of the gene expression metric and are the most rapidly changing transitions between peaks or types. Therefore, in the Morse approach a cell type is a representative cell with a local maximum in Jaccard index. The Morse graph is optimal in the sense that while these edge paths occur in the initial k-NN graph they will not remain in the Morse graph if Jaccard index is low [96]. In this way, we define a Morse cell type as a cell achieving a local maximum in Jaccard threshold. Second, if two cell types are truly distinct, then these cell clusters will have a meaningful drop in Jaccard index separating the two, i.e. "peak-saddle-peak." The Morse graphs of scDMGR capture relative maxima and drops in Jaccard index (saddles) through optimal gradient paths that measure the fastest changes in gene expression separating types. Depth of the gradient path to the saddle can be interpreted as measuring the strength of evidence for distinction between the two related types.

To illustrate the Morse cell type approach we consider two somatostatin (Sst) positive cell type clusters identified in the mouse cortex (Sst 88, Sst 91) [157]. Both types express Neuromedin B Receptor (Nmbr) and originate in the medial ganglionic eminence (MGE), and consist of (n=2775,1515) cells. Highlighting these cells within a UMAP embedding of all 45,467 Sst cells is shown in Figure 6.2A, with Sst 88 (orange), Sst 91 (brown). A Morse path (peak-saddle-peak) (L=44) from the full Morse graph M having 239 cells shows the distinction between the Sst 88 and Sst 91 and a loss of local distance relations in the UMAP representation. A histogram of maximum incident Jaccard index of all cells of M is shown in 6.2B together with the labels of several local maxima. As there are cells occurring as local maxima of type Sst 88 and 91 we recognize these as representative Morse cell types, where the inset shows most of these cells are in primary motor cortex MOp.

To investigate the relationship of these two labeled Sst 88 and 91 cells, we examine the optimal Morse path L connecting them (Figure 6.2C) with the saddle edge (dashed red) and regional cortical identity of cells. The path illustrates the fraction of shared neighbors (Jaccard

index) of each cell along the gradient between the Sst 91 labeled cell to the left of the saddle and Sst 88 to the right. The fraction of common neighbors along this path indicates a more continuous transition in expression between these cells. The mean of highest differentially expressing genes ($n = 50$) between [157] defined cell clusters Sst 88 and Sst 91 (Figure 6.2D) shows that the Morse path captures the continuous cell type transition. The curves cross precisely at the saddle edge, confirming the distinction of these types.

The neuronal development and signaling gene Reln is a dramatic marker of the distinction between these types (Figure 6.2E), while kinase family Erbb4, implicated in several neurocognitive disorders [151, 153], has a reverse differential at the saddle. More continuous expression transition is illustrated by Edil3 and Mgat4c in Fig. 2F. These results show how scDMGR identifies specific cells as a representative cell type and a path of cells having the largest transcriptomic variation in ambient space between these types. A gene ontology analysis [30] of differentially expressed genes shows cell type Sst 91 significant for synaptic structure and organization (FDR BH $q < 1.88x10^{-8}$), and Sst 88 more weakly associated with ion channel activity ($q < 8.93x10^{-3}$).

6.6 scDMGR and Cell Type Gradients

Molecular gradients across areas in the cortical sheet, likely a remnant of brain development [143], were described in earlier studies in human bulk tissue profiling [70, 122]. Single cell genomics surprisingly shows that cell types that are specific to a cortical area or shared amongst areas often exhibit gradients in distribution or gene expression across those areas [157, 24]. Whether cortical cell types represent discrete classes with sharp inter-class boundaries, or points along a continuum is actively studied [156, 158, 81].

The scDMGR approach provides a principled way to describe and quantify transcriptomic gradients between putative types without preliminary dimension reduction. To illustrate this application to gradient analysis we used data in the mouse hippocampus CA1 [81] that identified

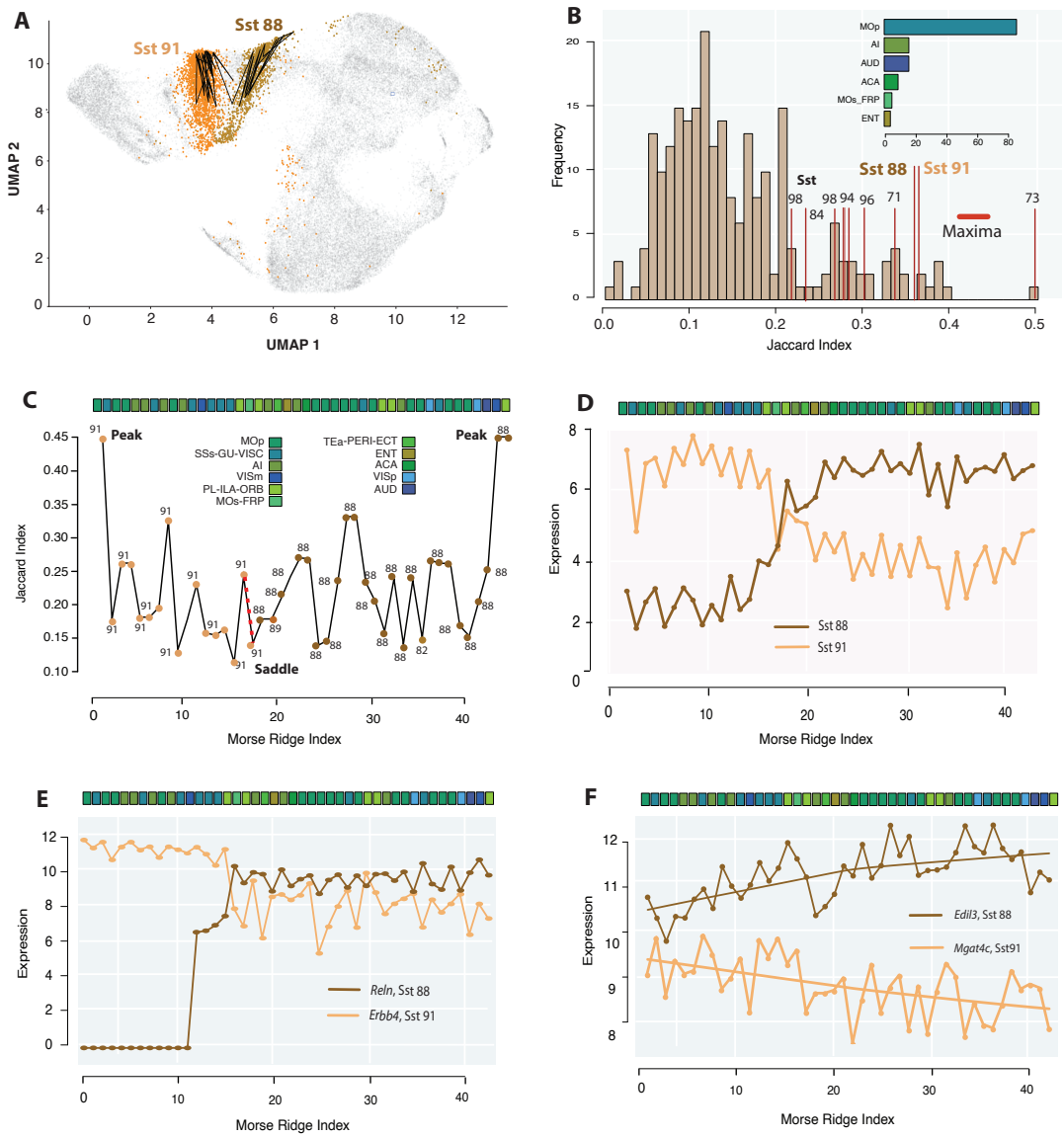


Figure 6.2. (A) UMAP projection of Sst cells from [157] with Sst 88 and Sst 91 cells labeled brown and orange and projection of a single Morse path of scDMGR graph computed on the Sst cells in ambient space. (B) Histogram of maximum incident Jaccard index of all —G—=239 Morse cells is shown with maximum attained for several cells labelled Sst types. Cells labeled have maximum density for the indicated type. Inset shows anatomic distribution cells from cortex MOp, AI, AUD, ACA, MOs FRP, and ENT. (C) Optimal Morse path in M (L=44) connecting two local maxima for Sst 88, Sst 91 with the saddle edge shown in dashed red and regional location of cells. Anatomic abbreviations from [157]. (D) Mean gene expression trajectory of the highest 50 differentially expressing genes (Methods) between cells from clusters Sst 88, Sst 91 showing specific transition at Morse saddle. (E) Genes Reln and Erbb4 illustrate strong differential between these types, with more continuous variation (F) of Edil3, Mgat4c between types.

10 major GABAergic types divided into 49 fine-scale clusters. This division into discrete classes, however, was found insufficient to describe the diversity of these cells, as continuous variation also occurred between and within classes. A primary cluster exhibiting this effect consists of 465 Cck Cxcl14 expressing cells suggesting more continuous gradation in expression between GABAergic types Slc17a8, Calb1 Tac2, Calb1 Kctd12, Calb1 Igfbp5, Calb1 Tnfaip813, and Vip (Figure 6.3A, Figure C.2). Harris and colleagues [81] described a gradient using a latent factor model that predicts the expression of genes using a single continuous variable fit by maximum likelihood. To examine this gradient at cellular resolution, a Morse graph using their gene set (g = 27,998) at persistence threshold $\delta = 0.2$ was chosen to produce a path L spanning the observed latent space gradient. The path has 26 cells with three peaks and two saddles (Figure 6.3A). The inset of Figure 6.3A presents the mean correlation of point distances in ambient Euclidean, PCA reduced, t-SNE, and Morse Jaccard based and shows that Jaccard distance has favorably preserved local point structure over t-SNE.

Figure 6.3B shows how the latent factor value (LFV) changes weakly non-linearly (Adj. $R^2 = 0.61$, $p = 0.074$) with the gradient Morse path. On the Morse path with peaks (red) and saddles (green), Slc18a7, Calb1 Tnfaip813, and Vip are seen as the primary Morse types. The Morse path allows us to describe the transcriptomic characteristics of the gradient from Slc17a8 to Vip. Over this path non-zero mean expression (Figure 6.3C) increases (solid black) through Tnfaip813 to Vip, while the expression variability of these genes decreases. Further, there are fewer genes expressing along the gradient (Figure C.2B) for a net result of a smaller number of highly expressing genes with increasing cell type specificity defining types Slc17a8, Tnfaip813, to Vip.

Harris and colleagues [81] identify 40 genes functionally associated with axon targeting with expression along the gradient. Correlation of these genes with the Morse path agrees well with the latent factor (Figure 6.3D) (Spearman 0.828) although with some variation (Adj. $R^2 = 0.554$, $p < 1.18x10^{-7}$). In agreement with [81], soma-targeting cannabinoid receptor (Cnr1) most strongly defines the gradient (inset) along with mitochondrial mt-Cytb, neurofilament Krt73,

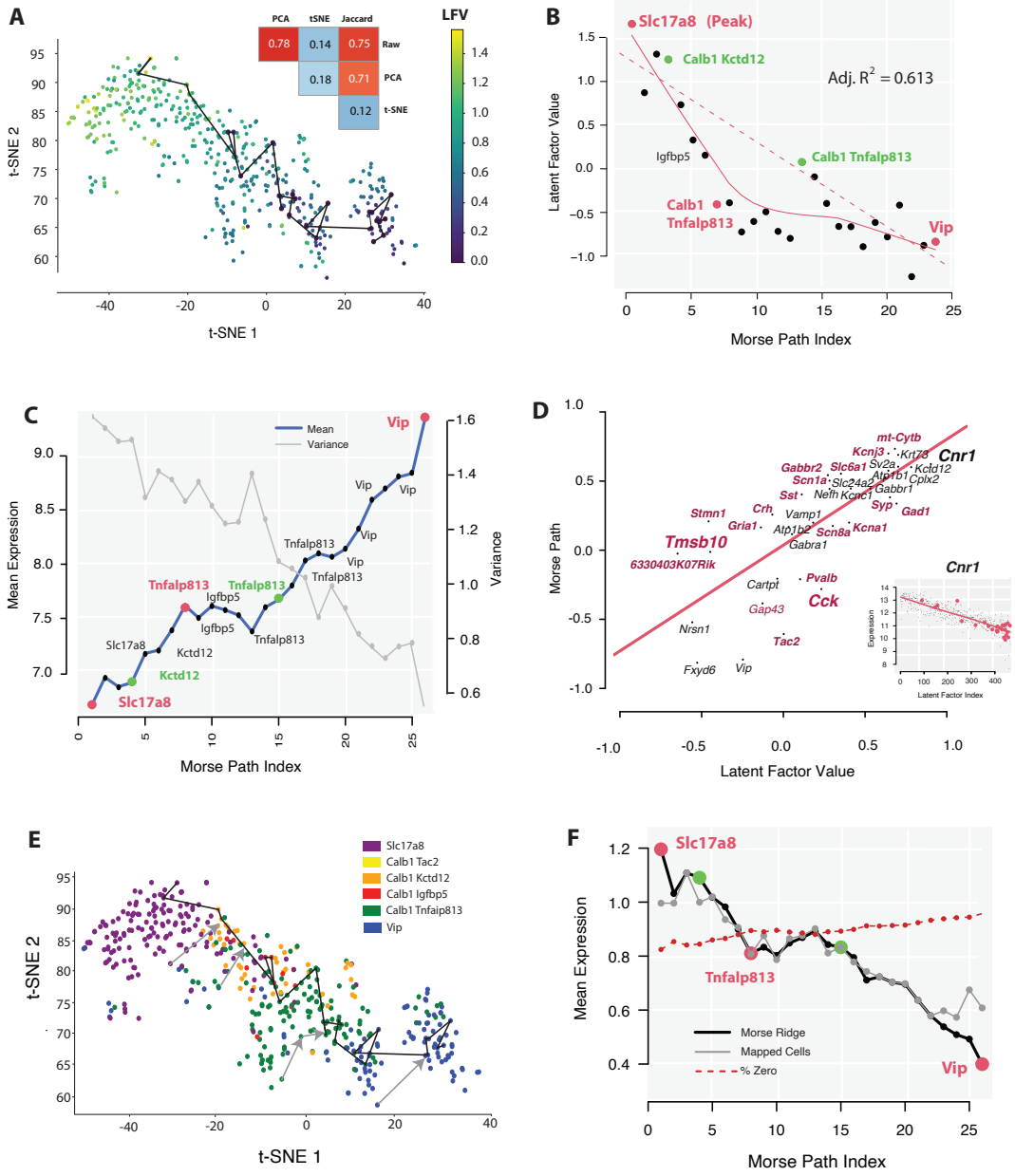


Figure 6.3. (A) 465 Cck cells from mouse CA1 hippocampus [81] represented by t-SNE. LFVs (color) show expression variation over gradient and Morse path. Inset shows mean distance correlations between points for Raw, PCA (d=100), t-SNE, and Jaccard. (B) LFV with Morse path index with peaks (red), saddles (green), linear Adj. R^2 and Lowess fit. (C) Mean non-zero gene expression (solid black) along gradient with [81] identified cell types and variance of non-zero expression (gray) (D) 36 axon targeting genes with expression along the gradient and correlation to LFV versus correlation to Morse path(Spearman 0.828, Adj. $R^2 = 0.554$). Inset shows Cnr1 gene with Morse path dots in red. (E) All cells colored by the cell type of their Morse representative. (F) Averaging all genes (gray) assigned to Morse representative agrees more with gradient (black) ($\rho = 0.951$) than LFV ($\rho = 0.816$)

and GABA-receptors Kctd12, Gabbr1. Morse path analysis affirms the hypothesis that genes associated with fast-spiking phenotype (e.g. Kcnc1, Kcna1, Scn1a, Scn8a) were amongst the genes most positively correlated with the gradient [81]. Important differences include actin processing Tmsb10, several neuropeptides including Vip, Tac2, and ubiquitously expressing Cck (Figure C.2) whose expression has more variation along the optimal gradient.

The scDMGR algorithm produces a gradient map from all cells to its closest cell on the Morse graph. Figure 6.3E colors all Cck cells by the label of its representative Morse cell using the gradient map. The distribution of reassigned cells shows limited agreement of 49.8% with Slc17a8, Vip, Tnfaip813 agreeing with the original classification (Figure C.2). and where the Calb 1 Tac 2 cluster with limited cells (n=36) does not appear at this Jaccard resolution. Indeed, ANOVA analysis of the original latent factor values (Figure C.2E) shows while Slc17a8, Vip are quite distinct types ($p < 4.82x10^{-51}$), while Calb1 Tac2 is indistinguishable from Igfbp5 (p¡ 0.16), and in the Morse framework these cells have been absorbed into other clusters. By assigning gene expression vector of every cell to its mapped Morse representative (Figure 6.3F) affirms the Morse path captures the overall decreasing gradient structure of the data ($\rho = 0.951$, $p < 8.16x10^{-14}$).

The cell types Calb 1 Tac 2 and Kctd12 are weak determinants of the Morse gradient. To investigate the identify of these cell types we lower the threshold to J=0.11 so that Slc18a7, Tac2, Calb1 Kctd12, Calb1 Igfbp5, Calb1 Tnfaip813, and Vip all occur as Morse maxima. This new graph (Figure C.3) spans the Cck dataset although it correlates weakly with the gradient. Mapping all cells to the Morse graph now improves classification to 70.5% agreeing with original type (Slc17a8 0.992, Vip 0.833, Tfnfaip813 0.618, Igfvp5 0.597, Kctd12 0.437, Tac2 0.361) indicating that while each of the 6 Cck [81] cell types are molecularly distinct not all are determinants of the observed gradient.

6.7 scDMGR for Cell Type Taxonomy

Advances in single cell and nucleus sequencing have produced a wealth of data for the study and classification of transcriptomic based cell types in the brain [156, 157, 155, 120, 3, 74]. Clustering of transcriptomic data reduces the dimensionality of the data and allows researchers to better analyze, visualize and interpret the results [132]. Many approaches have been developed to identify minimal statistically coherent groups, or transcriptomic cell types, based on hierarchical clustering, machine learning based, and other statistical methods [156, 81, 74, 144, 98, 145]. A common hierarchical approach [156, 157, 155, 120, 3] first selects anchor cells and high variance genes for a dataset, computes k nearest neighbors with respect to some metric, then performs Louvain clustering [136] on the similarity of cell groups. Clusters are then merged if insufficiently separable and the process is recursively repeated to capture the multiresolution nature of brain cell types.

As scDMGR can compare transcriptomic differences between individual cells with any annotation, it provides a natural framework for investigating the validity and coherency of a labelled taxonomy. In Figure 6.4A we have constructed a Morse graph on 57,902 cells from the anterior cingulate cortex area (ACA) [157], an area involved in both the acquisition and expression of contextual fear to predatory threat [37]. The dataset consists of 10,503 GABAergic and 47,399 glutamatergic cells. A Morse graph at persistence threshold $\delta = 0.3$ having 384 cells (160 GABAergic, 224 glutamatergic) is shown in Figure 6.4A. The Morse graph cells carry annotation from cell type clustering labels from [157], with labels of GABAergic subtypes (Sst, Sst Chodl, Pvalb, Sncg, Lamp5, Vip) and glutamatergic subtypes (L2/3 IT, L4/5 IT, L5 IT, L5 PT, L5/6 NP, L6, L6 IT, L6b) and region specific types (L4-RSP-ACA, CA2-IG-FC).

To measure the consistency of externally derived cluster labels, and therefore the consistency of those cell types with respect to the Morse embedding, we use a metric derived from scan statistics [59]. The k-LCI metric (Methods) measures the fraction of neighbors of each cell v with label l_i reachable within k edges in the graph that have the same label l_i. In practice 1-LCI is

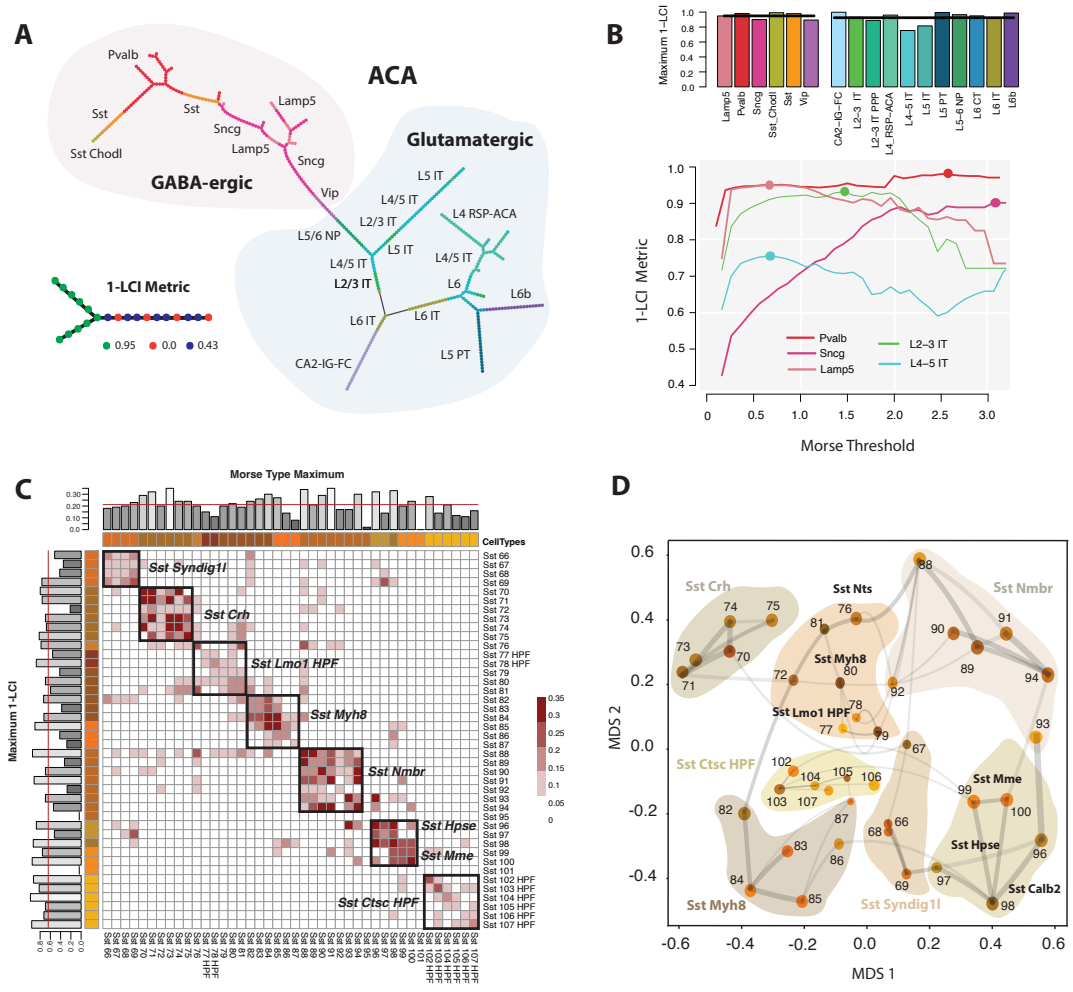


Figure 6.4. (A) Morse graph on 57,902 cells from the anterior cingulate area (ACA) [157] at persistence at threshold $\delta = 0.3$ showing separation of GABAergic, glutamatergic, and subclass cells. (B) As a measure of cell type specificity 1-LCI metric (Methods) measures fraction of the neighbors of Morse cell v that have the same label as v itself. Curves of 1-LCI by threshold shown for GABA-ergic (Pvalb, Sncg, Lamp5) and glutamatergic (L2/3 IT, L4/5 IT) (Figure C.4) Points label curve maximum, barplot is maximum 1-LCI for all subclasses. (C) A Morse Persistence Taxonomy (MPT) is derived as a matrix where each row and column represent Morse types, and where the (i,j)-th entry is the maximum persistence threshold for which a minimal connection in the Morse graph is retained between those types. Cell types are labeled by representative cluster labels from [157]. Top annotation bar is Morse Type Maximum MTM(c) the largest δ for which c occurs on M_δ as a relative maximum. Left annotation bar is maximum 1-LCI as a measure of consistency with annotated cell types. Color bar is the supertype group from [157]. Boxed types are optimal Louvain groups corresponding with supertype labels. (D) Constellation map representation of MPT with multidimensional scaling (MDS) for the coordinates for each type. The size of the vertices scales as the maximum 1-LCI score with edge thickness determined by the magnitude of off-diagonal MPT entries and representing cell type confusion.

sufficient to capture label coherency (Figure 6.4A). In Figure 6.4A GABAergic and glutamatergic labeled cells of the graph are well separated. As we specialize to subclass and cluster level types, 1-LCI stability will rise as threshold δ increases up to a maximum value beyond which there are no coherent groups with this concentration of cells (Figure 6.4B) For a well-defined cell type, increasing δ denoises neighbors in the k-NN graph until groups of maximum coherency occur. Beyond the maximum δ labeled groups are splintered and eventually disappear from the graph. Figure 6.4B shows several curves for GABAergic (Pvalb, Sncg, Lamp5) and glutamatergic (L2/3 IT, L4/5 IT) subclasses with maxima for all ACA subclasses (mean GABAergic 0.949, Glutamatergic 0.924) consistent with the uniqueness of these clusters (Figure C.4).

Another important metric is the Morse Type Maximum (MTM) which captures the strength of cell type identity in the Morse graph. To define MTM, we recall that a Morse cell type is a cell achieving a local maximum at some threshold, and construct a sequence of Morse graphs M_δ, $\delta \in [\delta_{Min}, \delta_{Max}]$, where δ_{Min} (δ_{Max}) is the smallest (largest) threshold for which a Morse cell type occurs on M_δ. We then define the Morse Type Maximum MTM(c) to be the largest δ for which c occurs on M_δ as a relative maximum. MTM(c) is a measure the of strength of a Morse cell type c, increases as the fraction of neighbors of c with similar expression increases (Figure C.5).

We use scDMGR and the metrics (MTM, 1-LCI) to investigate the structure of 42 Sst cell types brainwide identified in [157]. Computing a Morse graph on 45,467 cells over persistence thresholds $\delta \in [0, 0.35]$ we first identify the Morse types and find that all but two types, Sst 95 and Sst 101, occur as labels of Morse Types (Figure C.5). A complete summary of the Morse cell types and their relationship to the annotated taxonomy is described by the Morse Persistence Taxonomy (MPT) (Figure 6.4C), a matrix where each row and column are unique Morse types labeled by the associated taxonomy [157]. The $(i,\ j)$-th entry of MPT is the maximum persistence threshold for which a minimal connection in the Morse graph is retained between those types (Methods) and measures the proximity relationship of these types in the ambient space. Annotation bars show MTM measuring Morse cell type strength and maximum

1-LCI measure coherence of the labels from the inherited taxonomy.

The MPT describes the relationship of Morse type to the annotated taxonomy labels [157]. The result shown in Figure 6.4C corroborates 40 out of 42 clusters, with mean MTM = 0.21, range [0.0 (Sst 101), 0.35 (Sst 91)] and mean 1-LCI= 0.614, range [0.0 (Sst 95), 0.947 (Sst HPF 104)]. The types Sst 95 and Sst 101 have been removed having insufficient evidence of transcriptomic uniqueness. The MPT shows rich transcriptomic structure with coherent groups corresponding to larger supertype clusters such as Sst Crh (Sst 70-Sst 75) and Sst Nmbr (Sst 88-Sst 94). Hippocampal Sst cell types of [157] are comparatively rare types (mean) Max MT = 0.16 but transcriptomicaly highly coherent 1-LCI = 0.784. Louvain community detection [136] identifies 7 groups corresponding with supertypes [157] although there are also complex transcriptomic relationships between types indicated by off-diagonal band entries.

A constellation map representation is depicted in Figure 6.4D, obtaining a more quantitative realization of the constellation maps used in [157]. Here, we apply multidimensional scaling (MDS) to the MPT matrix to obtain coordinates for each type. The size of the vertices scale as the maximum 1-LCI score with edge thickness determined by the magnitude of off-diagonal MPT entries, representing cell type confusion. Figure 6.4D agrees with the supertype groupings colored [157] with important divergences labeled in black (e.g. Sst Calb2, Nts, Etv1). We also derive an MPT for glutamatergic types L4 IT, L4/5 IT, L5 IT, and, L5/6 IT CTX in Figure C.5 in agreement with [157].

Applying this approach, we can derive a Morse Persistence Taxonomy for all 177,594 GABAergic cells of [157] extending the approach of Figure 6.4D measuring MTM for identifiability and 1-LCI for coherency of cell types. The taxonomy of 119 Morse representative types (Figure 6.5) shows predominant diagonal structure of subclasses Lamp5, Pvalb, Sncg, Sst, Sst Chodl, and Vip. Overall mean GABAergic MTM is 0.235 (above) and 1-LCI 0.663 (left). Whole brain transcriptomic studies in the mouse have shown reproducible structure of major cell types with increasing variability among finer types [156, 155, 157, 120, 3, 74, 49] and using scDMGR corroborates this finding.

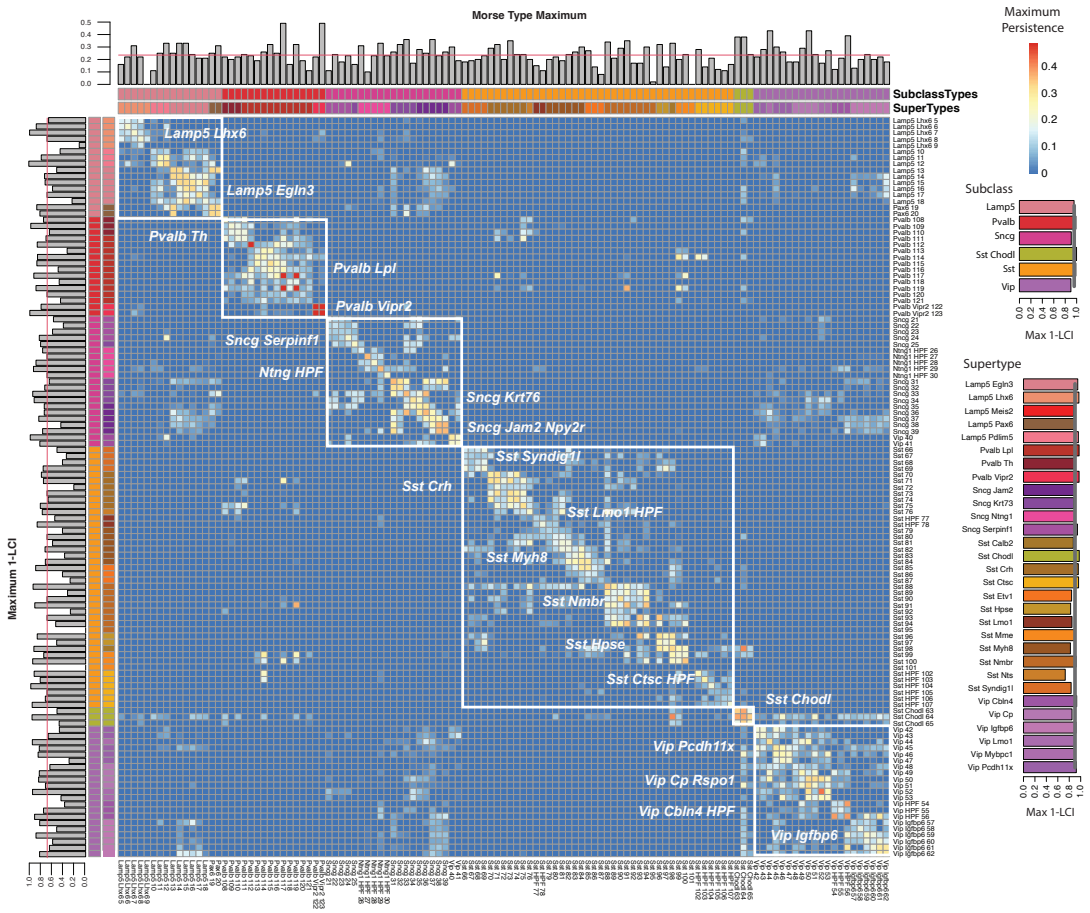


Figure 6.5. MPT for 119 identified Morse cell types of major GABAergic inhibitory classes with (Lamp5 (n=42,144), Pvalb (30,461), Sncg (13,877), Sst (45,467), Sst Chodl (1961), Vip (43,684)) with 21 of 30 supertypes shown annotated. Matrix entries are maximum persistence for which edges occur in Morse graph connecting types. Top bar shows Morse Type Maximum (MTM) (mean 0.235) and left bar 1-LCI (mean 0.663). Color annotation bars show subclass and supertype membership from [157]. Bar plots (right) show maximum 1-LCI value indicating coherency of types. The MPT shows supertype 1-LCI is much stronger than cluster level cell type identity (mean Cluster: 0.665, Supertype: 0.923, Subclass: 0.948).

An intermediate level of transcriptional type between subclass and cluster level, called supertypes, has been defined and exhibits strong reproducibility between studies [11, 101]. The coherent expression structure of 21 supertypes is confirmed in Figure 6.5 which shows 1-LCI coherency is much larger than cluster level cell type coherency (mean Cluster: 0.665, Supertype: 0.923, Subclass 0.948) (Figure C.6) and showing that supertypes are essentially as identifiable as subclass types. The GABAergic MPT taxonomy in Figure 6.5 illustrates a high degree of consistency with the hierarchical cell type classification obtained in [157] with agreeing supertypes annotated. Strong and isolated supertypes such as Pvalb Vipr2 (1-LCI = 0.866, MTM=0.355), Lamp5 Lhx6 (1-LCI=0.647, MTM= 0.182), Sst Chodl (1-LCI=0.625, MTM=0.333) are captured in the taxonomy which also describes variation at the finer cluster level.

6.8 Cell Type Loss in Alzheimer's Disease

Alzheimer's disease (AD) is the leading cause of dementia in older adults [133] and is characterized by stereotyped accumulation of proteinopathies (amyloid and tau) along a continuum [76] leading to profound cognitive decline. A recent study [11, 101, 94] used single nucleus transcriptomics and a reference taxonomy from the BRAIN Initiative [11] to study changes in middle temporal gyrus (MTG) cell types in 84 donors with varying AD pathologies. The study produced the Seattle Alzheimer's Disease Cell Atlas (SEA-AD) [11, 101, 94], which extended the BICCN MTG taxonomy to include 139 cell supertypes. Quantitative neuropathology was used to place donors along a disease pseudo-progression score (CPS), that revealed two disease phases: an early phase with a slow increase in pathology, presence of inflammatory microglia, and loss of somatostatin positive inhibitory neurons, and a later phase with exponential increase in pathology, and loss of excitatory neurons and Pvalb and Vip inhibitory neuron subtypes.

The study [11, 101, 94] found changes in Sst and Pvalb cell type distribution early in

the CPS continuum and that loss of cell type was either due to cell death or to inability to map diseased cells to an identifiable type. Mapping cells to an identifiable type requires the analysis of coherent cell type expression patterns and is a natural application of scDMGR. To directly quantify changes in gene expression for supertypes along the CPS continuum we build a Morse graph for each of 109 neuronal supertypes maintaining annotated cell labels early (CPS $\leq$ .5) or late (CPS $>$ .5). We then compute the cell type coherency measure 1-LCI at several thresholds to determine maximum separability between the early and late cells (Methods). Here, the maximin of the 1-LCI of both labels over all thresholds measures the change in gene expression of supertypes as a function of disease burden.

Figure 6.6A shows a scatterplot 1-LCI values by CPS effect size loss for all 109 GABAergic (Lamp5, Pax6, Pvalb, Sst, Vip) and glutamatergic (Chandelier, L2/3 IT, L4 IT, L5 IT, L6 IT, L5 ET, L5/6 NP, L6 CT, L6b) supertypes and colored by subclass. The plot is divided into 4 quadrants based on median values showing no linear relationship of cell type loss to overall 1-LCI changes in gene expression (Adj. R^2 = 0.02). However, when viewed at subclass level (Figure 6.6B) the (-/+) quadrant consists primarily of L2/3 IT supertypes that are both lost with disease progression and have significant gene expression changes. Similarly, the quadrant (-,-) shows supertypes lost with disease progression with less variable gene expression across CPS having only GABAergic Sst, Pvalb and Sncg cell types, likely due to cell death. A soft margin SVM (Figure 6.6C, Methods) shows significant gene expression changes by CPS between GABAergic and glutamatergic cell types (F1 = 0.862), and although glutamatergic types have more variably expressed genes, CPS effect size is not fully explainable by this observation (Figure C.7) [11, 101, 94].

Dysfunction of Sst and Pvalb interneurons and associated memory deficits has been reported in both model organisms and humans AD [88, 131, 2] and correlated with disease progression [101]. To examine gene expression changes with CPS cell type loss we focus on Sst 13 and Pvalb 12 (labeled, Figure 6.6A), two affected GABAergic supertypes with the most significant 1-LCI changes and examine Morse cell paths (lengths 15,17) in transition

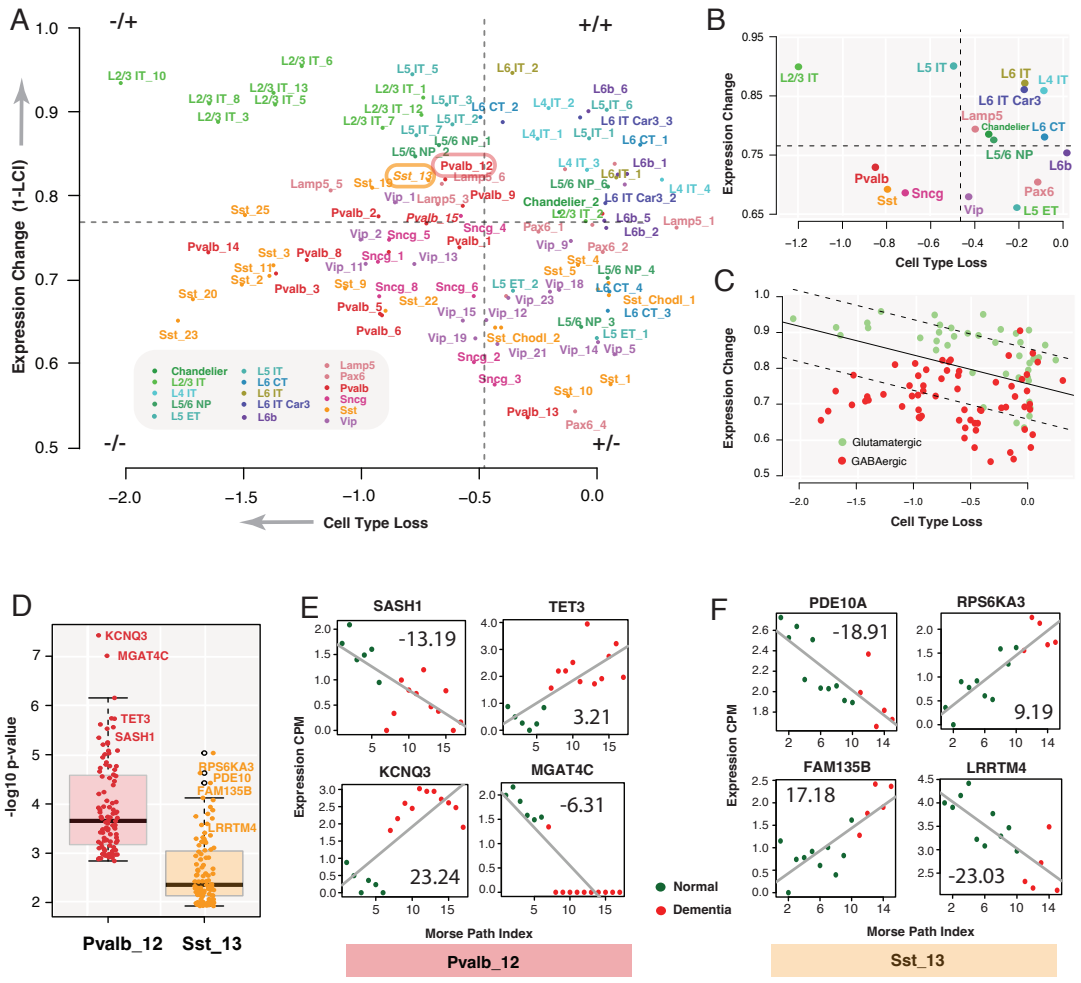


Figure 6.6. (A) Scatterplot 1-LCI values by cell type loss for all GABAergic (Lamp5, Pax6, Pvalb, Sst, Vip) and glutamatergic (Chandelier, L2/3 IT, L4 IT, L5 IT, L6 IT, L5 ET, L5/6 NP, L6 CT, L6b) supertypes, colored by subclass label. Maximum of minimum 1-LCI scores across thresholds for early and late cells (Y-axis) and the cell type loss distribution change calculated in (X-axis) for each supertype in the Sea-AD dataset [101, 94]. Quadrants indicate median delineated regions of supertype loss and quantifiable gene expression changes. Supertypes Sst 13 and Pvalb 12 are highlighted. (B) Same scatter plot colored by subclass level labels, (C) Soft margin linear SVM model separating GABAergic and Glutamatergic types with classification F1 = 0.862, (D) Box plot of p-values for 100 most differentially affected genes along supertype Morse paths. (E) Pvalb 12 expressing genes with strong differential gradient across Morse path by dementia status. SASH1, MGAT4C are decreasing across Morse gradient, TET3, KCNQ3 increasing. Inset is the differential effect size for each for this supertype found in [101], (F) Expression trajectory for genes PDE10A, RPS6KA3, FAM135B, LRRTM4 for supertype Sst 13.

from cognitively normal to dementia status. The study [101] found many genes selectively expressed in both vulnerable Sst and Pvalb supertypes, but not in unaffected supertypes from these subclasses. In Figure 6.6D we identify genes and cells that exhibit the most significant transcriptomic changes across the Morse path based on dementia status. The effect size for Pvalb 12 supertype expression changes with dementia (Cohen $D = 1.12$) is larger than for Sst 13 ($D = 0.899$) (Figure 6.6D) where the most differential genes for non-dementia/dementia genes in Pvalb 12 are enriched for neurofibrillary tangles measurement ($p < 2.59x10^{-9}$) [30].

Figure 6.6E,F illustrate the expression trajectory of several genes with strong gradient along the Morse paths and corroborating the differential effect size identified in [101]. Many of these genes with the highest differential gradient have been associated with AD and cognitive deficits [128, 126, 12]. In particular, Pvalb 12 potassium channel gene KCNQ3 is known to be required for the brain to generate accurate spatial maps and has been suggested as a target for AD [158, 55]. The role of Sst 13 expressing phosphodiesterase and their inhibitors in AD therapeutics is well-established [75, 71] through modulation of signaling pathways and physiologic effects [93], and ribosomal protein rPS6KA3 has been shown to predict amyloid burden in preclinical AD [107, 141]. The scDMGR framework provides a method for examining transitions in gene expression for cells and cell types altered in AD and by dementia status, potentially providing new avenues for identifying therapeutic targets.

6.9 Discussion

The field of single cell genomics has expanded enormously in recent years with improved detection sensitivity, multi-omic profiling of single cell/nucleus and chromatin state, and advanced data processing [105]. The complexity of these data has motivated development of advanced and high-performance algorithms for the identification of cell types and transitional states and for the annotation of these data [19]. Unlike morphological and projection data of individual neurons, the structure of omics data is fundamentally more abstract. While the ambient

space is evidently high dimensional, the precise nature of the underlying manifold is unclear [102]. Whereas a large fraction of the genome expressed in neuronal brain cells, which subsets of these genes are essential to defining cell type identity, remains an open question [105, 70].

The taxonomy and identity of brain cell types is at least partially hierarchical where at the highest level neurons and glia differentiate, neurons can be divided into excitatory and inhibitory classes, and well defined subclasses among these groups correspond to understood anatomic and to a certain extent functional properties [102, 155, 74]. Most approaches to cell type profiling in the brain have taken a multiresolution approach identifying first major class, subclass and then finest cluster level structure. However, there is uncertainty in distinguishing a cell's more permanent identity from its transient states [137]. Cell types have characteristics reflective of tissue development and organization, although there is variation even among types that may reflect cell states or transitions between less defined cell types [19, 77]. Biological and functional annotation wherever available are necessary to clarify distinctions [87]. scDMGR provides a principled way to model the observed continuous nature of cell type identity and for the analysis and characterization of cell types.

scDMGR exploits the computational approach of discrete Morse theory and persistent homology by developing a representation of the transcriptomic landscape in the input ambient space [80, 127, 44]. The underlying idea of Morse theory is to study a manifold by studying functions defined on that manifold and to examine the critical points of these functions as the places where the most significant changes occur [80, 127, 103]. The extension of Morse theory to discrete data is based on the work of Forman and colleagues [51, 80, 127, 103, 85] who develop combinatorial data structures for representing and effectively computing gradients. As an application to transcriptomic data analysis, scDMGR captures the desired characteristics of cell type identity and the relationship between types by finding a lower dimensional manifold of high-density neighborhoods of cells with common transcriptomic profile (cell types) and ridges between these peaks. The ridges are optimal gradient paths between peaks that are the most direct route between peaks.

In scDMGR we study cell type densities as measured by Jaccard index of common shared neighbors of cells with closet transcriptomic properties. We showed that scDMGR graphs are compact, faithful representation of the full dataset (Figure 6.1). The output graphs of scDMGR reveal the continuous nature of gene expression changes, in contrast to lower dimensional embeddings, which tend to show cells appearing in discrete, separated clusters [156, 155, 120]. Additionally, gene expression appears to change smoothly along the output graphs. While some genes may turn on or off transitioning between types, typically at the saddles of the scDMGR output graphs, there is continuity in gene expression change approaching the saddle. In this way scDMGR models the seemingly continuous transformations in expression are observed in anatomically or transcriptome proximal cells [159, 156]. Another useful statistical model of these transitions of state is the concept of pseudo-time, for which a number of methods have been developed [19, 60, 39]. The goal of pseudo-time modeling is to determine changes in underlying latent variables driving transition where single cell assays are susceptible to confounding as the measurements are not averaged over populations of cells. While several of these methods are suitable for cell type gradient analysis, scDMGR combines aspects of pseudo-time modeling with Morse theory methods to allow simultaneous measurement of distinguished peaks and continuous modeling of transitions.

The ability to examine the coherency of cell type labels is a powerful application of scDMGR, and k-LCI is an effective means of measuring cell type coherency in the assignment of cell type labels in a manner consistent with the underlying metric. In this way, examining the properties of a cell type on scDMGR graphs reveals some labels, particularly at the lower levels of clustering, that do not exhibit patterns consistent with desired cell type features (e.g. Figure 6.4, Sst 95 and Sst 101), and allows us to discard these as viable cell types. In further work, it will be interesting to explore cross regional cortical taxonomies and the consistency of these structures across brain regions. Additionally, the application of scDMGR is not restricted to single cell/nucleus RNA-seq and could be applied to epigenetic data such as ATAC-seq [61] and potentially providing a means of testing alignment in multiome data.

A limitation of most methods in topological data analysis is the complexity and running time of algorithms [44]. scDMGR is not an exception to this rule where the persistence computation within the scDMGR algorithm remains the largest computational bottleneck of the method. This computation has a worst-case running time that scales with the cube of the size of the input filtration, as opposed to just the size of the input dataset. This poses certain limitations on the size of data that can be run and scDMGR will not be a replacement for large scale clustering methods. The most powerful use cases of the methodology are in precise expression characteristics such as gradient analysis and detailed examination of localized cell type structure. There is ongoing work in developing sparse filtrations [16, 7] and the distributed persistence algorithms [6, 91], to make TDA more usable on large datasets, and these advances will increase viability of TDA on large scale scRNA-seq datasets. In summary, scDMGR is a novel and powerful tool for metrically faithful investigation of detailed and local transcriptomic structure as well as the corroboration of cell type identities derived in the many available studies.

6.10 Reprint

Chapter 6, in full, is a reprint of the material to be submitted in Discrete Morse Graph Reconstruction for High-Dimensional Transcriptomic Data. Magee, Lucas; Gala, Rohan; Travaglini, Kyle; Sümbül, Uygar; Wang, Yusu; Hawrylycz, Mike. 2024. The dissertation author was the primary investigator and author of this paper.

Chapter 7

Conclusion and Future Work

7.1 Summary

In this dissertation, we developed several topological methodologies for extracting true underlying graph structure from data with applications to neuroscientific datasets in mind. The starting point for this dissertation was the already developed DM graph reconstruction algorithm, which would extract the unstable 1-manifolds of density functions as the underlying graph structure. Chapter 2 covers this algorithm and the underlying mathematics required to intuitively understand the computation and its output.

First, we focus on 2D and 3D imaging applications. In Chapter 3, we applied the already developed DM graph reconstruction algorithm to both 2D and 3D mouse brain imaging datasets. In particular, neuronal process segmentation and skeletonization methodologies rely on machine learning techniques that only use local information and do not respect the connectivity of neuronal branches. We applied DM graph reconstruction outputs to the images. For the 2D images, we created gray-scale masks of graph reconstructions, and applied used these masks as input for a Siamese neural network to achieve state-of-the-art performance for neuronal process segmentation by observable connectivity improvements segmentation outputs. Additionally, we used the highly persistence vertices on the graphs to achieve state-of-the-art bouton detection. For 3D images, DM graph reconstruction outputs were used as the starting point for a single neuron reconstruction pipeline that achieved state-of-the-art performance, outperforming several

machine learning methodologies. We then developed full brain skeletonization pipelines for both 2D and 3D mouse brain imaging datasets, an important step toward reconstructing each neuron in an entire brain.

Next, in Chapter 4, we explore the decomposition of a graph representing all neuronal processes into individual neurons. We generalize this problem to decomposing a density graph into the minimum number of monotone trees. We prove this and several variations of the problem to be NP-Hard, and provide several approximation algorithms, highlighted by a 3-approximation algorithm for density cactus graphs.

Then, we shifted focus to extracting graph structure from high-dimensional PCDs. We focus on further developing DM graph reconstruction to handle such datasets in Chapter 5. Specifically, we started by changing how we view the DM graph reconstruction algorithm from a density-based perspective to a filtration-based perspective. We then generalize DM graph reconstruction to take any arbitrary filtration as input and prove its output graph is lex-optimal. We then combine the generalized algorithm with the sparsified weighted Rips filtration of [16] for effective graph extraction from high-dimensional PCDs. We demonstrate the effectiveness and efficiency of our algorithm with several experimental results.

Finally, we conclude by applying DM graph reconstruction to scRNA-seq datasets in Chapter 6. The high-dimensional PCD algorithm developed in Chapter 5 does not produce meaningful graph structure for scRNA-seq datasets. Thus, we start by combining the lower-star filtration with respect to Jaccard index of the k-NN graph with the generalized algorithm of Chapter 5 to extract meaningful graph structure from k-NN graphs. We experimentally demonstrate that the Morse graph is a compact, meaningful representation of the full dataset and is more faithful to the high-dimensional raw gene space than lower-dimensional embeddings. We then use DM graph reconstruction output graphs to define cell type identity, analyzed gene expression gradients between cell types, build cell type taxonomies, and quantify gene expression changes over Alzheimer's disease progression.

7.2 Future work

Time complexity is the biggest limitation of topological methods that use persistent homology. The persistence computation has cubic worst-case running time with respect to the size of the input filtration. Several developments that were used in this dissertation, such as sparsified filtration or distributed algorithms, have already made TDA more accessible. Any further advancements in these lines of work will only continue to make the methodologies we developed in this dissertation more and more viable to a wider range of applications. Even scRNA-seq datasets, which continue to get larger both in number of samples and dimensions due to rapid technological developments, could immediately benefit from such advancements in TDA. Although this work focuses on neuroscientific applications, we believe the methodologies can be applied as is to similar problems in different domains.

While the work of monotone tree decomposition of a desnity graph in Chapter 4 is strictly theoretical. It would be interesting to develop an efficient and effective algorithm that is useful for recovering different neural processes within a neuronal bundle or other applications that may arise. While we also provided some variations of the general monotone tree decomposition problem to be NP-Hard, different applications may give rise to new formulations of the problem.

Additionally, the scope of this dissertation is to extract meaningful graph structure from datasets using topological methods. However, topology also holds promise for higher dimensional structures beyond persistent homology. Although, persistent homology can also capture higher dimensional homology features, whereas in this work we are only concerned with zero and one dimensional homology features. Developments in topological methods that can extract higher dimensional structures from data will make topological data analysis an option for data that has true underlying structure that is more complicated than just a graph.

Appendix A

Chapter 4 Appendix

A.1 SC-1 Approximation Bound [86]

Theorem A.1.1. *There exists a constant $c > 0$ such that approximating the SC-1 problem within a factor of $c\frac{log(n)}{log(log(n))}$, where n is the number of elements in the universe, in deterministic polynomial time is possible only if $NP \subset DTIME(2^{n^{1-\varepsilon}})$ where ε is any positive constant less than $\frac{1}{2}$.*

A.2 Complexity

In this section, we prove Theorem 4.3.4, which states many variations of the minimum M-Tree set problem are also NP-Complete.

A.2.1 Proof of Theorem 4.3.4: CM-Tree Sets

Firstly, the problem is clearly in NP. We will follow the same reduction from SC-1 as seen in the proof of Theorem 4.3.1 to prove NP-Hardness, with one additional step.

The first direction we follow identical arguments to create an M-Tree Set of appropriate size, but do not yet have a CM-Tree Set. In particular, consider the M-Tree Set at the end of the proof - the only possible edges missing are edges (a_{S_j}, b_e) such that S_j is in the Set Cover and contains e, but another set S_i containing e is in the Set Cover and $f_i(b_e) = 1$. We will modify the M-Tree Set to ensure every such edge that is left out is included in a component. Consider

an element e that is in n sets in the set cover, where $n > 1$. Let (T_i, f_i) be the monotone tree in the M-Tree Set such that $f_i(b_e) = 1$. Set $f_i(b_e) = \frac{1}{n}$. Additionally, for each set S_j in the set cover such that $b_e \in S_j$, add (a_{S_j}, b_e) to monotone tree (T_j, f_j) and set $f_j(b_e) = \frac{1}{n}$. Then, for each set S_k such that S_k is not in the set cover and $b_e \in S_k$, add (b_e, a_{S_k}) to each (T_j, f_j) and set $f_j(a_{s_k}) = \frac{1}{n}$. We still have an M-Tree Set, as each component clearly remains a monotone tree, and the sum of function values at each node is equal to what it was prior to the modification. Once this modification is performed for every element contained within multiple sets in the set cover, we have an M-Tree set with every edge in the input domain included in at least one monotone tree. The second direction is identical to the previous proof.

A.2.2 Proof of Theorem 4.3.4: SM-Tree Sets

Firstly, the problem is clearly in NP. In order to prove this decision problem is NP-Hard - we first show that a specific instance of Vertex Cover - where for the given input graph $G(V,E)$, for any two verts $u, v \in V$, there is at most one vert $w \in V$ that is adjacent to both u and v - is NP-Complete. This will limit the number of connected components in the intersection between two components to be at most one in our reduction to the M-Tree problem.

Lemma A.2.1. *Given a graph $G(V,E)$ such that for any two verts $u, v \in V$, there is at most one vert $w \in V$ that is adjacent to both u and v and an integer k, determining whether or not there exists a vertex cover of size $\leq k$ is NP-Complete.*

Proof. This is a specific instance of Vertex Cover and is clearly in NP. To show it is in NP-Hard use the same reduction from 3-SAT to regular Vertex Cover as seen in [57], but use a "restricted" version of 3-SAT where we can assume the following:

- A clause has 3 unique literals
- A clause cannot have a literal and its negation

These assumptions are safe because we can transform any 3-SAT instance that has any such

clauses to an equivalent 3-SAT instance with no such clauses in polynomial time. Thus this restricted version of 3SAT is also NP-Complete.

Consider the graph created in the reduction of this restricted 3-SAT to Vertex Cover. Two literal vertices will have no shared neighbors by design. Any literal vertex and vertex in a clause will only have 1 neighbor if the literal vertex is also in the clause or the clause vertex is the negation of the literal vertex. Two vertices in clauses will only have a single shared neighbor - the literal vertex if they are the same literal (and are thus in different clauses), or the final clause vertex if they are in the same clause. Thus, the reduction is also a reduction to our special Vertex Cover problem, and follows the exact same proof. □

We will now reduce the special instance of Vertex Cover to the SM-Tree decision problem to prove NP-Hardness. We follow a very similar reduction as seen in the proof of Theorem 4.3.1. We construct a bipartite graph $G(V = A \cup B, E)$, with nodes in A corresponding to nodes in the instance of Vertex Cover, and nodes in B representing edges in the instance of Vertex Cover. We then build density function f on the nodes of V, setting $f(a_v) =$ degree of node v in Vertex Cover instance for each node $a_v \in A$, and $f(b_e) = 1$ for each edge e in the Vertex Cover instance. Prove both directions the exact same way as shown in the proof of Theorem 4.3.1, but note that for the first direction, because of the restriction on our input graph, any two components of the decomposition can have at most a single vertex in their intersection.

A.2.3 Proof of Theorem 4.3.4: FM-Tree Sets

Firstly, the problem is clearly in NP. To show the problem is in NP-Hard, we follow the exact same reduction from SC-1 as seen in proof of Theorem 4.3.1. For the first direction - we note that the M-Tree Set we have constructed is also an SM-Tree Set - as each set S_i in the set cover is a root of a component (T_i, f_i) such that $f_i(a_{S_i}) = f(a_{S_i})$. The second direction remains the same - though the argument that if a node b_e is not the maximum of any monotone tree then one of its neighbors must be is slightly different. In this case, the neighbor must be a maximum

in the same monotone tree it is a parent of b_e in - not being so would contradict that the set is in fact a FM-Tree Set.

A.3 Naive Approximation Algorithm

Given a density graph (G, f), a natural upper bound for $|\mathsf{minSMset}(G, f)|$ is the number of relative maxima on the density graph. Algorithm 9 constructs monotone trees rooted at each relative maxima on the input density graph. Starting at a root, depth-first search (DFS) is performed to reach every node that can be reached via a non-increasing path from the root. DFS stops once no nodes remain or all remaining nodes are not reachable from the root via a non-increasing path. We call this graph traversal algorithm *monotone DFS*. Perform monotone DFS from each relative maxima to build an M-Tree Set. The M-Tree Set will have size at most one less than the number of relative maxima more than the size of a minimum M-Tree Set. Figure A.1 shows an example output of Algorithm 9.

Algorithm 9: naive-algo((G(V, E), f))

Input: A density graph $(G(V,E), f)$
Output: An (S)M-Tree set of $(G(V,E), f)$
(Step 1) Compute set M containing the relative maxima of f on G.
(Step 2) For each relative maxima $m_i \in M$, perform monotone DFS to build a component (T_i, f_i)
(Step 3) Return all T_i, f_i

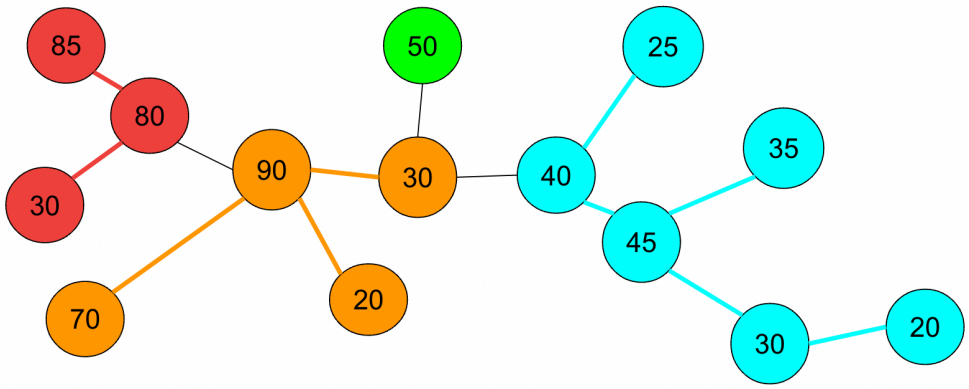


Figure A.1. M-Tree Set with of a density tree with 4 monotone trees computed by Algorithm 9.

Appendix B

Chapter 5 Appendix

B.1 Further Study of Alternative Approaches

B.1.1 Different input triangulations for baseline

Our main experiments compare the quality of our DM-PCD method to the baseline algorithm. baseline takes an input triangulation, for which we chose the Rips complex at a fixed radius. We tested other input triangulations to highlight that the baseline approach fails regardless of the input triangulation. Results are shown in Figure B.1. Even using sparse weighted Rips complex at a fixed radius large enough to capture the larger feature with less noise compared to a regular Rips complex, the points forming the smaller feature are connected by nearly a clique. Using this triangulation with any valid density function as input for the baseline algorithm results in the smaller feature being lost. It is also shown that using the weighted Rips complex without sparsification results in a similar triangulation and final output.

B.1.2 Different input filtrations for generalized algorithm

Our DM-PCD algorithm takes a sparse weighted Rips filtration of a point cloud dataset. However, we generalized the discrete Morse graph reconstruction algorithm to take an arbitrary filtration. To highlight the utility of the sparse weighted Rips filtration, we run the generalized discrete Morse graph reconstruction algorithm with both the regular Rips filtration and the regular weighted Rips filtration. Results are shown in Figure B.2. Using the regular Rips filtration,

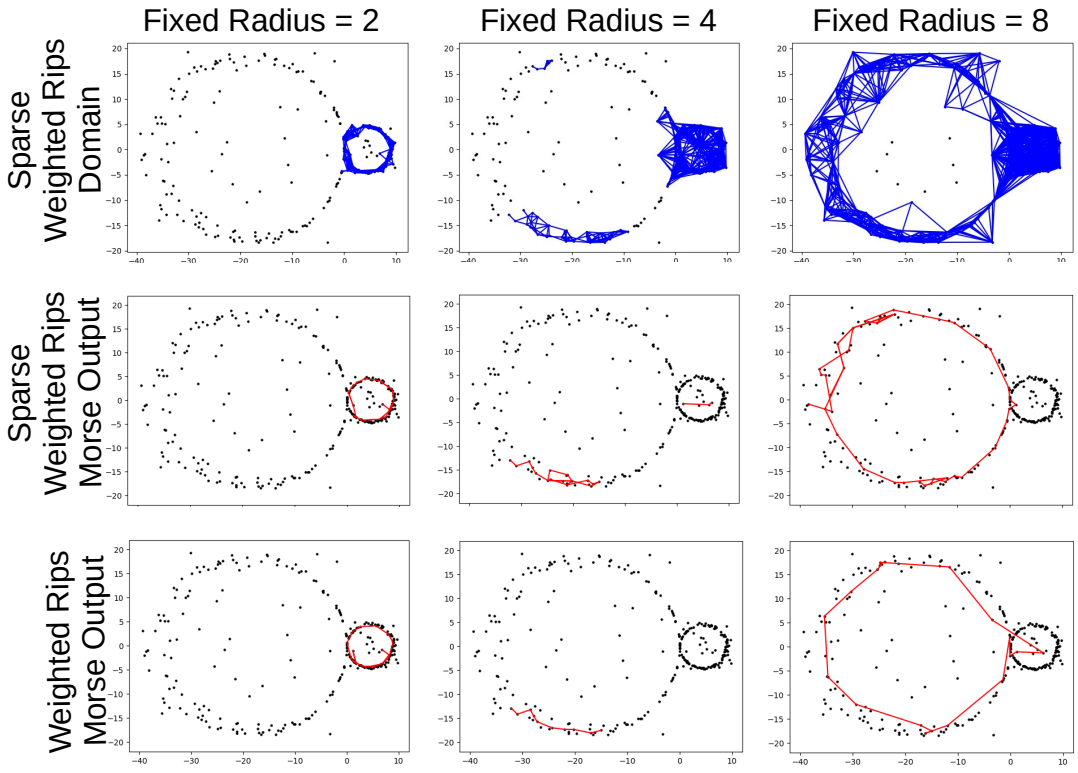


Figure B.1. Sparse weighted Rips complex at fixed radii (first row) with results of baseline using sparse weighted Rips complex (second row) and full weighted Rips complex (third row). Fixed radii values of 2 (first column), 4 (second column), and 8 (third column) are shown.

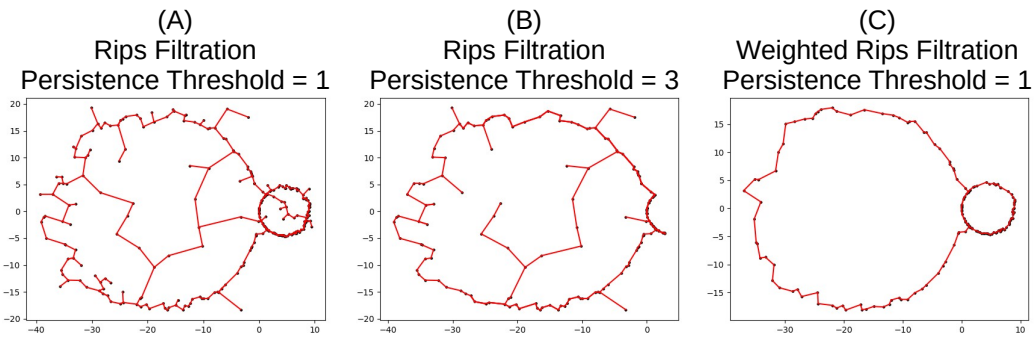


Figure B.2. Results of using the generalized discrete Morse algorithm with the regular Rips filtration as input. At a lower persistence threshold (A), both features are captured with additional noise. At a higher persistence threshold (B), the smaller feature is lost while some noise remains. Using the weighted Rips filtration as input (C), the algorithm is able to recover both features with no noise.

the output captures the two features with a lot of additional noise. Trying to use persistence thresholding to remove the noise will remove the smaller feature before all noise is removed. The regular weighted Rips filtration is able to perfectly capture both features, similarly to using the sparse weighted Rips filtration. However, because the persistence computation of the filtration is a bottleneck, the sparse filtration is a superior option for our DM-PCD algorithm.

B.1.3 Dimensionality reduction of noisy data

For noisy datasets, such as the image patches dataset, dimensionality reduction techniques alone fail to reveal meaningful structure. Results of such techniques are shown in Figure B.3. Chapter 5 shows our DM-PCD algorithm extracts a clear three circle structure that is known to be the true underlying structure of the image patches data. However, PCA, tSNE, and UMAP projections of the image patches dataset reveal no meaningful structure (Figure B.3 (A) - (C)). This is because these methods do not look to preserve metric relations. In particular, tSNE attempts to cluster data and UMAP attempts to preserve continuous structure. For cleaner data, such as Coil-20 (Figure B.3 (D)), we see that UMAP is able to capture structure. However, even applying PCA and UMAP (Figure B.3 (E) and (F)) to the much cleaner $X(15,30)$ subset of image patches, we see that UMAP is still unable to capture the known three circle structure of

the data. Running the baseline and Mapper approaches on the PCA reduced image patches data also fails to extract the correct structure. Results are shown in Figure B.4. Running baseline with a persistence threshold $\delta = 2$ results in a graph where three circles appear visible (Figure B.4 (B)). However, the topology is incorrect, as all circles intersect twice (the first Betti number is equal to 7). Raising the persistence threshold to 4 (Figure B.4 (C)) results in an output with the correct first Betti number equal to 5, but we have clearly lost the 3 circles. Mapper fails on the PCA reduced data and the output is very similar to the output on the original data (Figure B.4 (D)). This example highlights a general problem with performing dimensionality reduction then performing graph reconstruction - one needs to reduce to an appropriate dimension. Clearly it would not be possible to extract the correct graph structure from the images patches dataset if it were first reduced to 2 dimensions. It turns out that reducing to 3 dimensions is also too much, as we are unable to capture the proper (dis)connections between circles. Not having to reduce dimension, and more so not needing to know the limit for dimensionality reduction, is a huge advantage to our method.

B.2 More Details on Experiments

Comparison of methods Our experiments compare the quality of outputs and computational efficiency of our DM-PCD method with the baseline algorithm and the state-of-the-art ReebRecon algorithm. We also compare the quality of outputs to those of the Mapper algorithm. We do not include Mapper running times in our comparisons of computational efficiency because it is significantly faster than the other algorithms.

The ReebRecon algorithm has two outputs - a contracted output, which contains only non-degree two nodes, and an augmented output, which contains edges going through every possible node in the domain. While the contracted outputs are useful for examining the topology of the output, they do a poor job of preserving the underlying geometry of the output. On the other hand, the augmented outputs are very noisy because every node is included. For this reason,

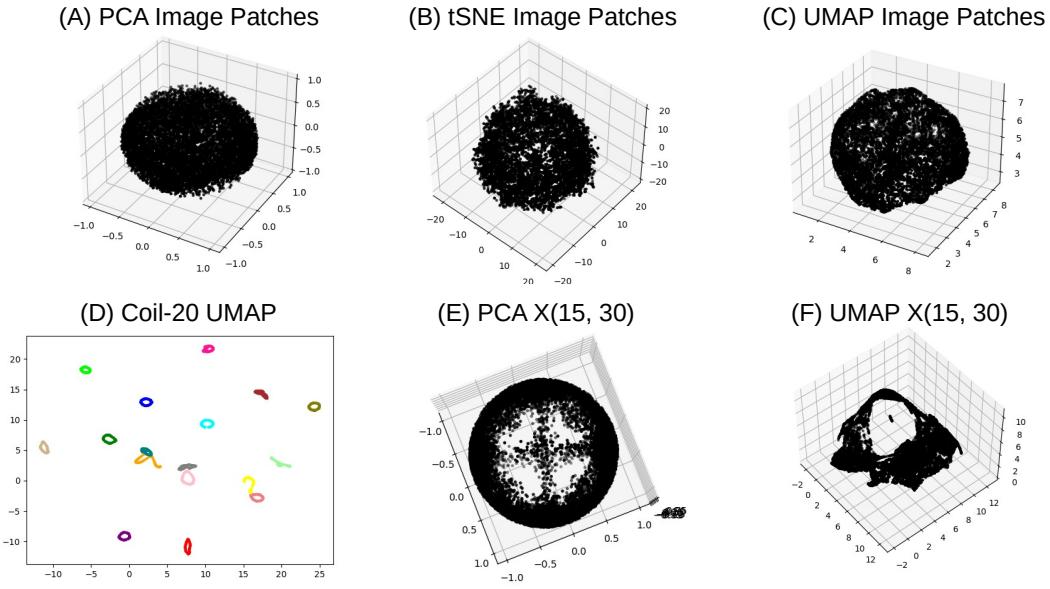


Figure B.3. Outputs of various dimensionality reduction techniques (PCA, tSNE, and UMAP) performed on the 10,000 image patch subset ((A) - (C)), Coil-20 (D), and $X(15, 30)$ ((E) and (F)).

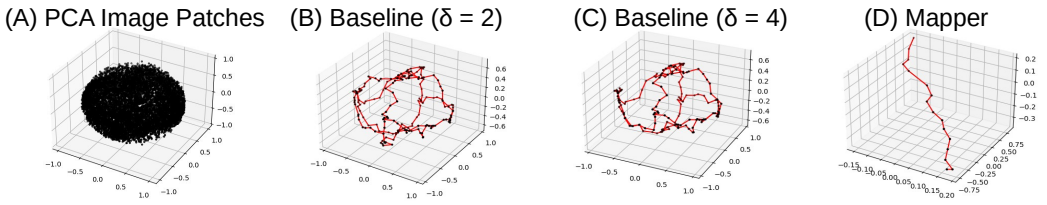


Figure B.4. PCA reduction of image patches dataset (A) and outputs of baseline with persistence thresholds 2 and 4 (B and C), and Mapper (base point filter) (D) on the PCA reduced image patches dataset.

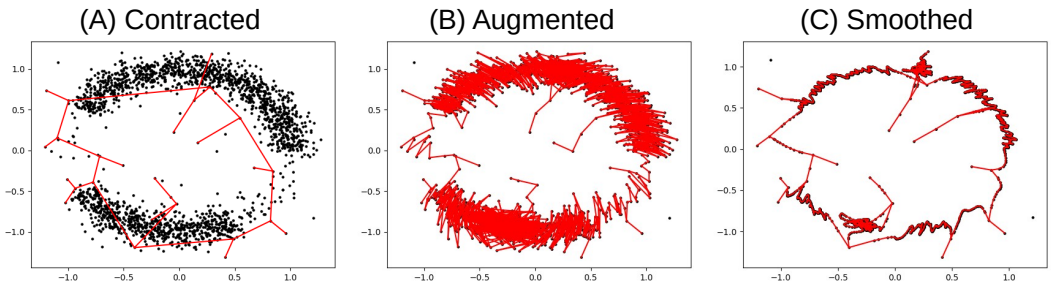


Figure B.5. ReebRecon outputs on one circle dataset. The contracted output (A) and the augmented output (B) of the ReebRecon algorithm with $r = .25$. The contracted output is useful in examining the topology of the output, but does a poor job of preserving the geometry of the underlying skeleton. The augmented output better preserves the geometry but contains a lot of noise. (C) is a smoothed augmented output with less noise.

the authors of the ReebRecon algorithm smooth outputs. We smooth the augmented outputs by subsampling the arcs (non-degree two paths), and then perform standard iterative smoothing on the remaining vertices. An example is shown in Figure B.5. Unless otherwise noted, the ReebRecon results displayed in figures are the smoothed augmented outputs. Ultimately, the quality of the output is now dependant on the smoothing, and we note that different smoothing techniques may result in better quality outputs. However, the topology of the outputs is often incorrect, and in such cases no smoothing can make the output "correct".

The Mapper algorithm traditionally outputs a simplicial complex and was not developed to explicitly extract underlying graph structures from data. For all of our experiments, we limit the Mapper output to be a graph (1-dimensional simplicial complex). Each node in a graph outputted by Mapper represents a cluster computed within the algorithm. We assign the coordinates of a node to be the average coordinates of the cluster it represents.

In our time comparisons, ReebRecon is much slower than both DM-PCD and baseline. While we are using an old implementation from 2011 that may not be optimized, it is known that ReebRecon is theoretically faster than both DM-PCD and baseline, which have persistence computation as a bottleneck. DM-PCD tends to be more efficient than baseline, as the sparsification in our algorithm builds a filtration that is linear in size with respect to the number of points, whereas the regular Rips complex used in baseline results in a filtration of size $O(n^3)$,

and r values large enough to capture the underlying skeleton will have much bigger filtrations.

B.2.1 One Circle dataset

Chapter 5 shows that our DM-PCD algorithm is able to successfully capture the circle, and both the baseline and ReebRecon capture the circle with $r = .25$. However, the quality of output for both baseline and ReebRecon is heavily dependent on the value of r - more specifically the corresponding rips$^r(P)$ complex. Shown in the first row of Figure B.6 is the rips$^r(P)$ complex for r values of .1 (A), .2 (B), .25 (C), and 1.05 (D). The second and third rows contain results of baseline and ReebRecon. All ReebRecon outputs are smoothed with no subsampling, a neighborhood radius of 2 neighbors, and 5 iterations - except for (D), where the output is a spanning tree and smoothing does not improve output quality. With an r value too small (.1), the underlying skeleton is not contained in rips$^r(P)$, and neither method will be able to produce a desirable output. It is not enough to select an r value that results in the complex containing the underlying skeleton. For $r = .2$, the circle is captured by the rips$^r(P)$ complex, but so is an additional spurious loop. Neither baseline or ReebRecon can produce an output not containing the spurious loop. For $r = 1.05$, there are no additional spurious loops in the rips$^r(P)$ complex, but ReebRecon produces a spanning tree and baseline, while producing a single loop, loses the geometry of the underlying skeleton.

For this dataset, $r = .25$ was an appropriate selection for both baseline and ReebRecon. However, as shown in Table B.1, the size of the rips$^r(P)$ complex is much bigger than the sparsified weighted Rips complex used in our DM-PCD algorithm. Complex size is particularly costly for our DM-PCD method and the baseline method because of the persistence computation. While baseline was able to produce a reasonable output at $r = .25$, it took significantly more time than our DM-PCD algorithm.

While smoothing certainly decreases the noise in the ReebRecon output, the output quality is still worse than that of both DM-PCD and baseline. We comment that a different smoothing method may result in a better quality output.

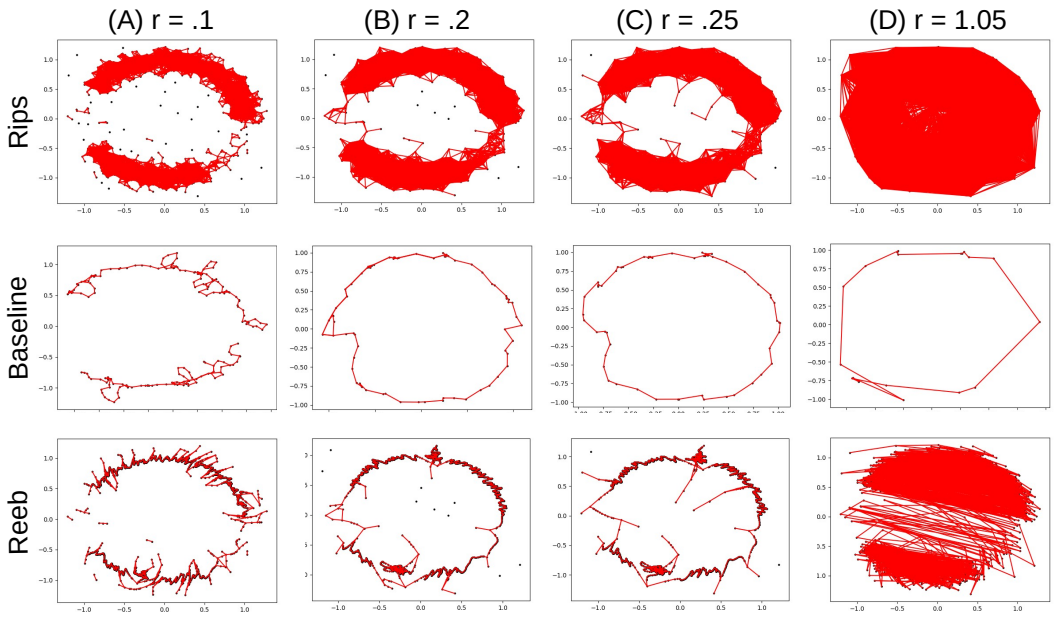


Figure B.6. One circle data - rips$^r(P)$ complex (first row), baseline outputs (second row), and ReebRecon outputs (third row). (A) $r = .1$ - rips$^r(P)$ complex fails to capture the circle, resulting in both methods failing to capture the circle. (B) $r = .2$ - rips$^r(P)$ complex now contains the circle, but also contains a spurious loop. Both the baseline output, which was generated with persistence threshold $\delta = \infty$, and the ReebRecon output must contain this spurious loop (C) $r = .25$ - rips$^r(P)$ complex now contains the circle without any additional spurious loops. Both the baseline output ($\delta = \infty$) and the ReebRecon output capture the loop. (D) $r = 1.05$ - rips$^r(P)$ complex still contains the circle, as well as many more simplices. As a result, the baseline output has lost its nice geometry, with long edges going through high density regions, and the ReebRecon output is a spanning tree.

Additionally, Chapter 5 shows that the Mapper approach is also able to successfully capture the circle. We show the Mapper results with a variety of filter functions in Figure B.7. The graph Laplacian filter and the distance to base point filter are able to capture the circle, while the eccentricity filter and the Gaussian density filter are unable to capture the true underlying structure of the data. The heat maps of the filter functions shown in the first row of Figure B.7 provide intuition on why each filters is either successfully or unsuccessfully used to extract the underlying structure with Mapper. These two filter functions will be the top choices for most of the remaining datasets.

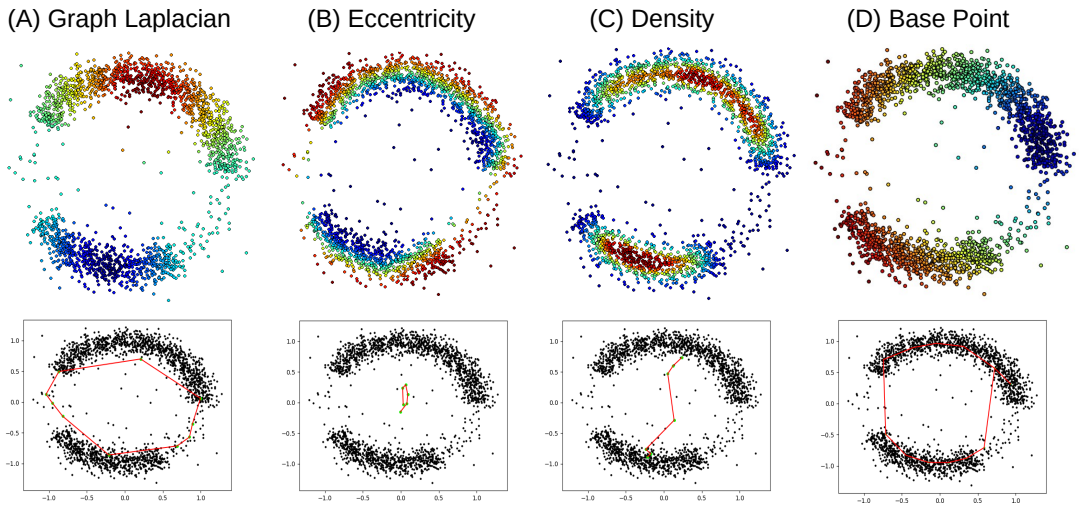


Figure B.7. One circle data - Filter function values (first row) and corresponding Mapper outputs (second row) with filter functions - (A) graph Laplacian filter ($k = 15$), (B) eccentricity filter, (C) Gaussian density filter, (D) distance to base point filter.

Table B.1. One circle dataset: Comparison of radius used, # simplices, and running time of DM-PCD, baseline, and ReebRecon. Our algorithm has radius ∞ as we run on the full sparse DTM-Rips filtration.

Method	**Radius**	**# Simplices**	**Time (seconds)**
Our Method	∞	368276	2.6
Baseline	.05	33368	.1
Reeb Graph	.05	33368	.03
Baseline	.10	356925	1.1
Reeb Graph	.10	356925	2.49
Baseline	.15	1490149	5.4
Reeb Graph	.15	1490149	21.05
Baseline	.2	3869507	13.0
Reeb Graph	.2	3869507	76.46
Baseline	.25	7708243	44.8
Reeb Graph	.25	7708243	221.25
Baseline	.5	43392850	231.1
Reeb Graph	.5	43392850	3445.10

B.2.2 Two Circle dataset

Chapter 5 shows that our DM-PCD algorithm is able to successfully capture both circles, while both baseline and ReebRecon failed to capture both circles. Again, this is because both methods are heavily dependent on the input triangulation (the $\text{rips}^r(P)$ complex). This complex at various values of r is shown in the first row of Figure B.8, while the corresponding baseline and ReebRecon outputs are shown in the second and third rows respectively. All ReebRecon outputs are smoothed with no subsampling, a neighborhood radius of 2 neighbors, and 10 iterations. Neither result can contain the larger circle if the input triangulation itself does not contain the larger circle, so we increase values of r until the complex contains the larger circle. $r = 1$ is too small to capture even the smaller circle. At $r = 1.5$, the complex does contain the smaller circle, and both baseline and ReebRecon are able to successfully extract the loop. However, at $r = 2$ and $r = 3$, the complex still does not contain the larger circle, and more noise around the smaller circle is added to the outputs. Finally, at $r = 4$, the larger circle is contained within the complex. However there are two issues. Firstly, there is a spurious loop in the complex along the larger circle, so while $r = 4$ is able to capture the larger circle, it is still not an appropriate value of r. We would need to try to find a new r value that better captures the data if not for the second issue - the smaller circle is lost in both outputs - meaning an appropriate value of r does not exist for either method. We can see that in the $\text{rips}^r(P)$ complex at $r = 4$, the points forming the smaller circle now nearly form a clique, which results in both baseline and ReebRecon outputs losing the smaller circle. We conclude that there is no value for r that will result in either method capturing both circles. Running time and simplicial complex size comparisons are shown in Table B.2. For radius $r = 4$, we see that the number of simplices used in both baseline and ReebRecon is nearly double that of the filtration used by DM-PCD. As a result, the running times of baseline and ReebRecon are longer than that of DM-PCD.

Additionally, Chapter 5 shows that Mapper was able to successfully capture both features of the two circle dataset. Further results for different filter functions are shown in Figure B.9.

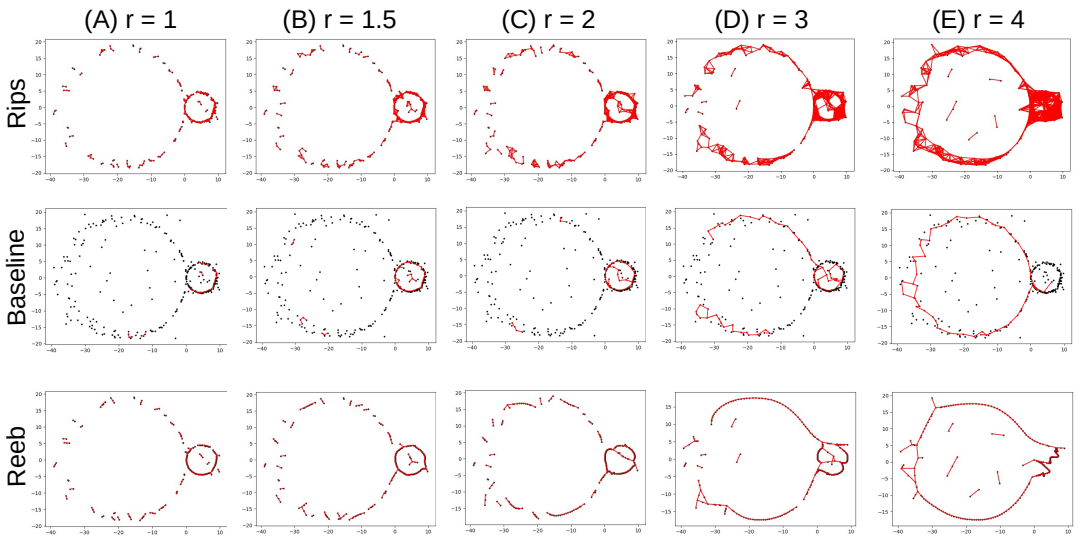


Figure B.8. The rips$^r(P)$ complex (first row), baseline outputs (second row), and ReebRecon outputs (third row). All baseline outputs were generated using persistence threshold zero, meaning that no simplification occurred. (A) $r = 1$ - rips$^r(P)$ complex fails to capture either circle, resulting in both methods failing to capture either circle. (B) $r = 1.5$ - rips$^r(P)$ complex now contains the smaller circle but not the larger circle. Both outputs successfully capture the smaller circle, but fail to capture the larger circle. (C) $r = 2$ - rips$^r(P)$ complex now connects the noise inside of the smaller circle to the smaller circle, while still not containing the larger circle. The outputs now capture the smaller circle and some noise inside of the circle, and still fail to capture the larger circle. (D) $r = 3$ - rips$^r(P)$ complex still does not contain the larger circle, and contains many edges cutting across the smaller circle. The ReebRecon output captures the smaller circle with more noise, while the baseline output has begun to lose the smaller circle. Both outputs fail to capture the larger circle. (E) $r = 4$ - rips$^r(P)$ complex now contains the larger circle, as well as a spurious loop, and the points forming the smaller circle now nearly form a clique. The outputs capture the larger circle, but contain a spurious loop, and the smaller circle is completely lost.

Similarly to the results of the one circle dataset, Mapper was able to successfully capture both features when using either the graph Laplacian filter or distance to base point filter. Looking at the heat map for the eccentricity filter, we see that it would also appear to be an acceptable choice for this particular dataset. The output captures the larger feature and is unable to capture the smaller feature. The density filter once again fails to extract any meaningful structure from the dataset.

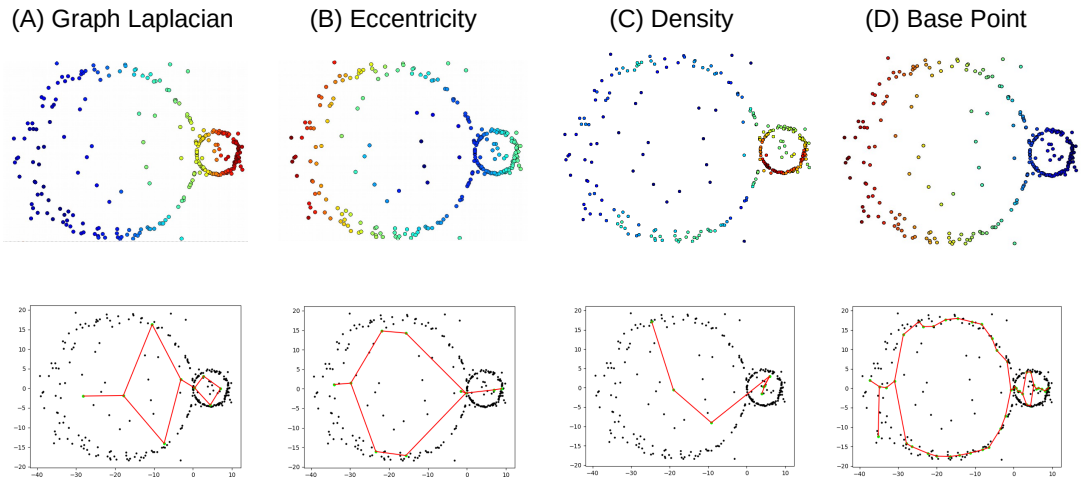


Figure B.9. Two circle dataset: filter function values (first row) and corresponding Mapper outputs (second row) using (A) graph Laplacian filter ($k = 15$), (B) eccentricity filter, (C) Gaussian density filter, (D) distance to base point filter.

Table B.2. Two circle dataset: Comparison of running time of DM-PCD, baseline, and ReebRecon. Our algorithm has radius ∞ as we run on the full sparse DTM-Rips filtration.

Method	**Radius**	**# Simplices**	**Time (seconds)**
Our Method	∞	19497	.06
Baseline	1	2182	.002
Reeb Graph	1	2182	.01
Baseline	2	8350	.013
Reeb Graph	2	8350	.02
Baseline	3	20005	.045
Reeb Graph	3	20005	.05
Baseline	4	41349	.13
Reeb Graph	4	41349	.25

B.2.3 Image patches dataset

Chapter 5 shows that our DM-PCD algorithm is able to successfully extract the "three-circle model" from a random 10,000 point subset of the image patches dataset from [22], while baseline, ReebRecon, and Mapper methods are unable to do so. We run baseline with $\text{rips}^{.75}(P)$ as the input complex. We tried several r values less than .75, as well as $r = .8$. For r values less than .75, there were many spurious loops that could not be removed with persistence thresholding. For $r = .8$, the desired three-circles are not completely recovered even with no persistence thresholding. Results at various persistence thresholds are shown in Figure B.10. Although the output does contain the three circles we wish to extract, it is also made up of several additional loops. Raising the persistence threshold to 5 removes some of the additional loops, but raising the persistence threshold to 10 removes part of the desired three circle model without removing the remaining additional loops. In fact, raising the persistence threshold to ∞, we see that some of these incorrect loops are a product of the input triangulation, and it is not possible to achieve a desired output from baseline with $r = .75$. While it may still be possible for a "good" r value to exist, it is extremely expensive to compute persistence on triangulations with this many simplices.

Running time and simplicial complex size comparisons are shown in Table B.3. For radius $r = .75$, we see that the number of simplices used in baseline is over 50,000,000 greater than the number of simplices in the filtration used by DM-PCD. Although DM-PCD takes longer to compute persistence even with a smaller filtration, the DM-PCD filtration has an implied $r = \infty$, and that any sizable increase to $r = .75$ for baseline will result in a significant increase in running time. We note that for all values of r, the number of simplices used in baseline would be the same number of simplices used by ReebRecon.

Additionally, Chapter 5 shows that Mapper was unable to capture the true underlying structure of the image patches dataset. Further results for different filter functions are shown in Figure B.11. Gaussian density and eccentricity filters fail, as seen in previous datasets. However, unlike the previous dataset, the graph Laplacian and distance to base point filters also fail

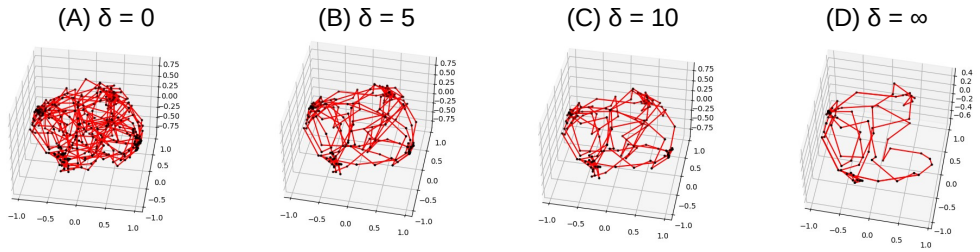


Figure B.10. Image Patches: Outputs of baseline with $\text{rips}^r(P)$ complex ($r = .75$) as the input triangulation at various persistence thresholds. (A) $\delta = 0$ - With no thresholding of critical edges, the output captures the three circles that we expect to, as well as additional loops. (B) $\delta = 5$ - Raising the persistence threshold allows for some of the additional loops to be removed while keeping the three circles we expect. (C) $\delta = 10$ - Further raising the persistence threshold results in losing part of the horizontal circle while keeping extra loops. (D) $\delta = \text{inf}$ - Removing all critical edges except those with infinity persistence removes more of the desired three circle output and keeps loops not part of the desired output.

to capture the underlying structure. This data is simply too noisy for Mapper to extract the underlying structure.

Finally, while our algorithm is deterministic, this dataset is generated from a random 10K point subset. In an attempt to quantify the error, we generated 10 different random subsets to apply our method to. On all 10 datasets, our method extracts the 3 circles correctly. To quantify error, we computed the distance between two output graphs G_i, G_j by calculating the average distance between each node in one graph to its nearest node in the other graph, and normalizing this distance by the diameter of the full 50K point dataset. The result was 0.036 average error.

Table B.3. Image Patches dataset: Comparison of running time of DM-PCD and baseline. Our algorithm has radius ∞ as we run on the full sparse DTM-Rips filtration.

Method	**Radius**	**# Simplices**	**Time (seconds)**
Our Method	∞	209397755	16089.4
Baseline	.25	77261	.15
Baseline	.75	263787145	1485.42

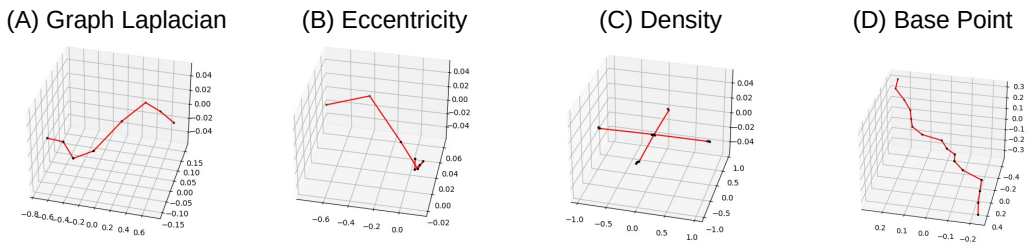


Figure B.11. Image patches: Outputs of Mapper using different filter functions - (A) graph Laplacian filter ($k = 15$), (B) eccentricity filter, (C) Gaussian density filter, (D), distance to base point filter. The dataset is too noisy and none of the filters result in Mapper outputting a graph representative of the true underlying structure.

B.2.4 Traffic flow dataset

We also test on point clouds derived from traffic flow data [21]. We extract two datasets: the time-series of traffic flow at detector #409529 from time-range 10/1/2017 to 10/14/2017 and from time-range 11/19/2017 to 12/2/2017 (which includes Thanksgiving). Each time-series is mapped to a point cloud dataset in $\mathbb{R}^6$ via time-delay embedding.

Given a time series $f : t- > \mathbb{R}$ and a parameter τ, the lift defined by $\phi(t) = (f(t), f(t+\tau), ..., f(t+M\tau))$ is called a time delay embedding. For each traffic flow function, we create a PCD using a time delay embedding with $M = 5$ and $\tau = 50$. The two dimensional projections of these PCDs are shown in Figure 5.5 (B). The first function's time delay embedding projection appears to be a single loop, while the second appears to have an inner loop and an outer loop.

Chapter 5 shows the results of our DM-PCD algorithm with $k = 30$ on both time series datasets. So far in our experiments, a default value of $k = 15$ has been used. By the nature of time delay embeddings, which may create clumps of points close together, different values of k can produce markedly different results. Shown in Figure B.12 are results of DM-PCD on the two datasets with k values of 15, 30, and 40. For the first dataset (10/1/2017 - 10/14/2017), a single loop is captured with all values of k. For the second dataset (11/19/2017 - 12/2/2017), changing the value of k results in more drastic changes in the output. The persistence thresholds for the outputs are 8.25 ($k = 15$), 12.5 ($k = 30$), and 12.84 ($k = 40$). In all cases, if the persistence

threshold were raised enough to further threshold the output, a portion of the outer loop would be lost. We note that our output must be connected, so the desired result is two loops with a single connection. The output with $k = 15$ contains many extra connections, while the output with $k = 30$ contains a single extra connection. With $k = 40$, the desired output is achieved.

Also shown in Chapter 5, baseline is able to successfully capture the single loop of the first time series dataset. Results for baseline on the second time series dataset are shown in Figure B.13. The persistence thresholds for the outputs are 8 ($k = 15$), 5 ($k = 30$), and 3 ($k = 40$). Just like the results for DM-PCD in Figure B.12, if the persistence thresholds were raised enough to further threshold the output, a portion of the outer loop would be lost, making the output of DM-PCD superior.

Running time and simplicial complex size comparisons for 10/1/2017 - 10/14/2017 and 11/19/2017 - 12/2/2017 traffic flows are shown in Table B.4 and B.5 respectively. Note that the baseline results shown in Chapter 5 use $r = 90$ and $r = 75$ respectively. For the first dataset, we see that the number of simplices used by the baseline with $r = 90$ is more than five times greater than the number of simplices used in DM-PCD. This results in longer running time for baseline. For the second dataset, the number of simplices used by the baseline with $r = 75$ is a little less than three times greater than the number of simplices used in DM-PCD. While the running time remained shorter for baseline in this particular instance, we note that an increase in the value of r can add a significant amount of simplices and push the running time to be longer than the DM-PCD running time. We again note that for all values of r, the number of simplices used in baseline would be the same number of simplices used by ReebRecon.

Additionally, Chapter 5 shows that Mapper was able to extract the structure behind traffic flow from 10/1/2017 to 10/14/2017, but was unable to do so for traffic flow from 11/19/2017 to 12/2/2017. Results of Mapper on both datasets using a variety of filter functions is shown in Figure B.14. Again, using the graph Laplacian and distance to base point filters allowed Mapper to extract the single loop structure behind the first dataset. However, Mapper is unable to extract the two loop structure behind the second dataset with these filters, along with eccentricity and

Gaussian density filters. In contrast, DM-PCD was able to get the true underlying structure behind both datasets.

Table B.4. Traffic (10/1/2017 - 10/14/2017) dataset: comparison of running time of DM-PCD and baseline. Our algorithm has radius ∞ as we run on the full sparse DTM-Rips filtration.

Method	Radius	# Simplices	Time (seconds)
Our Method	∞	6,879,338	98.6
Baseline	75	14,646,522	54.7
Baseline	90	35,543,784	123.903

Table B.5. Traffic (11/19/2017 - 12/2/2017) dataset: comparison of running time of DM-PCD and baseline. Our algorithm has radius ∞ as we run on the full sparse DTM-Rips filtration.

Method	Radius	# Simplices	Time (seconds)
Our Method	∞	5,720,309	106.1
Baseline	60	5,157,336	16.8
Baseline	75	15,659,797	57.7

B.2.5 Coil-20

Similarly to the two circle example, both the baseline and ReebRecon approaches are unable to capture all coils because the coils have varying scales in the original space. A concrete example is shown in Figure B.15, where Objects 1 and 17 cannot be captured at the same scale. We also applied Mapper to Coil-17 using a base point filter. While Mapper can extract the structure of individual objects quite well, the method also struggles to capture all coils. Focusing on Objects 1 and 17 again, we see that by changing the epsilon parameter of the density clustering scheme we use inside of Mapper, we are able to capture the structure of either Object 1 or Object 17, but not both (results shown in Figure B.16). While it may be possible that a different clustering scheme (or different covering and filter function combinations) could lead to a Mapper configuration that can capture both objects, finding parameters able to capture all coils would be difficult.

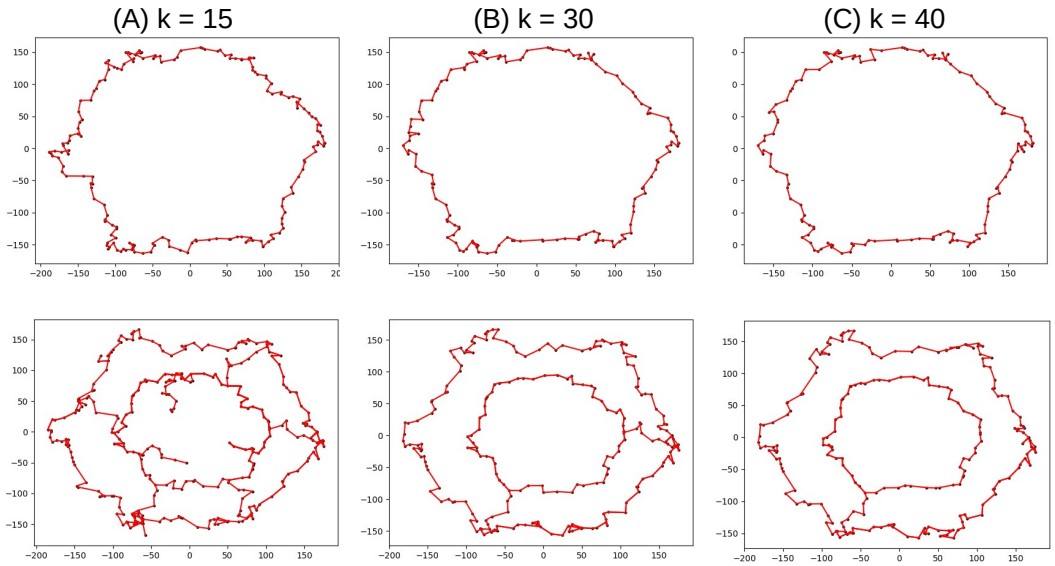


Figure B.12. Traffic flow: Outputs of DM-PCD on both datasets (10/1/2017 - 10/14/2017 top row, 11/19/2017 - 12/2/2017 bottom row) with different values of k. (A) $k = 15$ - Single loop captured in first dataset, two loops captured with extra connections in second dataset. (B) $k = 30$ - Single loop captured in first dataset, two loops captured with an extra connection in second dataset. (C) $k = 40$ - Single loop captured in first dataset, two loops captured with a single connection in second dataset.

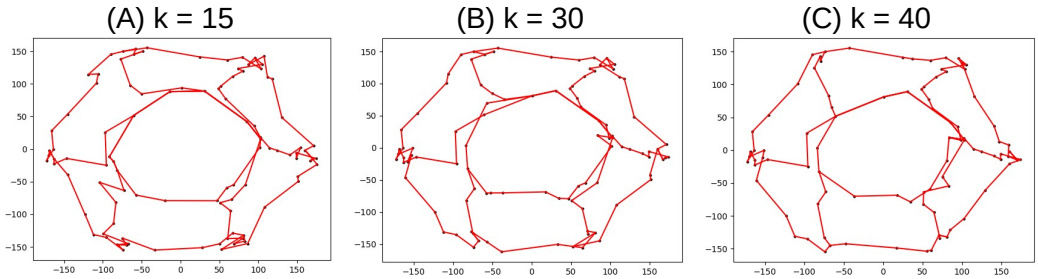


Figure B.13. Traffic flow: Outputs of baseline method on the second traffic time series dataset (11/19/2017 - 12/2/2017) at different values of k - (A) $k = 15$, (B) $k = 30$, (C) $k = 40$. In all cases, any further simplification will lose the outer loop before removing any connections with the inner loop.

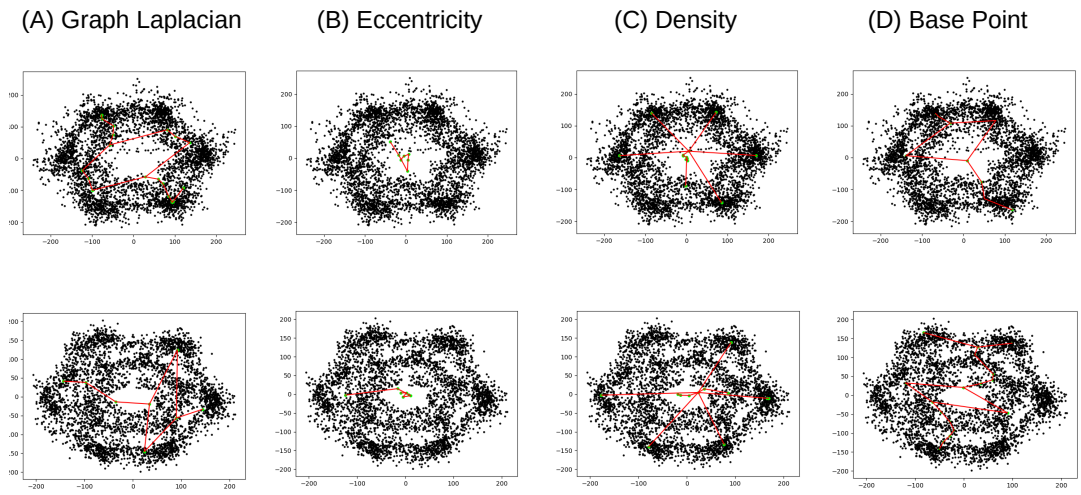


Figure B.14. Traffic flow: Outputs of Mapper on both datasets using different filter functions - (A) graph Laplacian filter ($k = 15$), (B) eccentricity filter, (C) Gaussian density filter, (D), distance to base point filter. The structure of traffic flow from 10/1/2017 to 10/14/2017 (first row) is captured by Mapper using the graph Laplacian filter function and the distance to base point filter function, while the structure of traffic flow from 11/19/2017 to 12/2/2017 (second row) is not captured by Mapper using any of the filter functions.

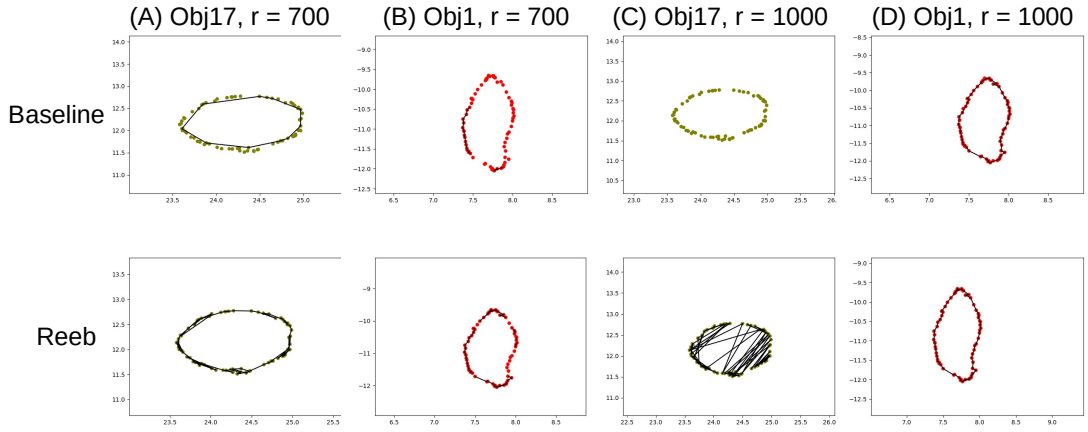


Figure B.15. Coil: Objects 1 and 17 with baseline and ReebRecon outputs. (A) Object 17, $r = 700$ - Object 17 is captured by both methods. (B) Object 1, $r = 700$ - Object 1 is not captured by either method. (C) Object 17, $r = 1000$ - Object 17 is not captured by either method. (D) Object 1, $r = 1000$ - Object 1 is captured by both methods.

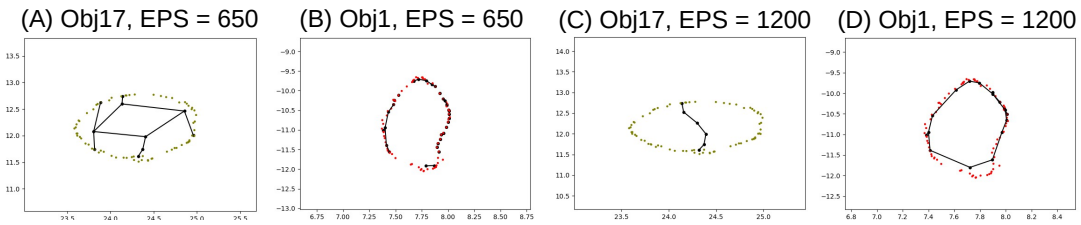


Figure B.16. Coil: Objects 1 and 17 with Mapper outputs generated using a base point filter function. (A) Object 17, $EPS = 650$ - structure of Object 17 is captured. (B) Object 1, $EPS = 650$ - structure of Object 1 is not captured. (C) Object 17, $EPS = 1200$ - structure of Object 17 is not captured. (D) Object 17, $EPS = 1200$ - structure of Object 1 is captured.

Appendix C

Chapter 6 Appendix

C.1 Supplementary Methods

C.1.1 Wasserstein Distance

The Wasserstein distance, a special case of optimal transport, is widely used to compare two probability distributions defined on the same space. In Chapter 6 we use the ℓ_1-Wasserstein distance, also known as the *earth-mover distance* in fields such as computer vision. Intuitively, given two distributions ρ and μ over $\mathbb{R}^d$, we can view them as two piles of earth. Our goal is to transport the mass from ρ to match μ while minimizing the total cost, where the cost is defined as the sum of the mass moved, weighted by the distance traveled. The *Wasserstein distance* $d_W(\rho,\mu)$ is defined as this minimal transport cost. See [142] for more precise definitions and details of the Wasserstein distance and related concepts.

The Wasserstein distance can be used to compare two point sets P and Q from $\mathbb{R}^d$, as each point set can be considered as a discrete measure. In particular, given a set of points $P = \{p_1,\ldots,p_n\}$, it induces an empirical distribution $\mu_P := \frac{1}{n}\sum_{i\in[n]}\delta_{p_i}$, where δ_x is the Dirac Delta function at x. We can then compare the two point sets P and Q by the Wasserstein distance between these induced empirical measures μ_P and μ_Q, that is, $d_W(P,Q) := d_W(\mu_P,\mu_Q)$. Furthermore, if P is a set of points sampled uniformly randomly from a well-behaved continuous measure μ, then it is $d_W(\mu_P,\mu)$ tends to 0 almost surely as the number of points in P goes to infinity.

In Chapter 6, we use the Wasserstein distance to compare the distributions of cells (which are point sets in the high dimensional space of RNA expression data). We use the implementation from the Python Optimal Transport library [50] to compute the Wasserstein distance.

C.1.2 W-L Graph Distance

Chapter 6 uses the Morse graph as a succinct summary of input data (a collection of cells, which can also be viewed as a point cloud in a certain high dimensional space). To compare the key structure of two collections of cells P and Q, we can also compare their Morse graph G_P and G_Q. Furthermore, note that each Morse graph is an *attributed graph* in the sense that each graph node $v \in V(G)$ is associated with a node feature represented by a point $x_v \in \mathbb{X}$ in some metric space $(\mathbb{X}, d_X)$: For example x_v could be the gene expression of the cell corresponding to the graph node v, represented as a point in $\mathbb{X} = \mathbb{R}^d$. In this case, the associated metric is just $d_X =$ the Euclidean distance in $\mathbb{X} = \mathbb{R}^d$. In our experiments, the node feature of each Morse graph cell (node) will be the cell's gene expression.

To compare these Morse graphs, we need a meaningful distance measure for attributed graphs which can also be computed efficiently. Unfortunately, common graph distances are often NP-hard to compute and even to approximate within a constant factor, e.g., the graph edit distance. Furthermore, they usually only focus on graph combinatorial structure and ignore node features. We use the so-called *Weisfeiler-Lehman (WL) distance* originally proposed in [31], which is an optimal transport-inspired metric for graph data with node features, combined with the classical Weisfeiler-Lehman graph isomorphism test [149]. We use the *differentiable variant* of the WL-distance as proposed in [15], as this version also allows the computation of objects such as the (Fréchet) mean of a set of graphs, if needed.

C.2 Extended Data Figures

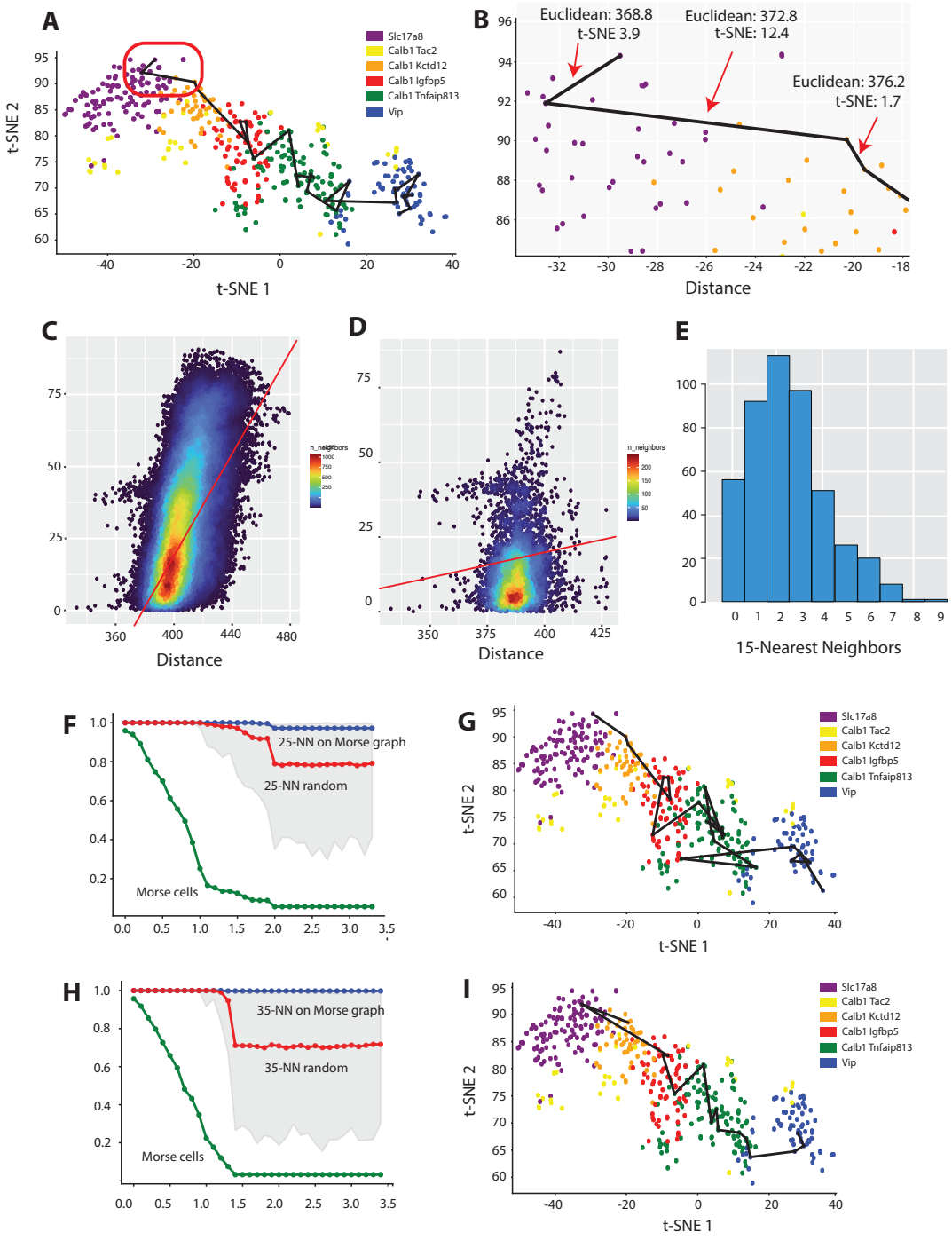


Figure C.1. (A) Cck cells colored by type and the raw gene space Morse graph embedded in tSNE by [81]. (B) Zoom-in on 3 edges from Morse graph. (C) Heatmap of distance in raw gene space vs in the embedding between (C) all cells ($\rho = 0.6$) and (D) 15-NN in raw gene space ($\rho = 0.1$). (E) Raw gene space 15-NN preserved in embedding for each cell. (F) Morse graph size and coverage compared to random and (G) embedding for $k = 25$. (H,I) Similar for $k = 35$.

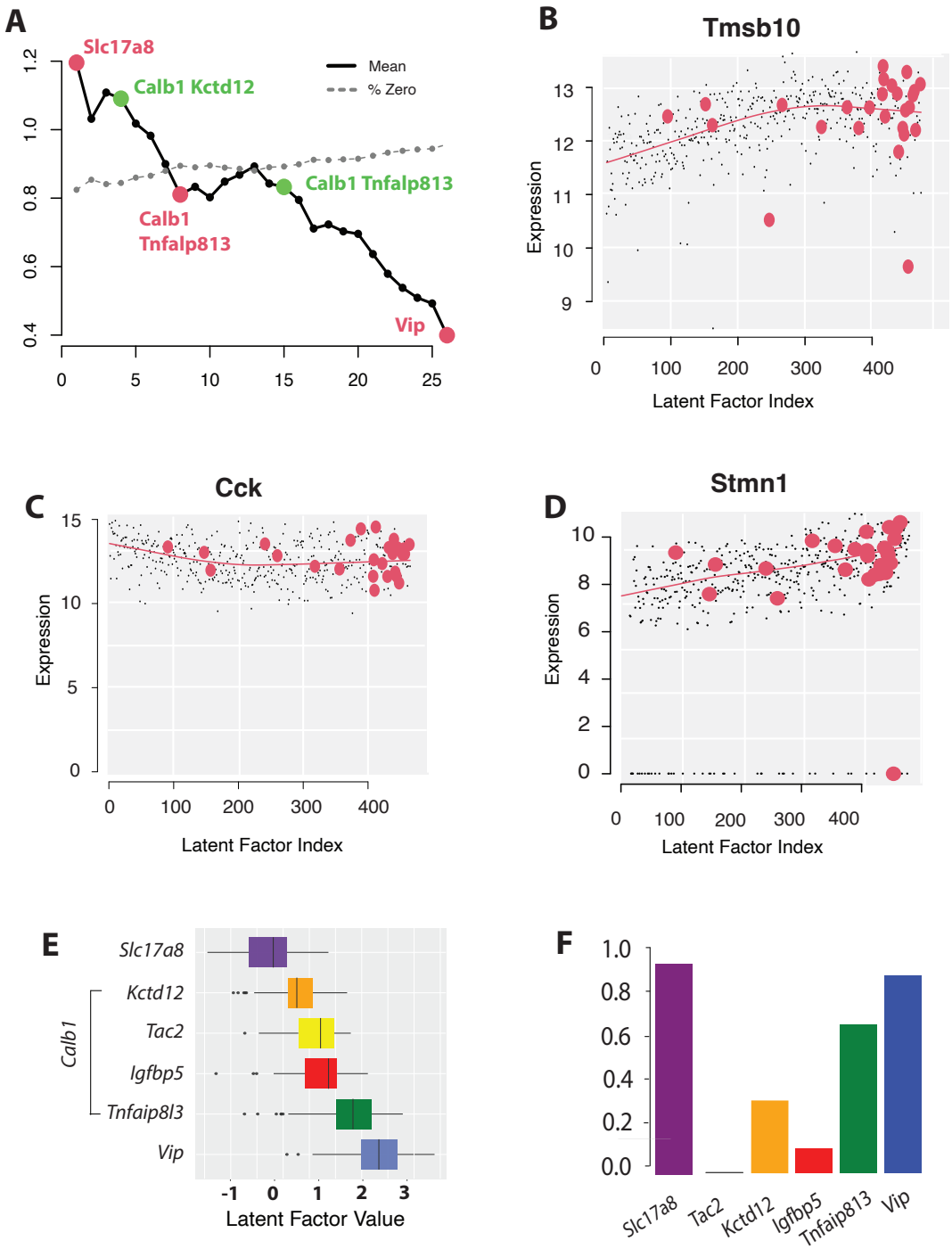


Figure C.2. (A) Mean expression along the Morse path gradient showing peak and saddle cells labeled. (B,C,D) Trajectory Lowess fit of expression along LFV ordered cells with Morse cells highlighted in red for (B) Tmsb10 (C) Cck and (D) Stmn1 genes. (E) Distribution of LFVs for major cell types. (F) Percentage of [81] cell types assigned to the same label in Morse gradient.

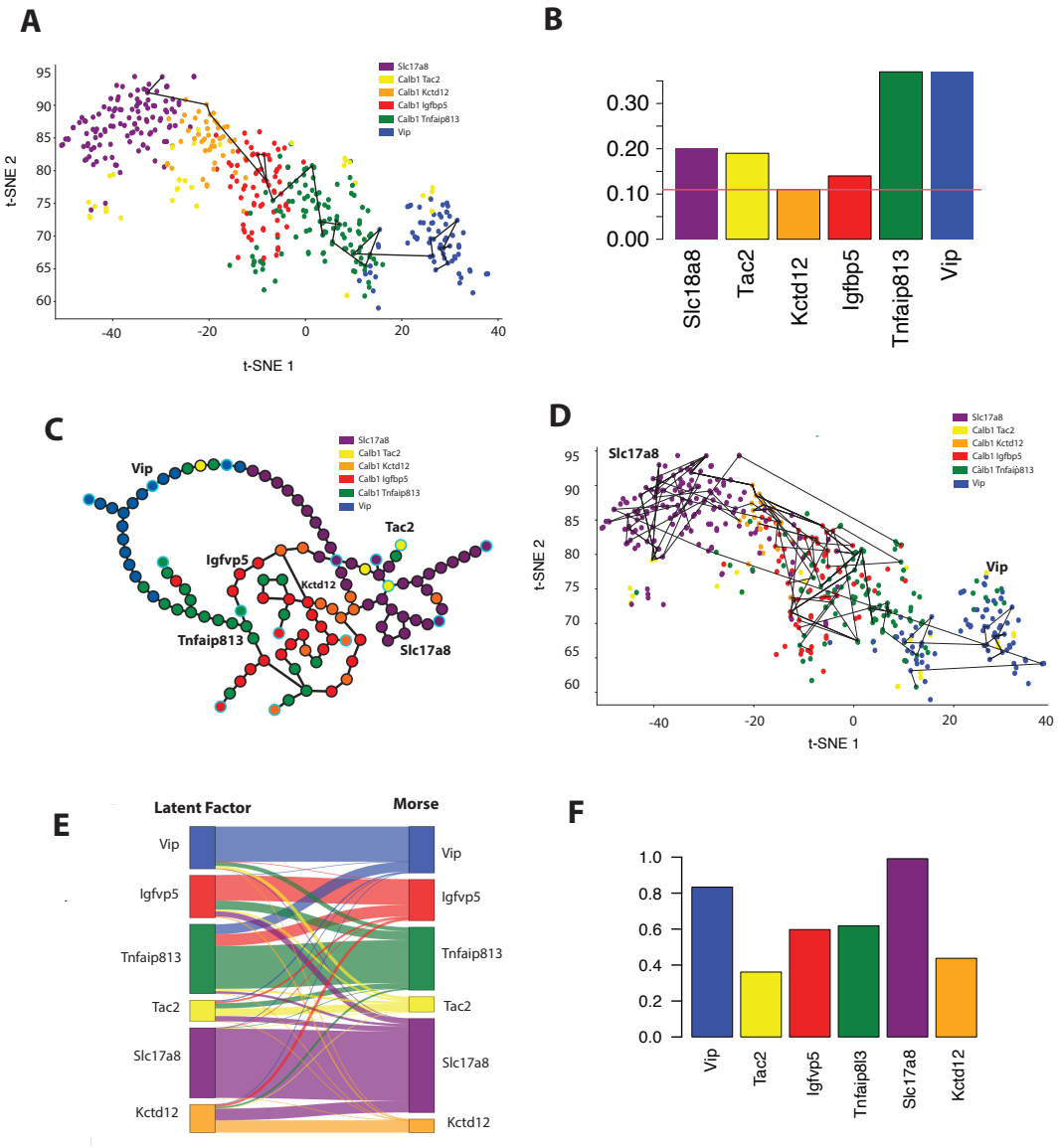


Figure C.3. (A) [81] uses a modified t-SNE to visualize the locations of the cells appropriate for data following a negative binomial distribution. 465 cells from the Cck Cxcl14 class together with labeled cell types Slc17a8, Calb1 Tac2, Calb1 Kctd12, Calb1 Igfbp5, Calb1 Tnfaib813, and Vip and Morse path with L=26 cells. (B) Maximum Jaccard threshold in k-NN graph for which each [81] labeled type occurs. Minimum is $\delta = 0.11$ for type Calb1 Kctd12. (C) Morse graph for the $\delta = 0.11$ threshold with labeled type from [81]. Maximal Morse types are shown as cyan circled. (D) t-SNE embedded with Morse graph shown and cells colored by gradient flow Morse representative. Every cell in the dataset flows via the underlying gradient path into a single cell on the Morse graph, the Morse representative of the cell. (E) Sankey flow plot showing how latent factor predicted cells are reassigned by Morse flow representatives.(F) Panel shows the original assignment [81] of cells to the types Slc17a8, Calb1 Tac2, Calb1 Kctd12, Calb1 Igfbp5, Calb1 Tnfaib813, and Vip. Overall agreement is 70.5%.

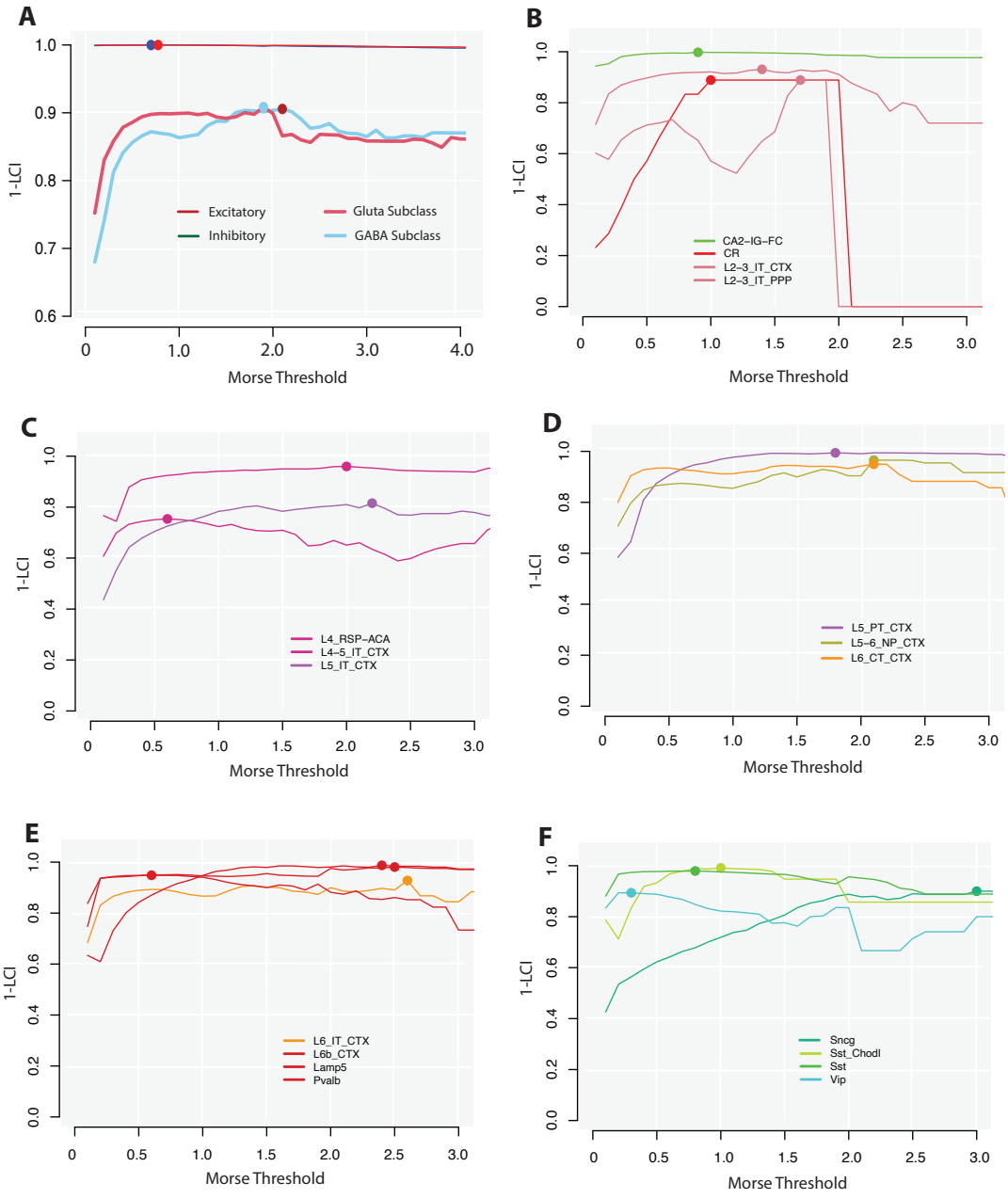


Figure C.4. (A) 1-LCI metric as a measure of cluster label stability. Highest lines show essentially perfect separation of GABAergic and glutamatergic cell types. Point indicates maximum Morse threshold for maximum 1-LCI. Mean of GABAergic and glutamatergic subclasses. (B-F) Subclass for 1-LCI curves with maxima for GABAergic and glutamatergic types [157]. 1-LCI depends on the Morse threshold and increases to a point of maximum cell type specificity, beyond which there are no coherent groups at that density. The maximum 1-LCI value is high for all subclass types (Inhibitory 0.949, Excitatory 0.924) and is consistent with the uniqueness of these cell classes.

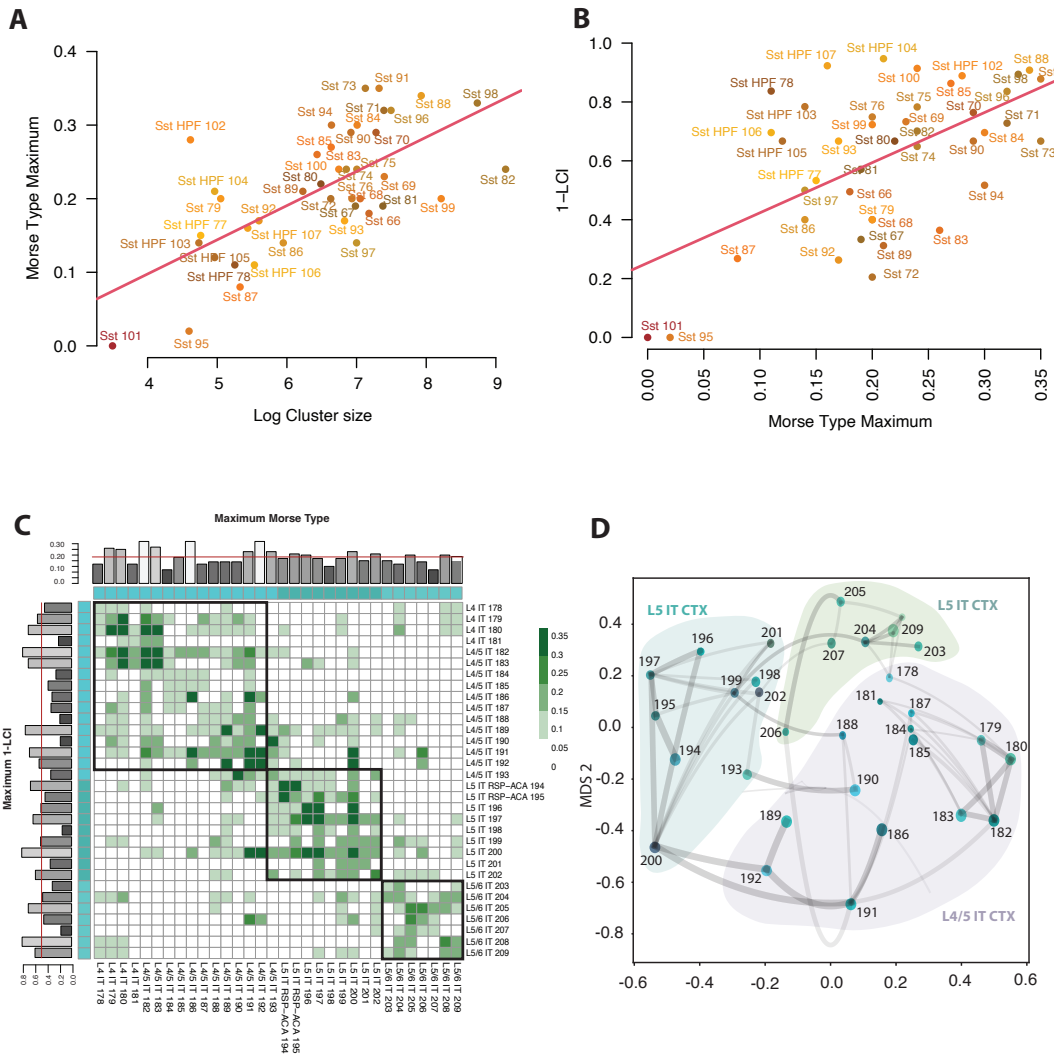


Figure C.5. (A) Plot of log of cell type cluster size as identified in [157] versus Morse Type Maximum (MTM), the largest δ for which c occurs on M_δ as a relative maximum. The measure MTM(c) increases as the fraction of neighbors of c with similar expression increases and is a natural measure of cell type uniqueness. MT is correlated ($\rho = 0.666$) with the log of cluster size for 42 Sst cell types. Sst 95 and Sst 101 are removed from the taxonomy. (B) Morse Type Maximum is correlated with 1-LCI label consistency ($\rho = 0.577$) and is measure of agreement or divergence of Morse taxonomy with the annotated labels. (C) Morse Persistence Taxonomy derived for glutamatergic L4 IT, L4/5 IT, L5 IT, and L5/6 IT cortex. Boxed types are three optimal Louvain groups that agree with [157]. (B) Visual constellation map representation is depicted similar to Figure 6.4D. Morse cell types are input to multidimensional scaling (MDS) to obtain the coordinates for each type. The size of the vertices scales as the maximum 1-LCI score with edge thickness determined by the magnitude of off-diagonal MPT entries and representing cell type confusion.

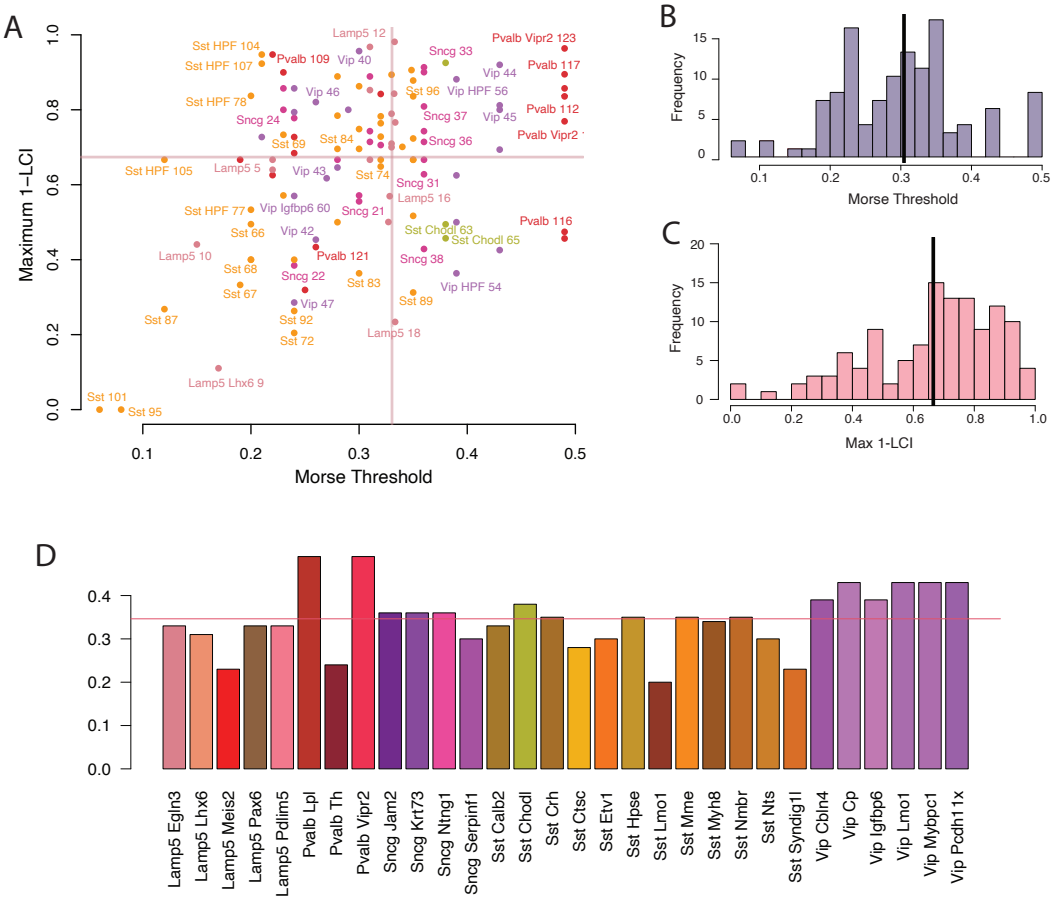


Figure C.6. (A) Scatter plot of ACA GABAergic supertypes showing relationship of Morse Type Maximum and maximum 1-LCI. Lines indicate mean values in each dimension. Colors as in [157]. (B) Histogram frequency of Morse threshold for GABAergic types with mean, (C) Maximum 1-LCI distribution. (D) Morse Type Maximum of each ACA GABAergic supertype.

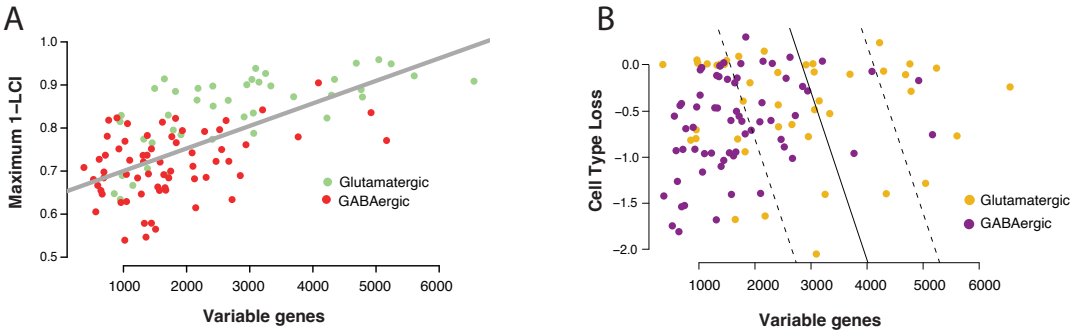


Figure C.7. (A) Variability in gene expression was measured in [101] over profiled cells finding that glutamatergic expressing neurons had a substantially larger number of variable genes. Maximum separability of supertypes based on maximum 1-LCI increases by the number of variable genes coded here by GABAergic or glutamatergic expressing supertypes. Adjusted R-squared: 0.3821, p-value: $p < 4.81x10 - 13$. (B) Effect loss is not fully explained by the increased number of variable genes. Soft margin SVM fitting variable genes to loss effect size separates GABAergic and glutamatergic neurons at F1 = 0.763.

Bibliography

[1] M. Aanjaneya, F. Chazal, D. Chen, M. Glisse, L. Guibas, and D. Morozov. Metric graph reconstruction from noisy data. In *Proc. 27th Sympos. Comput. Geom.*, pages 37–46, 2011.

[2] F Ali, SL Baringer, A Neal, EY Choi, and AC Kwan. Parvalbumin-positive neuron loss and amyloid-beta deposits in the frontal cortex of alzheimer's disease-related mice. *Journal of Alzheimer's Diesease*, 72(4):1323–1339, 2019.

[3] Tasic B, Yao Z, Graybuck LT, Smith KA, Nguyen TN, Bertagnolli D, Goldy J, Garren E, Economo MN, Viswanathan S, Penn O, Bakken T, Menon V, Miller J, Fong O, Hirokawa KE, Lathia K, Rimorin C, Tieu M, Larsen R, Casper T, Barkan E, Kroll M, Parry S, Shapovalova NV, Hirschstein D, Pendergraft J, Sullivan HA, Kim TK, Szafer A, Dee N, Groblewski P, Wickersham I, Cetin A, Harris JA, Levi BP, Sunkin SM, Madisen L, Daigle TL, Looger L, Bernard A, Phillips J, Lein E, Hawrylycz M, Svoboda K, Jones AR, Koch C, and Zeng H. Shared and distinct transcriptomic cell types across neocortical areas. *Nature*, 563(7729):72–78, 2018.

[4] S. Banerjee, L. Magee, D. Wang, X. Li, B. Huo, J. Jayakumar, K. Matho, M. Lin, K. Ram, M. Sivaprakasam, J. Huang, Y. Wang, and P. Mitra. Semantic segmentation of microscopic neuroanatomical data by combining topological priors with encoder-decoder deep networks. *Nature Machine Intelligence*, 2:585–594, 2020.

[5] Yuliy Baryshnikov and Robert Ghrist. Minimal unimodal decompositions on trees. *Journal of Applied and Computational Topology*, 4:199–209, 2020.

[6] Ulrich Bauer, Michael Kerber, and Jan Reininghaus. Distributed computation of persistent homology. *Proceedings of the Workshop on Algorithm Engineering and Experiments*, 10 2013.

[7] Ulrich Bauer, Talha Bin Masood, Barbara Giunti, Guillaume Houry, Michael Kerber, and Abhishek Rathod. Keeping it sparse: Computing persistent homology revised. *ArXiv*, abs/2211.09075, 2022.

[8] Etienne Becht, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immanuel Kwok, Lai Guan Ng, Florent Ginhoux, and Evan William Newell. Dimensionality reduction for visualizing single-cell data using umap. *Nature Biotechnology*, 37:38–44, 2018.

[9] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[10] Mikhail Belkin, Qichao Que, Yusu Wang, and X. Zhou. Toward understanding complex data: graph laplacians on manifolds with singularities and boundaries. In *Conf. Learning Theory (COLT)*, pages 36.1–36.26, 2012. Journal of Machine Learning Research – Proceedings Track 23.

[11] BRAIN Initiative Cell Census Network (BICCN). A multimodal cell census and atlas of the mammalian primary motor cortex. *Nature*, 598:86–102, 2021.

[12] Tatiana Bliskunova, Alma Delia Genis-Mendoza, José Jaime Martínez-Magaña, Julissa Gabriela Vega-Sevey, Janett Jiménez-Genchi, Andrés Roche, Rafael Guzmán, Leonor Zapata, Susana Castro-Chavira, Thalia Fernández, Jorge Ameth Villatoro-Velázquez, Beatriz Camarena, Clara Fleiz-Bautista, Marycarmen Bustos-Gamiño, María Elena Medina-Mora, and Humberto Nicolini. Association of mgat4c with major neurocognitive disorder in the mexican population. *Gene*, 778:145484, 2021.

[13] Jason W Bohland, Caizhi Wu, Helen Barbas, Hemant Bokil, Mihail Bota, Hans C Breiter, Hollis T Cline, John C Doyle, Peter J Freed, Ralph J Greenspan, et al. A proposal for a coordinated effort for the determination of brainwide neuroanatomical connectivity in model organisms at a mesoscopic scale. *PLoS computational biology*, 5(3), 2009.

[14] Eszter Boldog, Trygve E Bakken, Rebecca D. Hodge, Mark Novotny, Brian D. Aevermann, Judith Baka, Sándor Bordé, Jennie Close, Francisco Díez-Fuertes, Song-Lin Ding, Nóra Faragó, Agnes Katalin Kocsis, Balázs Kovács, Zoe Maltzer, Jamison M. McCorrison, Jeremy A. Miller, Gábor Molnár, Gáspár Oláh, Attila Ozsvár, Márton Rózsa, Soraya I. Shehata, Kimberly A. Smith, Susan M. Sunkin, Danny N. Tran, Pratap Venepally, Abby Wall, László G. Puskás, Pál Barzó, Frank Steemers, Nicholas J. Schork, Richard H. Scheuermann, Roger S. Lasken, Ed S. Lein, and Gábor Tamás. Transcriptomic and morphophysiological evidence for a specialized human cortical gabaergic cell type. *Nature neuroscience*, 21:1185 – 1195, 2017.

[15] Tristan Brugère, Zhengchao Wan, and Yusu Wang. Distances for markov chains, and their differentiation. In Claire Vernade and Daniel Hsu, editors, *Proceedings of The 35th International Conference on Algorithmic Learning Theory*, volume 237 of *Proceedings of Machine Learning Research*, pages 282–336. PMLR, 25–28 Feb 2024.

[16] Mickaël Buchet, Frédéric Chazal, Steve Y. Oudot, and Donald R. Sheehy. Efficient and robust persistent homology for measures. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 168–180, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics.

[17] Alexander Buslaev, Selim S Seferbekov, Vladimir Iglovikov, and Alexey Shvets. Fully convolutional network for automatic road extraction from satellite imagery. In *CVPR Workshops*, pages 207–210, 2018.

[18] Andrew Butler, Paul J. Hoffman, Peter Smibert, Efthymia Papalexi, and Rahul Satija. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature Biotechnology*, 36:411–420, 2018.

[19] Trapnell C, Cacchiarelli D, Grimsby J, Pokharel P, Li S, Morse M, Lennon NJ, Livak KJ, Mikkelsen TS, and Rinn JL. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature Biotechnology*, 32(4):381–386, 2014.

[20] Chen Cai, Nikolaos Vlassis, Lucas Magee, Ran Ma, Zeyu Xiong, Bahador Bahmani, Teng-Fong Wong, Yusu Wang, and WaiChing Sun. Equivariant geometric learning for digital rock physics: estimating formation factor and effective permeability tensors from morse graph, 2021.

[21] California Department of Transportation. Traffic flows at detector #409529, 2017.

[22] Gunnar Carlsson, Tigran Ishkhanov, Vin Silva, and Afra Zomorodian. On the local behavior of spaces of natural images. *International Journal of Computer Vision*, 76:1–12, 01 2008.

[23] Gunnar Carlsson and Mikael Vejdemo-Johansson. *Topological Data Analysis with Applications*. Cambridge University Press, 2021.

[24] Mark S. Cembrowski, Julia L. Bachman, Lihua Wang, Ken Sugino, Brenda C. Shields, and Nelson Spruston. Spatial gene-expression gradients underlie prominent heterogeneity of ca1 pyramidal neurons. *Neuron*, 89(2):351–368, 2016.

[25] T Chari and L Pachter. The specious art of single-cell genomics. *PLoS Computational Biology*, 19(8):e1011288, 2023.

[26] F. Chazal, D. Cohen-Steiner, and Q. Mérigot. Geometric inference for probability measures. *Foundations of Computational Mathematics*, 11:733–751, 2011.

[27] Frédéric Chazal, Ruqi Huang, and Jian Sun. Gromov—hausdorff approximation of filamentary structures using reeb-type graphs. *Discrete Comput. Geom.*, 53(3):621–649, April 2015.

[28] Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. *The structure and stability of persistence modules*. Springer, 2018.

[29] Hanbo Chen, Hang Xiao, Tianming Liu, and Hanchuan Peng. Smarttracing: self-learning-based neuron reconstruction. *Brain Informatics*, 2(3):135–144, 2015.

[30] J Chen, EE Bardes, BJ Aronow, and AG Jegga. Toppgene suite for gene list enrichment analysis and candidate gene prioritization. *Nucleic Acids Research*, 37 (Web Server issue):W305–11, 2009.

[31] Samantha Chen, Sunhyuk Lim, Facundo Memoli, Zhengchao Wan, and Yusu Wang. Weisfeiler-lehman meets gromov-Wasserstein. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 3371–3416. PMLR, 17–23 Jul 2022.

[32] Robert Clarke, Habtom W. Ressom, Antai Wang, Jianhua Xuan, Minetta C. Liu, Edmund A. Gehan, and Yue Joseph Wang. The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. *Nature Reviews Cancer*, 8:37–49, 2008.

[33] David Cohen-Steiner, André Lieutier, and Julien Vuillamy. Lexicographic optimal chains and manifold triangulations, 2019. available at URL: https://hal.archives-ouvertes.fr/hal-02391190/document.

[34] David Cohen-Steiner, André Lieutier, and Julien Vuillamy. Lexicographic optimal homologous chains and applications to point cloud triangulations. In *36th Sympos. Comput. Geom. (SoCG)*, 2020. to appear, see also url: https://hal.archives-ouvertes.fr/hal-02391240/document.

[35] Kobak D and Berens P. The art of using t-sne for single-cell transcriptomics. *Nature Communications*, 10(1):5416, 2019.

[36] Lähnemann D, Köster J, Szczurek E, McCarthy DJ, Hicks SC, Robinson MD, Vallejos CA, Campbell KR, Beerenwinkel N, Mahfouz A, Pinello L, Skums P, Stamatakis A, Attolini CS, Aparicio S, Baaijens J, Balvert M, Barbanson B, Cappuccio A, Corleone G, Dutilh BE, Florescu M, Guryev V, Holmer R, Jahn K, Lobo TJ, Keizer EM, Khatri I, Kielbasa SM, Korbel JO, Kozlov AM, Kuo TH, Lelieveldt BPF, Mandoiu II, Marioni JC, Marschall T, Mölder F, Niknejad A, Raczkowska A, Reinders M, Ridder J, Saliba AE, Somarakis A, Stegle O, Theis FJ, Yang H, Zelikovsky A, McHardy AC, Raphael BJ, Shah SP, and Schönhuth A. Eleven grand challenges in single-cell data science. *Genome Biology*, 21(1):31, 2020.

[37] MAX de Lima, MVC Baldo, FA Oliveira, and NS Canteras. The anterior cingulate cortex and its role in controlling contextual fear memory to predatory threats. *eLife*, 11:e67007, 2022.

[38] O. Delgado-Friedrichs, V. Robins, and A. Sheppard. Skeletonization and partitioning of digital images using discrete morse theory. *IEEE Trans. Pattern Anal. Machine Intelligence*, 37(3):654–666, March 2015.

[39] Van den Berge K, Roux de Bézieux H, Street K, Saelens W, Cannoodt R, Saeys Y, Dudoit S, and Clement L. Trajectory-based differential expression analysis for single-cell sequencing data. *Nature Communications*, 11(1):1201, 2020.

[40] T. K. Dey, J. Wang, and Y. Wang. Graph reconstruction by discrete morse theory. In *Proc. Internat. Sympos. Comput. Geom.*, pages 31:1–31:15, 2018.

[41] Tamal Dey, Jiayuan Wang, and Yusu Wang. Road network reconstruction from satellite images with machine learning supported by topological methods. In *Proc. 27th ACM SIGSPATIAL Intl. Conf. Adv. Geographic Information Systems (GIS)*, pages 520–523, 2019.

[42] Tamal K. Dey, Tao Hou, and Sayan Mandal. Computing minimal persistent cycles: Polynomial and hard cases. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2587–2606. SIAM, 2020.

[43] Tamal K. Dey, Jiayuan Wang, and Yusu Wang. Improved road network reconstruction using discrete morse theory. In *Proc. 25th ACM SIGSPATIAL Intl. Conf. Adv. Geographic Information Systems (GIS)*, pages 58:1–58:4, 2017.

[44] Tamal Krishna Dey and Yusu Wang. *Computational Topology for Data Analysis*. Cambridge University Press, 2022.

[45] Hans-Ulrich Dodt, Ulrich Leischner, Anja Schierloh, Nina Jährling, Christoph Peter Mauch, Katrin Deininger, Jan Michael Deussing, Matthias Eder, Walter Zieglgänsberger, and Klaus Becker. Ultramicroscopy: three-dimensional visualization of neuronal networks in the whole mouse brain. *Nature methods*, 4(4):331–336, 2007.

[46] D.L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.

[47] Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. Amer. Math. Soc., Providence, Rhode Island, 2010.

[48] Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. 01 2010.

[49] S Fischer and J Gillis. How many markers are needed to robustly determine a cell's type? *iScience*, 24(11):103292, 2021.

[50] Remi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, AurÃ©lie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, LÃ©o Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021.

[51] R. Forman. Combinatorial vector fields and dynamic systems. *Mathematische Zeitschrift*, 228(4):629–681, 1998.

[52] Robin Forman. A user's guide to discrete Morse theory. *Séminare Lotharinen de Combinatore 48*, 2002.

[53] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 1*, 2:559–572, 1901.

[54] Rohan Gala, Julio Chapeton, Jayant Jitesh, Chintan Bhavsar, and Armen Stepanyants. Active learning of neuron morphology for accurate automated tracing of neurites. *Frontiers in Neuroanatomy*, 8:37, 2014.

[55] X Gao, F Bender, H Soh, C Chen, M Altafi, S Schütze, M Heidenreich, M Gorbati, M Corbu, M Carus-Cadavieco, T Korotkova, A Tzingounis, T Jentsch, and A Ponomarenko. Place fields of single spikes in hippocampus involve kcnq3 channel-dependent entrainment of complex spike bursts. *Nature Communications*, 12(1):4801, 2021.

[56] Manuel Garber, Manfred G. Grabherr, Mitchell Guttman, and Cole Trapnell. Computational methods for transcriptome annotation and quantification using rna-seq. *Nature Methods*, 8:469–477, 2011.

[57] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[58] Xiaoyin Ge, Issam I. Safa, Mikhail Belkin, and Yusu Wang. Data skeletonization via reeb graphs. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 837–845. Curran Associates, Inc., 2011.

[59] J Glaz, J Naus, and S Wallenstein. *Scan Statistics*. Springer Science and Business Media, 2013.

[60] Wuming Gong, Il-Youp Kwak, Naoko Koyano-Nakagawa, Wei Pan, and Daniel Garry. Tcm visualizes trajectories and cell populations from single cell data. *Nature Communications*, 9, 07 2018.

[61] Fiorella Grandi, Hailey Modi, Lucas Kampman, and M. Corces. Chromatin accessibility profiling by atac-seq. *Nature Protocols*, 17:1–35, 04 2022.

[62] Minzhe Guo, Erik L. Bao, Michael Wagner, Jeffrey A. Whitsett, and Yan Xu. Slice: determining cell differentiation and lineage based on single cell entropy. *Nucleic Acids Research*, 45:e54 – e54, 2016.

[63] A. Gyulassy, M. Duchaineau, V. Natarajan, V. Pascucci, E. Bringa, A. Higginbotham, and B. Hamann. Topologically clean distance fields. *IEEE Trans. Visualization Computer Graphics*, 13(6):1432–1439, Nov 2007.

[64] Maryam Halavi, Kelly Andrew Hamilton, Ruchi Parekh, and Giorgio Ascoli. Digital reconstructions of neuronal morphology: three decades of research trends. *Frontiers in neuroscience*, 6:49, 2012.

[65] Zhou Hang, Li Shiwei, Li Anan, Xiong Feng, Li Ning, Han Jiacheng, Kang Hongtao, Chen Yijun, Li Yun, Fang Wenqian, Liu Yidong, Lin Huimin, Jin Sen, Li Zhiming, Xu Fuqiang, Zhang Yu-hui, Lv Xiaohua, Liu Xiuli, Gong Hui, Luo Qingming, Quan Tingwei, and Zeng Shaoqun. Dense reconstruction of brain-wide neuronal population close to the ground truth. *bioRxiv*, 2018.

[66] Yuhan Hao, Tim Stuart, Madeline H Kowalski, Saket Choudhary, Paul Hoffman, Austin Hartman, Avi Srivastava, Gesmira Molla, Shaista Madad, Carlos Fernandez-Granda, and Rahul Satija. Dictionary learning for integrative, multimodal and scalable single-cell analysis. *Nature Biotechnology*, 2023.

[67] Frank Harary and George E Uhlenbeck. On the number of husimi trees: I. *Proceedings of the National Academy of Sciences*, 39(4):315–322, 1953.

[68] E. Hare, S. Hedetniemi, R. Laskar, K. Peters, and T. Wimer. Linear-time computability of combinatorial problems on generalized-series-parallel graphs. In David S. Johnson, Takao Nishizeki, Akihiro Nozaki, and Herbert S. Wilf, editors, *Discrete Algorithms and Complexity*, pages 437–457. Academic Press, 1987.

[69] T. J. Hastie. *Principal curves and surfaces*. PhD thesis, Stanford University, 1984.

[70] Michael J. Hawrylycz, Ed S. Lein, Angie Guillozet-Bongaarts, Elaine H. Shen, Lydia Ng, Jeremy A. Miller, Louie N. van de Lagemaat, Kimberly A. Smith, Amanda J. Ebbert, Zackery L. Riley, Chris Abajian, Christian F. Beckmann, Amy Bernard, Darren Bertagnolli, Andrew F. Boe, Preston M. Cartagena, M. Mallar Chakravarty, Mike Chapin, Jimmy Chong, Rachel Dalley, Barry David Daly, Chinh Dang, Suvro Sankha Datta, Nick Dee, Tim Dolbeare, Vance Faber, David Feng, David R. Fowler, Jeff Goldy, Benjamin W. Gregor, Zeb Haradon, David R. Haynor, John G. Hohmann, Steve Horvath, Robert E. Howard, A. Jeromin, Jayson M. Jochim, Marty Kinnunen, Christopher Lau, Evan T. Lazarz, Changkyu Lee, Tracy A. Lemon, Lingling Li, Yang Li, J. Morris, Caroline C. Overly, Patrick D. Parker, Sheana E. Parry, Melissa Reding, Joshua J. Royall, Jay Schulkin, P. Adolfo Sequeira, Cliff Slaughterbeck, Simon C. Smith, Andrew Sodt, Susan M. Sunkin, Beryl E. Swanson, Marquis P. Vawter, Derric Williams, Paul E. Wohnoutka, Horst Ronald Zielke, Daniel H. Geschwind, Patrick R. Hof, Stephen M. Smith, Christof Koch, Seth G. N. Grant, and Allan R. Jones. An anatomically comprehensive atlas of the adult human brain transcriptome. *Nature*, 489:391–399, 2012.

[71] P.R.A. Heckman, A. Blokland, J. Ramaekers, and J. Prickaerts. Pde and cognitive processing: Beyond the memory domain. *Neurobiology of Learning and Memory*, 119:108 – 122, 2015.

[72] Moritz Helmstaedter and Partha P Mitra. Computational methods and challenges for large-scale circuit mapping. *Current opinion in neurobiology*, 22(1):162–169, 2012.

[73] Gunawan I, Vafaee F, Meijering E, and Lock JG. An introduction to representation learning for single-cell data analysis. *Cell Reports Methods*, 3(8), 2023.

[74] Langlieb J, Sachdev NS, Balderrama KS, Nadaf NM, Raj M, Murray E, Webber JT, Vanderburg C, Gazestani V, Tward D, Mezias C, Li X, Flowers K, Cable DM, Norton T, Mitra P, Chen F, and Macosko EZ. The molecular cytoarchitecture of the adult mouse brain. *Nature*, 624(7991):333–342, 2023.

[75] Sheng J, Zhang S, Wu L, Kumar G, Liao Y, Gk P, and Fan H. Inhibition of phosphodiesterase: A novel therapeutic target for the treatment of mild cognitive impairment and alzheimer's disease. *Frontiers in Aging Neuroscience*, 14(October):1019187, 2022.

[76] KA Jellinger. Neuropathological assessment of the alzheimer spectrum. *Journal of Neural Transmission (Vienna)*, 127(9):1229–1256, 2020.

[77] Z Ji and H Ji. Tscan: Pseudo-time reconstruction and evaluation in single-cell rna-seq analysis. *Nucleic Acids Research*, 44(13):e117, 2016.

[78] William Johnson and Joram Lindenstrauss. Extensions of lipschitz maps into a hilbert space. *Contemporary Mathematics*, 26:189–206, 01 1984.

[79] Iain Johnstone and D. Titterington. Statistical challenges of high-dimensional data. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 367:4237–53, 11 2009.

[80] Harish Kannan, Emil Saucan, Indrava Roy, and Areejit Samal. Persistent homology of unweighted complex networks via discrete morse theory. *Scientific Reports*, 9, 2019.

[81] Harris KD, Hochgerner H, Skene NG, Magno L, Katona L, Bengtsson Gonzales C, Somogyi P, Kessaris N, Linnarsson S, and Hjerling-Leffler J. Classes and continua of hippocampal ca1 inhibitory neurons revealed by single-cell transcriptomics. *PLoS Biology*, 16(6):e2006387, 2018.

[82] B. Kégl and A. Krzyżak. Piecewise linear skeletonization using principal curves. *IEEE Trans. Pattern Anal. Machine Intell.*, 24:59–74, January 2002.

[83] PV Kharchenko. The triumphs and limitations of computational methods for scrna-seq. *Nature Methods*, 18(7):723–732, 2021.

[84] Dhruv Kohli, Alexander Cloninger, and Gal Mishne. Ldle: Low distortion local eigenmaps. *Journal of Machine Learning Research*, 22(282):1–64, 2021.

[85] Dmitry N Kozlov. *Organized Collapse: An Introduction to Discrete Morse Theory*. American Mathematical Society, 2020.

[86] VS Kumar, Sunil Arya, and Hariharan Ramesh. Hardness of set cover with intersection 1. In *International Colloquium on Automata, Languages, and Programming*, pages 624–635. Springer, 2000.

[87] Heumos L, Schaar AC, Lance C, Litinetskaya A, Drost F, Zappia L, Lücken MD, Strobl DC, Henao J, Curion F, Single cell Best Practices Consortium, Schiller HB, and Theis FJ. Best practices for single-cell analysis across modalities. *Nature Reviews Genetics*, 24(8):550–572, 2023.

[88] Schmid LC, Mittag M, Poll S, Steffen J, Wagner J, Geis HR, Schwarz I, Schmidt B, Schwarz MK, Remy S, and Fuhrmann M. Dysfunction of somatostatin-positive interneurons associated with memory deficits in an alzheimer's disease model. *Neuron*, 92(1):114–125, 2016.

[89] Fabrizio Lecci, Alessandro Rinaldo, and Larry Wasserman. Statistical analysis of metric graph reconstruction. *J. Mach. Learn. Res.*, 15(1):3425–3446, January 2014.

[90] A. Lee, K. Pedersen, and D. Mumford. The nonlinear statistics of high-contrast patches in natural images. *International Journal of Computer Vision*, 54:83–103, 2004.

[91] Ryan Lewis and Dmitriy Morozov. Parallel computation of persistent homology using the blowup complex. SPAA '15, page 323–331, New York, NY, USA, 2015. Association for Computing Machinery.

[92] Meng Kuan Lin, Yeonsook Shin Takahashi, Bing-Xing Huo, Mitsutoshi Hanada, Jaimi Nagashima, Junichi Hata, Alexander S Tolpygo, Keerthi Ram, Brian C Lee, Michael I Miller, et al. A high-throughput neurohistological pipeline for brain-wide mesoscale connectivity mapping of the common marmoset. *Elife*, 8:e40042, 2019.

[93] Claire Lugnier. Cyclic nucleotide phosphodiesterase (pde) superfamily: A new target for the development of specific therapeutic agents. *Pharmacology and Therapeutics*, 109(3):366–398, 2006.

[94] Hawrylycz M, Kaplan ES, Travaglini KJ, Gabitto MI, Miller JA, Ng L, Close JL, Hodge RD, Long B, Mollenkopf T, Mufti S, Gatto NM, Larson EB, Crane PK, Grabowski TJ, Keene CD, and Lein ES. Sea-ad is a multimodal cellular atlas and resource for alzheimer's disease. *Nature Aging*, 4(10):1331–1334, 2024.

[95] Piwecka M, Rajewsky N, and Rybak-Wolf A. Single-cell and spatial transcriptomics: deciphering brain complexity in health and disease. *Nature Revolutionary Neurology*, 19(6):346–362, 2023.

[96] L. Magee and Y. Wang. Graph skeletonization of high-dimensional point cloud data via topological method. *Journal of Computational Geometry*, 13:429–470, 2022.

[97] Lucas Magee and Yusu Wang. Minimum monotone tree decomposition of density functions defined on graphs. In Weili Wu and Jianxiong Guo, editors, *Combinatorial Optimization and Applications*, pages 107–125, Cham, 2024. Springer Nature Switzerland.

[98] Y Marghi, R Gala, Baftizadeh F, and U Sümbül. Joint inference of discrete cell types and continuous type-specific variability in single-cell datasets with mmidas. *bioRxiv*, 2024.

[99] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[100] Nimrod Megiddo and Arie Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1(5):194–197, 1982.

[101] Gabitto MI, Travaglini KJ, Rachleff VM, Kaplan ES, Long B, Ariza J, Ding Y, Mahoney JT, Dee N, Goldy J, Melief EJ, Brouner K, Campos J, Carr AJ, Casper T, Chakrabarty R, Clark M, Compos J, Cool J, Valera Cuevas NJ, Dalley R, Darvas M, Ding SL, Dolbeare T, Mac Donald CL, Egdorf T, Esposito L, Ferrer R, Gala R, Gary A, Gloe J, Guilford N, Guzman J, Ho W, Jarksy T, Johansen N, Kalmbach BE, Keene LM, Khawand S, Kilgore M, Kirkland A, Kunst M, Lee BR, Malone J, Maltzer Z, Martin N, McCue R, McMillen D, Meyerdierks E, Meyers KP, Mollenkopf T, Montine M, Nolan AL, Nyhus J, Olsen PA, Pacleb M, Pham T, Pom CA, Postupna N, Ruiz A, Schantz AM, Sorensen SA, Staats B, Sullivan M, Sunkin SM, Thompson C, Tieu M, Ting J, Torkelson A, Tran T, Wang MQ, Waters J, Wilson AM, Haynor D, Gatto N, Jayadev S, Mufti S, Ng L, Mukherjee S, Crane PK, Latimer CS, Levi BP, Smith K, Close JL, Miller JA, Hodge RD, Larson EB, Grabowski TJ, Hawrylycz M, Keene CD, and Lein ES. Integrated multimodal cell atlas of alzheimer's disease. *Nature Neuroscience*, 2024.

[102] Zhen Miao, Ben Humphreys, Andrew McMahon, and Junhyong Kim. Multi-omics integration in the age of million single-cell data. *Nature Reviews Nephrology*, 17:1–15, 08 2021.

[103] J Milnor. *Morse Theory. (AM-51), Volume 51*. Princeton University Press, 2016.

[104] MH Murdock and LH Tsai. Insights into alzheimer's disease from single-cell genomic approaches. *Nature Neuroscience*, 26(2):181–195, 2023.

[105] Nicholas Navin, Orit Rozenblatt-Rosen, and Nancy Zhang. New frontiers in single-cell genomics. *Genome Research*, 31:ix–x, 10 2021.

[106] Nayar and H. Murase. Columbia object image library: Coil-100. Technical Report CUCS-006-96, Department of Computer Science, Columbia University, February 1996.

[107] Ashton NJ, Nevado-Holgado AJ, Barber IS, Lynham S, Gupta V, Chatterjee P, Goozee K, Hone E, Pedrini S, Blennow K, Schöll M, Zetterberg H, Ellis KA, Bush AI, Rowe CC, Villemagne VL, Ames D, Masters CL, Aarsland D, Powell J, Lovestone S, Martins R, and Hye A. A plasma protein classifier for predicting amyloid burden for preclinical alzheimer's disease. *Science Advances*, 5(2):eaau7220, 2019.

[108] Jorstad NL, Song JHT, Exposito-Alonso D, Suresh H, Castro-Pacheco N, Krienen FM, Yanny AM, Close J, Gelfand E, Long B, Seeman SC, Travaglini KJ, Basu S, Beaudin M, Bertagnolli D, Crow M, Ding SL, Eggermont J, Glandon A, Goldy J, Kiick K, Kroes T, McMillen D, Pham T, Rimorin C, Siletti K, Somasundaram S, Tieu M, Torkelson A, Feng G, Hopkins WD, Höllt T, Keene CD, Linnarsson S, McCarroll SA, Lelieveldt BP, Sherwood CC, Smith K, Walsh CA, Dobin A, Gillis J, Lein ES, Hodge RD, and Bakken

TE. Comparative transcriptomics reveals human-specific cortical features. *Science*, 382(6667):eade9516, 2023.

[109] Seung Wook Oh, Julie A Harris, Lydia Ng, Brent Winslow, Nicholas Cain, Stefan Mihalas, Quanxin Wang, Chris Lau, Leonard Kuan, Alex M Henry, et al. A mesoscale connectome of the mouse brain. *Nature*, 508(7495):207–214, 2014.

[110] U. Ozertem and D. Erdogmus. Locally defined principal curves and surfaces. *Journal of Machine Learning Research*, 12:1249–1286, 2011.

[111] Hanchuan Peng, Michael Hawrylycz, Jane Roskams, Sean Hill, Nelson Spruston, Erik Meijering, and Giorgio A Ascoli. Bigneuron: large-scale 3d neuron reconstruction from optical microscopy images. *Neuron*, 87(2):252–256, 2015.

[112] Hanchuan Peng, Erik Meijering, and Giorgio A Ascoli. From diadem to bigneuron, 2015.

[113] Jose A. Perea and John Harer. Sliding windows and persistence: An application of topological methods to signal analysis. *Found. Comput. Math. (FoCM)*, 15:799—838, 2015. https://doi.org/10.1007/s10208-014-9206-z.

[114] Azam Peyvandipour, Adib Shafi, Nafiseh Saberian, and Sorin Draghici. Identification of cell types from single cell data using stable clustering. *Scientific Reports*, 10, 07 2020.

[115] Vadim Pinskiy, Jamie Jones, Alexander S Tolpygo, Neil Franciotti, Kevin Weber, and Partha P Mitra. High-throughput method of whole-brain sectioning, using the tape-transfer technique. *PloS one*, 10(7), 2015.

[116] Tingwei Quan, Hang Zhou, Jing Li, Shiwei Li, Anan Li, Yuxin Li, Xiaohua Lv, Qingming Luo, Hui Gong, and Shaoqun Zeng. Neurogps-tree: automatic reconstruction of large-scale neuronal populations with dense neurites. *Nature Methods*, 13(1):51–54, 2016.

[117] Yuste R, Hawrylycz M, Aalling N, Aguilar-Valles A, Arendt D, Armañanzas R, Ascoli GA, Bielza C, Bokharaie V, Bergmann TB, Bystron I, Capogna M, Chang Y, Clemens A, de Kock CPJ, DeFelipe J, Dos Santos SE, Dunville K, Feldmeyer D, Fiáth R, Fishell GJ, Foggetti A, Gao X, Ghaderi P, Goriounova NA, Güntürkün O, Hagihara K, Hall VJ, Helmstaedter M, Herculano-Houzel S, Hilscher MM, Hirase H, Hjerling-Leffler J, Hodge R, Huang J, Huda R, Khodosevich K, Kiehn O, Koch H, Kuebler ES, Kühnemund M, Larrañaga P, Lelieveldt B, Louth EL, Lui JH, Mansvelder HD, Marin O, Martinez-Trujillo J, Chameh HM, Mohapatra AN, Munguba H, Nedergaard M, Němec P, Ofer N, Pfisterer UG, Pontes S, Redmond W, Rossier J, Sanes JR, Scheuermann RH, Serrano-Saiz E, Staiger JF, Somogyi P, Tamás G, Tolias AS, Tosches MA, García MT, Wozny C, Wuttke TV, Liu Y, Yuan J, Zeng H, and Lein E. A community-based transcriptomics classification and nomenclature of neocortical cell types. *Nature Neuroscience*, 23(12):1456–1468, 2020.

[118] Raul Rabadan and Andrew J. Blumberg. *Topological Data Analysis for Genomics and Evolution: Topology in Biology*. Cambridge University Press, 2019.

[119] Timothy Ragan, Lolahon R Kadiri, Kannan Umadevi Venkataraju, Karsten Bahlmann, Jason Sutin, Julian Taranda, Ignacio Arganda-Carreras, Yongsoo Kim, H Sebastian Seung, and Pavel Osten. Serial two-photon tomography for automated ex vivo mouse brain imaging. *Nature methods*, 9(3):255, 2012.

[120] Hodge RD, Bakken TE, Miller JA, Smith KA, Barkan ER, Graybuck LT, Close JL, Long B, Johansen N, Penn O, Yao Z, Eggermont J, Höllt T, Levi BP, Shehata SI, Aevermann B, Beller A, Bertagnolli D, Brouner K, Casper T, Cobbs C, Dalley R, Dee N, Ding SL, Ellenbogen RG, Fong O, Garren E, Goldy J, Gwinn RP, Hirschstein D, Keene CD, Keshk M, Ko AL, Lathia K, Mahfouz A, Maltzer Z, McGraw M, Nguyen TN, Nyhus J, Ojemann JG, Oldre A, Parry S, Reynolds S, Rimorin C, Shapovalova NV, Somasundaram S, Szafer A, Thomsen ER, Tieu M, Quon G, Scheuermann RH, Yuste R, Sunkin SM, Lelieveldt B, Feng D, Ng L, Bernard A, Hawrylycz M, Phillips JW, Tasic B, Zeng H, Jones AR, Koch C, and Lein ES. Conserved cell types with divergent features in human versus mouse cortex. *Nature*, 573(7772):61–68, 2019.

[121] Nicolas Rey-Villamizar, Vinay Somasundar, Murad Megjhani, Yan Xu, Yanbin Lu, Raghav Padmanabhan, Kristen Trett, William Shain, and Badri Roysam. Large-scale automated image analysis for computational profiling of brain tissue surrounding implanted neuroprosthetic devices using python. *Frontiers in neuroinformatics*, 8:39, 2014.

[122] Jonas Richiardi, Andre Altmann, Anna-Clare Milazzo, Catie Chang, M. Chakravarty, Tobias Banaschewski, Gareth Barker, Arun Bokde, Uli Bromberg, Christian Büchel, Patricia Conrod, Mira Fauth-Bühler, Herta Flor, Vincent Frouin, Jürgen Gallinat, Hugh Garavan, Penny Gowland, Andreas Heinz, Hervé Lemaître, and The consortium. Brain networks. correlated gene expression supports synchronous activity in brain networks. *Science*, 348:1241–1244, 06 2015.

[123] V. Robins, P. J. Wood, and A. P. Sheppard. Theory and algorithms for constructing discrete morse complexes from grayscale digital images. *IEEE Trans. Pattern Anal. Machine Intelligence*, 33(8):1646–1658, Aug 2011.

[124] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[125] S.T. Roweis and L.K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323, 2000.

[126] JV Sanchez-Mut and J Gräff. Epigenetic alterations in alzheimer's disease. *Frontiers in Behavioral Neuroscience*, 9(December):347, 2015.

[127] Nicholas A Scoville. *Discrete Morse Theory*. American Mathematical Society, 2019.

[128] M Sekine and T Makino. Inference of causative genes for alzheimer's disease due to dosage imbalance. *Molecular Biology and Evolution*, 34(9):2396–2407, 2017.

[129] Manu Setty, Michelle D. Tadmor, Shlomit Reich-Zeliger, Omer Angel, Tomer Meir Salame, Pooja Kathail, Kristy Choi, Sean C. Bendall, Nir Friedman, and Dana Pe'er. Wishbone identifies bifurcating developmental trajectories from single-cell data. *Nature biotechnology*, 34:637 – 645, 2016.

[130] Gurjeet Singh, Facundo Memoli, and Gunnar Carlsson. Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition. In M. Botsch, R. Pajarola, B. Chen, and M. Zwicker, editors, *Eurographics Symposium on Point-Based Graphics*. The Eurographics Association, 2007.

[131] YH Song, J Yoon, and SH Lee. The role of neuropeptide somatostatin in the brain and its application in treating neurological disorders. *Experimental and Molecular Medicine*, 53(3):328–338, 2021.

[132] P Sonpatki and N Shah. Recursive consensus clustering for novel subtype discovery from transcriptome data. *Scientific Reports*, 10(1):11005, 2020.

[133] H Sontheimer. *Diseases of the Nervous System*. Elsevier, 2021.

[134] Thierry Sousbie. The persistent cosmic web and its filamentary structure – i. theory and implementation. *Monthly Notices of the Royal Astronomical Society*, 414:350 – 383, 06 2011.

[135] J.B. Tenenbaum, V. Silva, and J.C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319, 2000.

[136] VA Traag, L Waltman, and NJ van Eck. From louvain to leiden: guaranteeing well-connected communities. *Nature Scientific Reports*, 9(1):5233, 2019.

[137] C Trapnell. Defining cell types and states with single-cell genomics. *Genome Research*, 25(10):1491–8, 2015.

[138] Eze UC, Bhaduri A, Haeussler M, Nowakowski TJ, and Kriegstein AR. Single-cell atlas of early human brain development highlights heterogeneity of human neuroepithelial cells and early radial glia. *Nature Neuroscience.*, 24(4):584–594, 2021.

[139] Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

[140] A Vasighizaker, S Danda, and L Rueda. Discovering cell types using manifold learning and enhanced visualization of single-cell rna-seq data. *Scientific Reports*, 12(1):120, 2022.

[141] C Villa. *Novel Biomarkers in Alzheimer's Disease*. MDPI, 2021.

[142] Cédric Villani. *Optimal transport – Old and new*, volume 338, pages xxii+973. Springer Science and Business Media, 01 2008.

[143] Jacob W. Vogel, Aaron F. Alexander-Bloch, Konrad Wagstyl, Maxwell A. Bertolero, Ross D. Markello, Adam Pines, Valerie J. Sydnor, Alex Diaz-Papkovich, Justine Y. Hansen, Alan C. Evans, Boris Bernhardt, Bratislav Misic, Theodore D. Satterthwaite, and Jakob Seidlitz. Deciphering the functional specialization of whole-brain spatiomolecular gradients in the adult brain. *Proceedings of the National Academy of Sciences*, 121(25):e2219137121, 2024.

[144] Kiselev VY, Kirschner K, Schaub MT, Andrews T, Yiu A, Chandra T, Natarajan KN, Reik W, Barahona M, Green AR, and Hemberg M. Sc3: consensus clustering of single-cell rna-seq data. *Nature Methods*, 14(5):483–486, 2017.

[145] B Wang, J Zhu, E Pierson, D Ramazzotti, and S Batzoglou. Visualization and analysis of single-cell rna-seq data by kernel-based similarity learning. *Nature Methods*, 14(4):414–416, 2017.

[146] Dingkang Wang, Lucas Magee, Bing-Xing Huo, Samik Banerjee, Xu Li, Jaikishan Jayakumar, Meng Kuan Lin, Keerthi Ram, Suyi Wang, Yusu Wang, and Partha Mitra. Detection and skeletonization of single neurons and tracer injections using topological methods. *arXiv preprint arXiv:2004.02755*, 2020.

[147] S. Wang, Y. Wang, and Y. Li. Efficient map reconstruction and augmentation via topological methods. In *Proc. 23rd ACM SIGSPATIAL*, page 25. ACM, 2015.

[148] Suyi Wang. *Analyzing data with 1D non-linear shapes using topological methods*. PhD thesis, The Ohio State University, Computer Science and Engineering Department, 2018.

[149] Boris Weisfeiler and A. A. Lehman. A Reduction of a Graph to a Canonical Form and an Algebra Arising During This Reduction. *Nauchno-Technicheskaya Informatsia*, Ser. 2(N9):12–16, 1968.

[150] Pengxiang Wu, Chao Chen, Yusu Wang, Shaoting Zhang, Changhe Yuan, Zhen Qian, Dimitris N. Metaxas, and Leon Axel. Optimal topological cycles and their application in cardiac trabeculae restoration. In *Information Processing in Medical Imaging - 25th International Conference, IPMI 2017, Boone, NC, USA, June 25-30, 2017, Proceedings*, pages 80–92, 2017.

[151] Deng XH, Liu XY, Wei YH, Wang K, Zhu JR, Zhong JJ, Zheng JY, Guo R, Zhu YF, Ye QH, Wang MD, Chen YJ, He JQ, Chen ZX, Huang SQ, Lv CS, Zheng GQ, Liu SF, and Wen L. Erbb4 deficiency exacerbates olfactory dysfunction in an early-stage alzheimer's disease mouse model. *Acta Pharmacologica Sinica*, July 2024.

[152] Hang Xiao and Hanchuan Peng. App2: automatic tracing of 3d neuron morphology based on hierarchical pruning of a gray-weighted image distance-tree. *Bioinformatics*, 29(11):1448, 2013.

[153] Kwon Y, Kang M, Jeon YM, Lee S, Lee HW, Park JS, and Kim HJ. Identification and characterization of novel erbb4 variant associated with sporadic amyotrophic lateral sclerosis (als). *Journal of Neuroscience*, 457:122885, 2024.

[154] Guo-Cheng Yuan. *Computational Methods for Single-Cell Data Analysis*. Humana Press, 2019.

[155] Yao Z, Liu H, Xie F, Fischer S, Adkins RS, Aldridge AI, Ament SA, Bartlett A, Behrens MM, Van den Berge K, Bertagnolli D, de Bézieux HR, Biancalani T, Booeshaghi AS, Bravo HC, Casper T, Colantuoni C, Crabtree J, Creasy H, Crichton K, Crow M, Dee N, Dougherty EL, Doyle WI, Dudoit S, Fang R, Felix V, Fong O, Giglio M, Goldy J, Hawrylycz M, Herb BR, Hertzano R, Hou X, Hu Q, Kancherla J, Kroll M, Lathia K, Li YE, Lucero JD, Luo C, Mahurkar A, McMillen D, Nadaf NM, Nery JR, Nguyen TN, Niu SY, Ntranos V, Orvis J, Osteen JK, Pham T, Pinto-Duarte A, Poirion O, Preissl S, Purdom E, Rimorin C, Risso D, Rivkin AC, Smith K, Street K, Sulc J, Svensson V, Tieu M, Torkelson A, Tung H, Vaishnav ED, Vanderburg CR, van Velthoven C, Wang X, White OR, Huang ZJ, Kharchenko PV, Pachter L, Ngai J, Regev A, Tasic B, Welch JD, Gillis J, Macosko EZ, Ren B, Ecker JR, Zeng H, and Mukamel EA. A transcriptomic and epigenomic cell atlas of the mouse primary motor cortex. *Nature*, 598(7879):103–110, 2021.

[156] Yao Z, van Velthoven CTJ, Kunst M, Zhang M, McMillen D, Lee C, Jung W, Goldy J, Abdelhak A, Aitken M, Baker K, Baker P, Barkan E, Bertagnolli D, Bhandiwad A, Bielstein C, Bishwakarma P, Campos J, Carey D, Casper T, Chakka AB, Chakrabarty R, Chavan S, Chen M, Clark M, Close J, Crichton K, Daniel S, DiValentin P, Dolbeare T, Ellingwood L, Fiabane E, Fliss T, Gee J, Gerstenberger J, Glandon A, Gloe J, Gould J, Gray J, Guilford N, Guzman J, Hirschstein D, Ho W, Hooper M, Huang M, Hupp M, Jin K, Kroll M, Lathia K, Leon A, Li S, Long B, Madigan Z, Malloy J, Malone J, Maltzer Z, Martin N, McCue R, McGinty R, Mei N, Melchor J, Meyerdierks E, Mollenkopf T, Moonsman S, Nguyen TN, Otto S, Pham T, Rimorin C, Ruiz A, Sanchez R, Sawyer L, Shapovalova N, Shepard N, Slaughterbeck C, Sulc J, Tieu M, Torkelson A, Tung H, Valera Cuevas N, Vance S, Wadhwani K, Ward K, Levi B, Farrell C, Young R, Staats B, Wang MM, Thompson CL, Mufti S, Pagan CM, Kruse L, Dee N, Sunkin SM, Esposito L, Hawrylycz MJ, Waters J, Ng L, Smith K, Tasic B, Zhuang X, and Zeng H. A high-resolution transcriptomic and spatial atlas of cell types in the whole mouse brain. *Nature*, 624(7991):317–332, 2023.

[157] Yao Z, van Velthoven CTJ, Nguyen TN, Goldy J, Sedeno-Cortes AE, Baftizadeh F, Bertagnolli D, Casper T, Chiang M, Crichton K, Ding SL, Fong O, Garren E, Glandon A, Gouwens NW, Gray J, Graybuck LT, Hawrylycz MJ, Hirschstein D, Kroll M, Lathia K, Lee C, Levi B, McMillen D, Mok S, Pham T, Ren Q, Rimorin C, Shapovalova N, Sulc J, Sunkin SM, Tieu M, Torkelson A, Tung H, Ward K, Dee N, Smith KA, Tasic B, and Zeng H. A taxonomy of transcriptomic cell types across the isocortex and hippocampal formation. *Cell*, 184(12):3222–3241, 2021.

[158] M Zhang, GA Bouland, H Holstege, and MJT Reinders. Identifying aging and alzheimer disease-associated somatic variations in excitatory neurons from the human frontal cortex. *Neurology Genetics*, 9(3):e200066, 2023.

[159] Meng Zhang, Stephen Eichhorn, Brian Zingg, Zizhen Yao, Kaelan Cotter, Hongkui Zeng, Hongwei Dong, and Xiaowei Zhuang. Spatially resolved cell atlas of the mouse primary motor cortex by merfish. *Nature*, 598:137–143, 10 2021.