FACULTY OF ENGINEERING
AND ARCHITECTURE

Topological Inference in Graphs and Images

Robin Vandaele

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisors
Prof. Tijl De Bie, PhD* - Prof. Yvan Saeys, PhD**

* Department of Electronics and Information Systems
Faculty of Engineering and Architecture, Ghent University
** Department of Applied Mathematics, Computer Science and Statistics
Faculty of Sciences, Ghent University

December 2020

GHENT
UNIVERSITY

ISBN 978-94-6355-450-3
NUR 919, 958
Wettelijk depot: D/2020/10.500/127

Acknowledgements

"Well, why don't you do both?", is what my supervisor, Tijl De Bie, asked me when I told him I was still in doubt whether I should first do an additional Master in Statistical Data Analysis, rather than starting a PhD. Looking back to this, I don't think he will ask this a second time to someone else.

With a Master of Science in Mathematics from Ghent University, you are not qualified to directly enroll for a PhD in Computer Science at Ghent University. You have to enroll in an additional doctoral training program consisting of—brace yourselves—courses in mathematics. Yes. I was such a (proud) mathematician enrolled for a PhD in Computer Science, a doctoral training program, and a MSc in Stastical Data Analysis. I furthermore had the brilliant idea to start working on a topic on which neither I nor my supervisors knew anything about in advance. Last but not least, I combined this with multiple (and awesome) teaching responsibilities during the first three years of my research. Coming from a Master program where the only time I needed to use a computer was to write my thesis, it is safe to assume that it took some time—as in years—for my research finally took off. I'm not even sure if my 'plane' is fully airborne to this day.

The fact that I did not give up after what might be the roughest start of a PhD compared to all of my friends and colleagues, would never have been possible without having such a great mentor as Tijl. Hence, undoubtedly he deserves the first 'thank you' in this acknowledgments. Thank you Tijl, for your guidance, patience, kindness, and hospitality.

I furthermore would like to thank my other supervisor Yvan Saeys, who also stated one of my favorite quotes: "well, when is your sandwich ever not full?". Apart from his excellent guidance, he ensured the possibility of investigating many biological research topics and applications, which are among my favorite. I definitely look forward to our future collaborations.

I also thank the members of the examination board for accepting to be part of the jury, thoroughly reading and evaluating my work, and providing me with useful and critical feedback that had a noticeable positive impact on this thesis. I highly appreciate the time and effort you have all voluntarily put into this.

Naturally, my gratitude is not restricted to persons in my professional life. There is someone in my personal life who I would also like to thank deeply. Someone who chose to stick by me during my entire career as a PhD researcher, in good times and bad times. Someone who showed unconditional love during these past few months of working late, during the weekends, and holidays. Of course, I'm talking about my rabbit Smoef. It should be mentioned that the same applies to my girlfriend Sarah, although she appreciated the working late, during the weekends, and holidays a tad less.

I would also like to express my gratitude to many of my former and current colleagues at the AIDA research group (in alphabetical order): Ahmad, Alex, Bo, Dieter, Edith, Jefrey, Lemon, Len, Maarten, Paolo, Raphaël, Sander, Xi, and Yoosof, for the enjoyable times we shared and those that are yet to come. Similarly, I would like to thank all my many other former and current colleagues at the DaMBi research group, with a special thanks to Wouter Saelens and Robrecht Cannoodt for guiding me through my cell trajectory inference experiments, and Jonathan Peck, with whom I experienced great times both during my former life as a mathematics student, as well as during my time as a PhD researcher.

I furthermore thank some of my external colleague researchers, namely Olivier Gevaert who showed to be a great mentor during my stay at Stanford University and remains such to this day, Bastian Rieck with whom I can actually discuss topological data analysis in full mathematical detail, as well as Gert De Cooman and his entire research group with whom I experienced amazing times teaching and in Pizzeria La Rustica. Gert, if you are reading this thesis, note that it is important to realize that we consider 0 a natural number. Of course, this applies to anybody else reading this thesis as well.

A special thanks goes to my high school teacher in mathematics Jan Van Langenhove, who enriched me with the beautiful world of mathematics. He lies at the roots of my journey that started off by choosing to become a mathematician rather than an engineer. No regrets.

Finally, I deeply thank my (step)parents, family members, and friends, for their thorough support during this tremendous journey.

Oh wait! I forgot someone! I furthermore thank one of my dearest former colleagues, Florian Adriaens, with whom I experienced some of the most fun times during my time as a PhD researcher, such as during our trip to Alaska. Similar to how Bo always welcomed me with a pretty smile when I arrived at work around 8.30am, I welcomed Florian with the same smile when he arrived at noon.

Damn you coronavirus for making me miss these smiles.

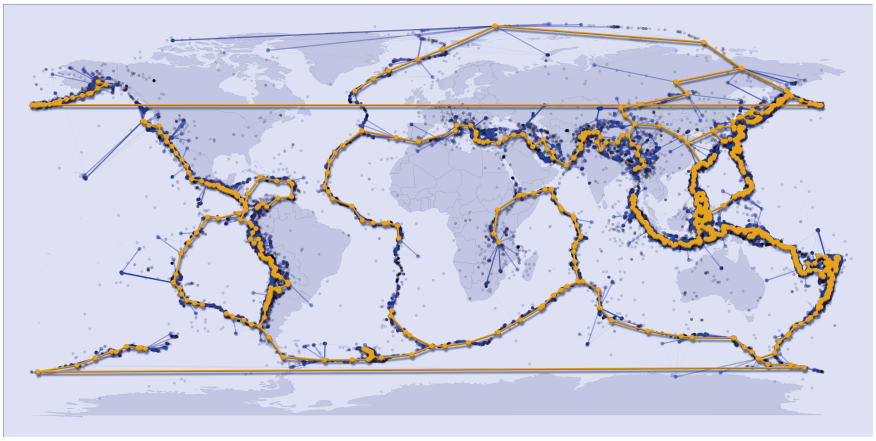
Gent, oktober 2020
Robin Vandaele

Samenvatting

Grafen zijn naar voren gekomen als krachtige datarepresentaties, van voor de hand liggende voorbeelden zoals sociale netwerken tot nabijheidsgrafen van hoog-dimensionale metrische data. Het begrijpen van de topologische structuren van deze grafen geeft ons cruciale inzichten in hun globale relationele eigenschappen. Ze leren ons hoe verschillende sociale gemeenschappen met elkaar verbonden zijn door middel van sleutelfiguren, hoe verschillende biologische cellen uit elkaar evolueren tijdens celdifferentiatie, of welke groepen knooppunten samenhangende objecten vormen in biomedische afbeeldingen.

Een eerste voorbeeld van een dergelijke topologische structuur vind je op de omslag van dit proefschrift,[1] alsook in Figuur 1, die de distributie van aardbe-

[1]Mogelijks ontbreekt de omslag in een online versie van dit proefschrift. In dit geval verwijzen we de lezer naar Figuur 1.



Figuur 1: (Nederlands). De meeste aarbevingen vinden plaats nabij de randen waar tektonische platen samenkomen, die een graafgestructureerd model vormen.

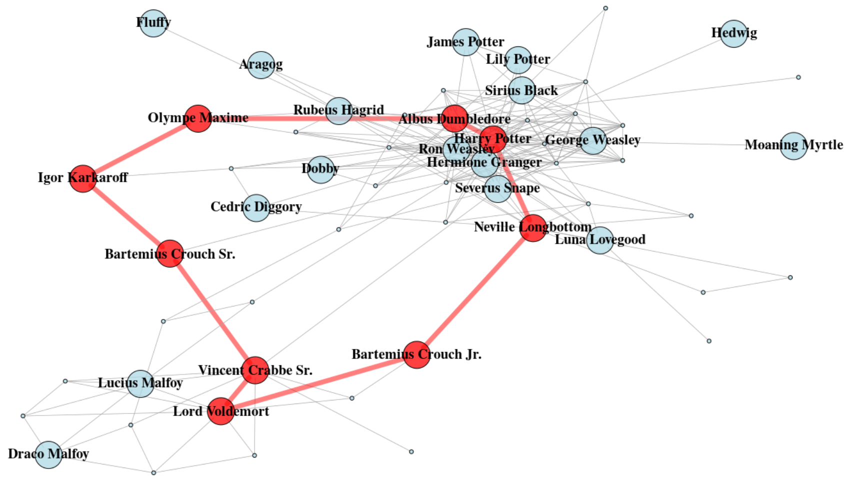
(English). Most earthquakes occur at the boundaries where tectonic plates meet, forming a graph-structured topology.

vingslocaties op aarde laat zien. De meeste aardbevingen vinden plaats nabij een grens tussen twee tektonische platen. Het zijn zelfs deze locaties die geologen helpen de grenzen van deze platen te bepalen. Deze grenzen kunnen worden weergegeven door verschillende lijnsegmenten op (een kaart van) de aarde, en samen vormen ze het *onderliggende* graafgestructureerd topologisch model van de aardbevingslocaties. Dit model wordt benaderd door de oranje graaf op de omslag van dit proefschrift.

Merk op dat dit formeel gezien geen topologisch model *in* een graaf is, maar eerder een graafgestructureerd model op zichzelf. Men kan echter eerst een bovenliggende graaf construeren uit de aardbevingslocaties, door elke aardbevingslocatie x met elk van diens 'naburige' aardbevingslocaties y te verbinden door middel van een boog $\{x, y\}$. Deze bogen kunnen formeel bepaald worden door middel van een afstandsmetriek tussen de aardbevingslocaties, bijvoorbeeld door middel van de sferische (geografische) afstanden op aarde, en een nabijheidsregel dat specificeert wanneer twee locaties (lokaal) dichtbij genoeg zijn volgens deze afstanden om verbonden te worden met een boog. Het onderliggende model van de originele data (de aardbevingslocaties), valt dan samen met het onderliggende model van dergelijke *nabijheidgraaf* geconstrueerd uit deze data. Zoals bovendien zal blijken uit dit proefschrift, is dit precies hoe we te werk zullen gaan voor het bepalen van graafgestructureerde modellen in *puntenwolken*, d.w.z. eindige metrische ruimten die overeenkomen met onze waargenomen data, zoals de aardbevingslocaties.

Merk op dat hier sprake is van nogal verwarrende terminologie: we beschouwen graafgestructureerde modellen in grafen, die als het ware zelf als graafgestructureerde modellen kunnen worden beschouwd. In dit proefschrift zullen we dit conceptueel duidelijk maken door middel van veel verschillende voorbeelden, waarvan de eerste al gegeven wordt op de omslag van dit proefschrift. Een van de belangrijkste doelen van dit proefschrift, alsook wat we als onze belangrijkste bijdrage beschouwen, is dan ook dat we op een overtuigende manier aantonen dat graafgestructureerde modellen in veel grafen voorkomen en een veel lagere topologische complexiteit (in termen van hoeveel knooppunten, cycli, bladeren, multifurcaties, . . . ,) hebben dan de originele bovenliggende graaf.

Een tweede voorbeeld wordt gegeven in Figuur 2. Hier zien we een graaf die weergeeft hoe verschillende karakters uit de Harry Potter-serie met elkaar verbonden zijn door middel van vriendschappelijke relaties. Elk personage kan nauw verbonden worden met, d.w.z. is hoogstens een paar vriendschappen verwijderd van, één van de hoofdpersonages, zoals Perkamentus, Voldermort of natuurlijk Harry Potter zelf. Verder zijn er ook directe en indirecte verbanden tussen deze hoofdpersonages. Het is waarschijnlijk voldoende dat u zich ervan bewust bent dat de serie bestaat, om te weten dat Harry Potter en Voldermort niet de beste vrienden zijn. Desalniettemin toonde Bartemis Crouch Jr. (niet de vriendelijkste van alle personages voor degenen die minder bekend zijn met de serie), die rechtstreeks verbonden is met Voldermort, een bijzondere interesse in Marcel Lubbermans (beslist een vriendelijker personage), die op zijn beurt rechtstreeks is verbonden met Harry door middel van een vriendschappelijke relatie. Het zijn precies die directe en indirecte verbanden tussen deze hoofdpersonages die het model vormen dat



Figuur 2: (Nederlands). Een circulair topologisch model (rood) in de Harry Potter graaf (blauw), die vrienschappelijke relaties tussen personages uit de sage weergeeft. De layout van de graaf is gegeven door een krachtgestuurd layout algoritme.

(English). A circular topological model (red) in the Harry Potter network (blue), which displays friendly relationships between characters of the saga. The graph layout is provided through a force-directed layout algorithm.

ten gronde ligt aan de originele graaf, hier benaderd door de rode graaf in Figuur 2. Niettemin hoeven niet alle hoofdpersonages noodzakelijkerwijs in dit model te worden opgenomen. Zo zijn bijvoorbeeld Harry Potter, Ron Wemel en Hermelien Griffel zo nauw met elkaar verbonden, dat één van hen (zoals Harry) voldoende is om dit cluster van personages in het model te representeren.

Merk op dat we in tegenstelling tot het voorgaande voorbeeld van aardbevingslocaties, nu een expliciet gegeven graaf beschouwden en het dan ook ingewikkelder is om over een continue versie van dit model spreken, of over een extrinsieke ruimte waarin het zich bevindt. Bovendien zijn de aardbevingslocaties een gevolg van hun onderliggend graafgestructureerd model, d.w.z. de verdeling van aardbevingen is het resultaat van de positionering van de tektonische platen (en de wrijving daartussen) en niet andersom. Het is voor discussie vatbaar of soortgelijk causaal verband tussen het 'model' en de 'data' geldt voor de Harry Potter graaf, waarbij J.K. Rowling, de schrijfster van de serie, eerst de globale relaties tussen 'goed' en 'kwaad' zou kunnen hebben vastgelegd, om pas nadien de karakters hier omheen te plaatsen (onwaarschijnlijk met grafentheorie in gedachten). Echter, voor een groot aantal andere realistische grafen zou dergelijke analoge interpretatie minder steek houden. Een goed voorbeeld hiervan zal geïllustreerd worden in Hoofdstuk 3, door middel van de *Karate graaf*, een welbekende graaf binnen het domein van graafdelving. Zoals we in Hoofdstuk 3 zullen bespreken, kan deze graaf eenvoudig gemodelleerd worden door middel van een lineaire verbinding tussen twee afzonderlijke gemeenschappen. Echter, gezien dit een voorbeeld is van een volledig non-fictieve graaf—waarbij we filosofische vragen zoals "of er goddelijke entiteiten aan het werk zijn" en "of we allemaal in een gigantische computersimulatie leven, waarin we een virtuele wereld in Matrix-stijl ervaren waarvan we denken dat die echt is" achterwege laten—was dit lineaire model in de Karate graaf niet aanwezig vóór de werkelijke data, d.w.z. de entiteiten in de graaf. Doorheen dit proefschrift bespreken we dan ook dat vereenvoudigde graafgestructureerde modellen van nature voorkomen in veel realistische grafen, maar omgekeerd zijn niet al deze grafen het causaal gevolg van een dergelijk model.

In het eerste en grootste deel van dit proefschrift (hoofdstukken 3-6) bestuderen we het probleem van topologische inferentie van graafgestructureerde modellen *in* grafen, hetgeen overeenkomt met het identificeren van de modellen zoals we hierboven besproken hebben. Daartoe onderzoeken we bestaande en ontwikkelen we nieuwe methoden binnen het opkomende gebied van de *topologische data-analyse* (TDA). Merk op dat ons gebruik van de term TDA moet worden geïnterpreteerd in de zin dat we de onderliggende 'vorm' van onze data bestuderen, zonder ons daarvoor noodzakelijk te beperken tot methoden uit de topologie. Dergelijke methoden worden evenals gecategoriseerd binnen (of worden soms zelfs beschouwd als een synoniem voor) TDA, zelfs wanneer hun einddoel niet specifiek topologische inferentie is. Inderdaad, de meest prominent gebruikte en bestudeerde methode binnen TDA onder de naam van *persistente homologie*—die ook een belangrijke rol zal spelen in meerdere van onze hoofdstukken—heeft recent geleid tot veel nieuwe toepassingen binnen machinaal leren, zoals voor regressie- en classificatieproblemen. Echter, in dit proefschrift zullen we ons niet beperken tot louter methodes uit de topologie om topologische inferentie uit te voeren, maar zullen we hiervoor ook een groot aantal methodes uit de grafentheorie en optimalisatie bestuderen en ontwikkelen.

Doorheen dit proefschrift zullen we ons in het bijzonder focussen op directe toepassingen van topologische inferentie in grafen voor *celtraject-inferentie* (CTI), hetgeen formeel zal worden geïntroduceerd in Hoofdstuk 3. Hier is het de taak om het graafgestructureerd model dat het differentiatieproces voorstelt dat biologische cellen ondergaan tijdens cellulaire ontwikkelingen, te bepalen . Denk bijvoorbeeld aan gezonde cellen die plotseling evolueren tot kankercellen, of immuuncellen die zich moeten aanpassen om ons lichaam te beschermen tegen nieuwe interne gevaren. Het meten van de gen- of eiwitexpressie van vele individuele cellen op een bepaald moment en positie in een dergelijk differentiatieproces, leidt tot hoogdimensionale puntenwolkgegevens (elk punt komt overeen met één cel) waarbij elke dimensie overeenkomt met de expressie van een bepaald gen of eiwit. Net als bij de aardbevingslocaties die we hierboven beschreven, is ons doel om het model dat het biologische differentiatieproces van de cellen voorstelt te bepalen uit een nabijheidsgraaf opgebouwd uit deze (in dit geval hoogdimensionale) data. Het zal echter blijken dat dit een enorm moeilijk probleem is, met veelal slechte resultaten in de praktijk, zowel voor de methoden die wij zullen ontwikkelen alsook de andere huidige methoden die hiervoor worden toegepast. Een belangrijk onderdeel van ons proefschrift is dan ook dat we de inherente moeilijkheden waarmee CTI wordt geconfronteerd in een wiskundig geformaliseerde manier zullen beschrijven en aantonen. Dit is het onderwerp van Hoofdstuk 6.

In het tweede deel van ons proefschrift (Hoofdstuk 7) bestuderen we het probleem van topologische inferentie in afbeeldingen. Zoals we zullen bespreken, kan dit wiskundig gezien ook als een 'topologische inferentie in grafen'-probleem worden beschouwd. We maken dit onderscheid dan ook vooral voor de duidelijkheid. Echter zullen we geen graafgestructureerde modellen afleiden in dit deel, maar meer bepaald goed gedefinieerde 2D-vormen en objecten die worden weergegeven door de afbeelding. Daartoe breiden we de toepasbaarheid van persistente homologie uit tot realistische afbeelding, door middel van een casestudy waarin we biomedische huidlaesiebeelden beschouwen binnen de context van ongesuperviseerde segmentatie.

Onze belangrijkste bijdragen in dit proefschrift (en het werk dat ertoe geleid heeft,) zijn als volgt.

- Zoals hierboven vermeld, beschouwen we het feit dat we op een overtuigende manier aantonen dat topologische modellen in veel verschillende grafen voorkomen als de belangrijkste bijdrage van dit werk. Ons werk zal daarom vergezeld zijn van vele en grondige visualisaties van dergelijke modellen, met toepassingen die variëren van simplistische voorbeelden zoals het modelleren van onze geliefde vriend Pikachu, tot meer serieuzere toepassingen zoals sociale netwerken en CTI.

- We voorzien een autonome inleiding tot de gebieden van grafentheorie, topologie en TDA, in Hoofdstuk 2.

- We voorzien een inleiding tot topologische inferentie in grafen en bespreken de moeilijkheden met betrekking tot het wiskundig formaliseren van dit probleem in Hoofdstuk 3.

- We ontwikkelen en bestuderen een methode voor topologische inferentie uit nabijheidsgrafen (meer specifiek uit *Rips-grafen* opgebouwd uit puntenwolkgegevens) gebaseerd op locale topologische informatie (Hoofdstuk 4).

- We ontwikkelen en bestuderen een methode voor topologische inferentie uit meer algemeen gegeven grafen met behulp van tussenliggende *woudrepresentaties*, waarvoor we ook het concept van een *f-den* introduceren (Hoofdstuk 5).

- We introduceren een nieuwe functie geëvalueerd op knopen in grafen, genaamd de *grenscoëfficiënt*, die intuïtief de binnen- en buitenkant van een gegeven graaf aangeeft door middel van meetkundige informatie (Hoofdstuk 5).

- We introduceren een nieuw optimalisatieprobleem in grafen voor topologische inferentie door middel van woudrepresentaties. Verder leiden we veel belangrijke theoretische resultaten af uit dit probleem, waaruit blijkt dat deze enorm interessant is vanuit zowel een wiskundig als computationeel standpunt (Hoofdstuk 5).

- We tonen dat zelfs de meest performante onder de huidige CTI-methoden nog steeds slecht presteren voor veel celtraject-datasets, of het aantal bladeren, d.w.z. de begin- en eindstadia, binnen boomgestructureerde biologische differentiatieprocessen onderschatten (Hoofdstuk 5). Verder verantwoorden we het gebruik van een topologische afstandsmetriek om deze moeilijkheden in een wiskundig geformaliseerde manier te beschrijven (Hoofdstuk 6).

- We breiden de toepasbaarheid van TDA uit tot ongesuperviseerde objectdetectie in realistische afbeelding door middel van *topologische beeldmodificatie*. Alsook ontwikkelen we een methode die dergelijke topologische informatie gebruikt voor *topologische beeldverwerking*, waarbij de topologische objecten van belang worden gemarkeerd op de afbeelding. We tonen zowel kwalitatief als kwantitatief aan dat topologische beeldverwerking in staat is om verschillende methoden voor ongesuperviseerde binaire segmentatie, oversegmentatie, alsook randdetectie te verbeteren, door middel van een casestudy van biomedische huidlaesiebeelden (Hoofdstuk 7).

- Ten slotte en gerelateerd aan onze eerstgenoemde bijdrage, door aan te tonen dat topologisch modellen voorkomen in enorm veel verschillende grafen, ongeacht of ze gegeven zijn of afgeleid uit puntenwolken, stimuleren we nieuw onderzoek naar het formaliseren van deze modellen, nieuwe en effectievere methoden om deze te infereren, alsook nieuwe toepassingen van deze modellen zoals visualisatie, graafinbedding en graaf-neurale-netwerken. We bespreken deze onderwerpen in detail als toekomstig werk en geven alvast enkele concrete aanwijzingen hiervoor (Hoofdstuk 8).

Summary

Graphs have emerged as powerful representations of data, from obvious examples such as social networks, to proximity graphs of high-dimensional metric data. Understanding the *topological structures* of these graphs provides us crucial insights into their global relational properties. They teach us how different social communities are connected through key figures, how different biological cells evolve from each other during cell differentiation, or which groups of nodes form coherent objects in biomedical images.

A first example of such topological structure is illustrated on the cover of this thesis,[2] as well as in Figure (Figuur) 1 in the Dutch summary of this thesis above, which shows the distribution of earthquake locations on the Earth. Most earthquakes tend to occur at the boundaries where the tectonic plates meet. As a matter of fact, the locations of earthquakes actually help geologists to define the boundaries of these tectonic plates. These boundaries can be displayed by various line segments on (a map of) the Earth, and together, they form the *underlying* graph-structured topological model of earthquake locations. This model is approximated by the orange graph on the cover of this thesis.

Note that this is not formally a topological model *in* a graph, but rather a graph-structured model itself. However, one may first construct a superimposed graph from the earthquake locations, by connecting each earthquake location x to each one of its 'neighboring' earthquake locations y by an edge $\{x, y\}$. These edges may formally be defined through a distance measure between the earthquake locations, e.g., the great-circle (geographic) distances, and a neighborhood rule that specifies when two locations are (locally) close enough according to these distances to be connected by an edge. The underlying model of the original data (the set of earthquake locations), then coincides with the underlying model of such a *proximity* graph constructed thereof. Moreover, as we will show in this thesis, this is exactly how we proceed for inferring graph-structured models in *point clouds*, i.e., finite metric spaces corresponding to our observed data, such as the earthquake locations.

Remark the rather confusing mix of terminology here: we are considering graph-structured models *in* graphs, which can be regarded as graph-structured mo-

[2]Possibly the cover is missing from an online version of this thesis. In this case we refer the reader to Figure (Figuur) 1.

dels themselves. Throughout this thesis, we will make this conceptually clear by means of many different examples, the first of which is already illustrated on the cover of this thesis. Indeed, one of the main purposes of this thesis, as well what we consider our main contribution, is that we convincingly show that graph-structured models occur in many graphs, and have a far lower topological complexity (in terms of how many nodes, cycles, leaves, multifurcations, ...,) than the original superimposed graph.

A second example is given in Figure (Figuur) 2 in the Dutch summary. Here, we are given a graph that captures how different characters from the Harry Potter series are linked to each other by means of friendly relations. Every single character can be closely connected to, i.e., is at most a few friendships away, from one of the main characters, such as Dumbledore, Voldermort, or of course, Harry Potter himself. Furthermore, there are also direct and indirect connections between these main characters. E.g., it is likely to be sufficient that you are aware that the series exists, to know that Harry Potter and Voldermort are not the dearest friends. Nevertheless, Bartemis Crouch Jr. (not the kindest of all characters for those who are less familiar with the series), who is directly connected to Voldermort, showed a particular interest in Neville Longbottom (definitely a more kind character), who in turn is directly connected to Harry through friendship. It are exactly those direct and indirect links between these main characters that make up the topological model underlying the original graph, in this case, represented by the red graph in Figure (Figuur) 2. Nevertheless, not all main characters must necessarily be included in this model. E.g., Harry Potter, Ron Weasley, and Hermione Granger are all so closely connected to each other, that one of them (such as Harry) suffices to represent this cluster in the model.

Observe that unlike the former example of earthquake locations, we now considered a directly given graph, and it is far more complicated to talk about a continuous version of this model, or an extrinsic space in which it resides in. Furthermore, the earthquake locations are a consequence of their underlying graph-structured model, i.e., the distribution of earthquakes is a result from the positioning of the tectonic plates (and the friction between them), and not vice versa. Arguably, a similar causal relationship between the 'model' and 'data' may hold for the Harry Potter network, where J.K. Rowling, the writer of the series, might have derived the global high-level relationships between 'good' and 'evil' first, and only then placed the actual characters around it (although unlikely keeping graph theory in mind). However, for a wide variety of other real-world graphs, such an analogous interpretation would make far less sense. A good example of this will be illustrated in Chapter 3 through the *Karate network*, a well-known graph within the field of graph mining. As we will discuss in Chapter 3, this graph can be modeled well by a linear connection between two separate communities. However, as this is an example of a real-world graph—neglecting philosophical questions such as "whether there are any divine entities at work" or "whether we are all living inside a gigantic computer simulation, experiencing a Matrix-style virtual world that we think is real"—this linear model in the Karate network was not present before the actual data, i.e., the entities in the graph. Hence, throughout this thesis, we argue that simplified graph-structured models occur naturally in many real-world graphs, but conversely, not all of these graphs are the causal result of such model.

In the first and major part of our thesis (Chapters 3-6), we consider the problem of topological inference of graph-structured models *in* graphs, which coincides with inferring the models as discussed above from our data. To this end, we investigate existing and develop new methods within the rising field of *topological data analysis* (TDA). Note that our use of the term TDA should be interpreted in the sense that we are studying the underlying 'shape' of our data, without necessarily restricting to techniques from topology for this purpose. These techniques are also categorized into (or sometimes even considered synonym to) the field TDA, even though their end goal may be different from actual topological inference. Indeed, the most prominently used and studied method within TDA under the name of *persistent homology*—which will also play a major role in multiple of our chapters—has led to many recent advantages within general machine learning problems such as regression and classification. However, we will not restrict to purely techniques from topology for performing topological inference throughout this thesis, but will also study and develop a vast amount of techniques from graph theory and optimization for this purpose.

Throughout this thesis, we will particularly focus on direct applications of topological inference in graphs within the field of *cell trajectory inference* (CTI), which will be formally introduced in Chapter 3. Here, the task is to infer the graph-structured model that represents the differentiation process the biological cells undergo during cellular development. Think of normal cells that suddenly evolve into a cancer cells, or immune cells that need to adapt to protect our body against new internal threats. Measuring the gene or protein expression of many individual cells at a particular time and point in such a differentiation process, leads to high-dimensional point cloud data (each point corresponding to one cell) where each dimension corresponds to the expression of one particular gene or protein. Similar to the earthquake locations above, the model representing the biological differentiation process of the cells is then to be inferred from a proximity graph constructed from this (in this case high-dimensional) data. It turns out that this is a very difficult problem, with often poor results in practice, both for the methods we will develop as well as for the state-of-the-art. An important part of our thesis will hence be explaining the inherent difficulties CTI is confronted with in a mathematically formalized manner. This is the topic of Chapter 6.

In the second part of our thesis (Chapter 7), we consider the problem of topological inference in images. As we will discuss, this can mathematically be considered to be a problem of topological inference in graphs as well, and we make this distinction mainly for clarity. However, we will not be inferring graph-structured models in this part, but rather well-defined 2D shapes and objects that are displayed by the image. To this end, we will extend the applicability of persistent homology to real-world images, by considering a case-study of biomedical skin lesion images within an unsupervised segmentation context.

Our main contributions in this thesis (and the work that led to it) are as follows.

- As mentioned above, what we consider the main contribution of our work is that we convincingly show that topological models occur in many different types of graphs. One will therefore find our work accompanied by many and thorough visualizations of such models, with applications ranging from toy examples such as modeling our beloved friend Pikachu, to more serious applications such as social networks and CTI.

- We provide a self-contained introduction to the fields of graph theory, topology, and TDA in Chapter 2.
- We provide an introduction to topological inference in graphs, and discuss the difficulties in terms of providing a mathematical formalization of this problem (Chapter 3).
- We develop and study a method for topological inference from proximity graphs (more specifically from *Rips graphs* constructed from point cloud data) based on local topological information (Chapter 4).
- We develop and study a method for topological inference from general given graphs by using intermediate *forest representations*, for which we also introduce the concept of an f*-pine* (Chapter 5).
- We introduce a new vertex-valued function termed the *boundary coefficient* (BC), that intuitively marks the inside and the outside of a given graph based on geometrical information (Chapter 5).
- We introduce a new graph optimization problem termed ***C**onstrained **L**eaves **O**ptimal sub**F**orest* (CLOF) for performing topological inference through forest representations. Furthermore, we derive many important theoretical results from this problem, showing that CLOF is highly interesting from both a mathematical as well as a computational perspective (Chapter 5).
- We show that even the highest ranked CTI methods still lead to a low performance on many cell trajectory data sets, or commonly underestimate the number of leaves, i.e., the start and end states, within tree-structured biological differentiation processes (Chapter 5). We furthermore justify the usage of a topological distance metric to explain these difficulties in a mathematically formalized manner (Chapter 6).
- We extend the applicability of TDA to improve unsupervised object detection in real-world images through *topological image modification* (TIM). We furthermore develop a method that uses such topological information for *topological image processing* (TIP), highlighting the topological objects of interest on the image. We qualitatively and quantitatively show that TIP improves various unsupervised methods for binary segmentation, oversegmentation, and edge detection, through a case-study of biomedical skin lesion images (Chapter 7).
- Finally, and related to our first stated contribution, by showing that topological models occur in a wide variety of graphs, whether given or derived from point clouds, we aim to stimulate new research into how to formalize these models, more effective methods for inferring these, as well as new applications of these models such as visualization, graph embedding, and graph neural networks. We discuss these topics more concretely as future work, and provide some first directions to these (Chapter 8).

Acronyms

BC	Boundary Coefficient
CLOF	Constrained Leaves Optimal subForest
CTI	Cell Trajectory Inference
GCN	Graph Convolutional Network
k**NN**	k Nearest Neighbor
LCC	Local Cluster Coefficient
LTDA	Local Topological Data Analysis
MDS	Multidimensional Scaling
MST	Minimum Spanning Tree
PCA	Principal Component Analysis
TDA	Topological Data Analysis
TIM	Topological Image Modification
TIP	Topological Image Processing

Publications

The contents of this thesis is based on, but not limited to, the results reported in the following papers.

Articles in Journals

- Robin Vandaele, Yvan Saeys, Tijl De Bie. *Mining Topological Structure in Graphs through Forest Representations.* Journal of Machine Learning Research, 21(215):1–68, 2020. [1]
- Robin Vandaele, Guillaume Adrien Nervo, and Olivier Gevaert. *Topological image modification for object detection and topological image processing of skin lesions.* Scientific Reports, 10(1):21061, 2020. [2]
- Robin Vandaele, Bastian Rieck, Yvan Saeys, and Tijl De Bie. *Stable Topological Signatures for Quantifying Patterns through Graph Approximations of Metric Trees.* Submitted to Pattern Recognition Letters, 2020. (Under revision)

Articles in Archived Proceedings

- Robin Vandaele, Tijl De Bie, and Yvan Saeys. *Local Topological Data Analysis to Uncover the Global Structure of Data Approaching Graph-Structured Topologies.* In Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, and Georgiana Ifrim, editors, Machine Learning and Knowledge Discovery in Databases, pages 19–36, Cham, 2019. Springer International Publishing. [3]

Articles in non-Archived Proceedings

- Robin Vandaele, Yvan Saeys, and Tijl De Bie. *The Boundary Coefficient: a Vertex Measure for Visualizing and Finding Structure in Weighted Graphs.* In Proceedings of the 15th International Workshop on Mining and Learning with Graphs (MLG), 2019. [4]

Contributions Outside of this Thesis

A particular type of topological structures, termed *metric graphs*, will play an important role throughout this thesis. The task of *cell trajectory inference* (CTI) is to infer such models from high-dimensional expression data, where each observation corresponds to a cell at some time during a biological differentiation process, and each feature (dimension) corresponds to how much a particular gene or protein is expressed by such cell. As we will see in Chapter 5, even the highest ranked state-of-the-art methods for performing CTI still struggle to infer the models in many cases. This will then be further explored in Chapter 6 in a mathematically formalized manner.

These observations also led to new contributions that are not contained within this thesis. First, we studied in which way we can and cannot extend the existing theoretical results for preserving geodesic distances on smooth manifolds [5], to the case of metric graphs. These results led to a recent paper, that is planned for presentation during the International Conference on Pattern Recognition (2020), which will be published in archived proceedings:

- Robin Vandaele, Tijl De Bie, and Yvan Saeys. *Graph Approximations to Geodesics on Metric Graphs.* In The International Conference on Pattern Recognition (ICPR), 2020.

We then continued this research, by providing probabilistic results that state how likely one will capture geometrical properties of metric graphs through Rips or kNN graph representations. The results are based on geometrical characteristics of the model relative to the (Euclidean) space it resides in. We furthermore studied and discussed the effect of two different types of noise, i.e., low-dimensional and high-dimensional, on the difficulty of this problem, connecting the latter to known phenomena that fall under the curse of dimensionality. We concluded that—as one would also intuitively expect—having a good representation of the data in the Euclidean space that places regions that are far apart in the model far apart in the Euclidean space as well (which is one of the main purposes of dimensionality reduction methods) increases the likeliness of one to infer true geometrical properties of the model through Rips or kNN graph representations. Furthermore, we studied the use of persistent homology to provide topological signatures that allows us to evaluate and compare different cell trajectory data representations (for varying dimensionality reduction methods or varying proximity graph constructions) in terms of how well they capture particular topological properties of interest (components, cycles, or leaves). Note that this study was partially based on the results from Chapter 6 of the current thesis, where we discuss these signatures in particular for leaves. This research was then presented through our recent Master thesis in Statistical Data Analysis:

- Robin Vandaele. *Topological Data Analysis of Metric Graphs for Evaluating Cell Trajectory Data Representations*. Master's thesis, Ghent University, 2020. [6]

Table of Contents

1 Introduction

The recent field of *topological data analysis* (TDA) aims to study the *shape of data*. However, real world data is often accompanied by the presence of noise and outliers. Furthermore, it is algorithmically difficult or even impossible to classify topological spaces up to a *homeomorphism* (formally defined in Chapter 2) [7, 8], and hence, through empirical data derived thereof. This limits the applications of TDA, notably to graphs and images, in an unsupervised setting.

Nevertheless, topological information and models can provide us crucial insights into such data. In biology, graphs inferred from high-dimensional cell trajectory data model the dynamic changes immune cells undergo to protect our body against environmental and internal threats [9]. In geoinformatics, inferring graphs from GPS coordinates allows one to obtain up-to-date road maps, which are critical for many applications, such as GPS-based navigation services and autonomous transportation [10]. In social sciences, graphs allow one to model how different communities are connected, and identify which figures play a key role in these connections [11]. In case of biomedical skin lesion images, extracting groups or clusters of nodes that mark well-defined topological objects on the image allow fast and automatic segmentation of the lesions.

We therefore study, extend, and develop new and existing methods within the field of TDA for unsupervised learning, exploratory data analysis, and topological inference in graphs and images. To this end, we develop the necessary theory and algorithms, and include thorough verification on both artificial and real world data.

1.1 Outline

Literature Overview: Topological Data Analysis and Graphs [Chapter 2] We start by providing a self-contained, yet basic introduction to topology, graphs (also called networks), and TDA. This chapter serves as a background chapter and includes the necessary material, definitions, and results that will be used in the rest of the thesis. Note that this chapter also touches on many aspects that may not be required to fully understand all problems we consider and methods we develop throughout this thesis. However, this chapter is of value to anyone who is interested in, but unfamiliar with the topic of TDA, and therefore can be considered as a valid contribution on its own (even though none of the material presented in this chapter is novel). Indeed, at the time of writing, the majority of the machine learning community is still unaware of the wide variety of applications in which TDA finds usage, or even unaware of the entire field of TDA. [1]

To help one decide whether one may skip (parts of) this chapter, we summarize its key observations and material below.

- Section 2.2 contains a very brief introduction to *category theory*, providing two of its most important definitions, i.e., *categories* and *isomorphisms*. Category theory can be regarded as the branch of mathematics that formalizes mathematical structures in a general setting. E.g., what are the *homeomorphisms* of *topological spaces*, and what are the *isometries* of *metric spaces*, can all be considered as isomorphisms within their respective categories. Nevertheless, category theory is rather unimportant throughout this thesis, and this section mainly serves anyone interested in more fundamental research in TDA.

- Section 2.3 summarizes the basic concepts of topology and metric spaces. The included definitions (e.g. a topological space in terms of open sets) are often more complicated than the key ideas we wish to present. The main takeaways of this section are as follows. Topology studies the properties of geometric objects that are preserved under continuous deformations, such as stretching, bending, and twisting. The term *homeomorphism* refers to a bijective map between geometric objects that preserves all such topological properties. It is important to realize that topology is often a rather weak characterization of structure. E.g., a coffee mug and a donut are topologically equivalent, and so are any finite metric spaces (point clouds) of the same size. Topological properties are therefore often much weaker than *geometrical properties*: although the spaces above may not be topologically distinguished, they might based on a metric (distance) defined on them. The

[1] As was I before I started my PhD.

term *isometry* refers to a bijective map between geometric objects that preserves all distances, and how close two such objects are to being isometric is expressed through the *Gromov-Hausdorff* distance between them.

- Section 2.4 presents the necessary definitions from graph theory, as well as of two *proximity graphs* we often consider throughout this thesis, termed the *Rips graph* (also known as the *unit disk graph*) and the kNN graph. We furthermore define *metric graphs*, which can be regarded as continuous drawings of graphs in some Euclidean space $\mathbb{R}^d$, and which will be the topological models underlying all point cloud data considered in this thesis.

- Section 2.5 provides an introduction to *persistent homology*, which may be considered the main tool of TDA. The first parts of this section (Sections 2.5.1-2.5.2.1) present fundamental results within the field of algebraic topology in which persistent homology finds its roots, and how these relate to its computation. Again, these parts mainly serve anyone interested in more fundamental research in TDA, and someone unfamiliar with such abstract mathematics might find them more challenging (yet perhaps, highly interesting).

  The following parts (Sections 2.5.2.2-2.5.2.4) illustrate the purpose of persistent homology and what it computes, i.e., *persistence diagrams*, through a variety of clear examples. For this thesis, it is more important to understand what persistent homology computes, rather than the actual results from algebraic topology on which is founded. Hence, we ensured that Sections 2.5.2.2-2.5.2.4 are sufficiently self-contained for this purpose.

  Another important takeaway of Section 2.4 is that the information encoded through (persistent) homology is generally weaker than the information encoded through topology, even though the latter was already relative weak (e.g., when compared to geometry). Neither topology, nor homology can distinguish between a coffee mug and a donut. However, unlike homology, topology can e.g. distinguish between a circle and a cylinder, which have distinct intrinsic dimensions. Fortunately, unlike topology, homology does admit effective computation and comparison algorithms.

- Section 2.6 presents the *mapper* algorithm, another extensively studied and applied method within the field of TDA, that furthermore strongly connects to graphs as well. We mainly include it because of its importance within TDA, but also use it for a baseline comparison in Chapter 4. For our purpose, it is more important to understand the intuitive working of the algorithm (illustrated in Figure 2.14), rather than the theory behind it, which we mainly include for completeness.

- Finally, Section 2.7 discusses the concept of a *merge tree*, which is studied much less commonly than persistent homology and the mapper algorithm by the current TDA community. The only reason we include it, is because the concepts and results from this section will play an important role in a proof of one of our results, i.e., Theorem 6.3.1 in Chapter 6. Apart from this result, the material in Section 2.7 will not be used in this thesis.

Introduction to Topological Models in Graphs [Chapter 3] In this chapter, we first present the problem of topological inference from and of graphs on an intuitive level. We will show this through an example of the *Karate network*, a well-known graph within the field of graph mining and analysis [12]. On both an intuitive and visual level, a *linear graph model* will fit this network well (Section 3.2). We discuss why it is however difficult to provide a theoretical formalization of this observation (Section 3.4). Indeed, we emphasize that

simplified graph-structured models occur naturally in many real-world graphs, but conversely, many graphs are not the causal result of such model.

Hence, although a linear graph model fits the Karate network well, this does not necessarily mean that the graph itself is a result of a linear graph model.

In contrast to this, the field of *cell trajectory inference* (CTI) considers high dimensional point cloud data that is derived from ground-truth graph-structured models. One of the most important applications we will consider multiple throughout this thesis is to infer these models from *proximity graphs* (Section 2.4.1) constructed from this data. Hence, a similar to all proximity graphs constructed from point cloud data throughout this thesis, these graphs will have (and result from) a ground truth graph-structured model, as we discuss in Section 3.3.

Models from Local Topological Data Analysis [Chapter 4] In this chapter, we present methods that reconstruct a global *non-subgraph model* from metric data based on local topological information obtained through *local topological data analysis* (LTDA) [3]. We briefly discuss their connection to *persistent local homology* in Section 4.2. These methods are inspired on the fact that for *metric graphs* (Section 2.4.2), it suffices to know the *degree* (Section 2.4.2) of each point locally everywhere to be able to reconstruct the entire topology. Even more than this, it suffices to know whether these degree are different from 2 or not [13]. These properties hold on a theoretical level, and are inferred in a (metric) data setting through clustering algorithms. In Section 4.3, we show how these are used to infer degrees and hence multifurcations underlying metric data. Furthermore, in Section 4.4, we show how we can use a similar procedure to infer the presence of cycles without the need of *1-dimensional (persistent) homology* (Section 2.5). In Section 4.5, we introduce a reconstruction algorithm that uses this inferred information to reconstruct the entire global model. We also discuss how storing and using inferred degrees improves our method over an existing reconstruction method that only classifies nodes according to their degree being different from 2 or not [13]. We evaluate our method for various synthetic and real-world metric data throughout this entire chapter, and conclude upon our work in Section 4.6.

Inferring Topological Models through Forest Representations [Chapter 5] In this chapter, we present a completely different approach towards inferring graph-structured models in graphs. These models will be subgraphs—which we term *backbones*—of the original given graphs. Unlike the reconstruction methods discussed in Chapter 4, our backbone inference method will be based on the assumptions that real world graphs contain 'noise' and 'outliers', instead of properties of the underlying ground truth model (if there even is such a model). Note that the concepts of noise and outliers may not be well-defined in any given graph. However, in terms of backbones, there interpretation will be intuitively the same as for point cloud data, being that they surround the underlying model of the graph.

We introduce the *boundary coefficient* (BC) [4] for quantifying core nodes in a graph in Section 5.2. This coefficient is used to construct a *forest representation*—in our case termed an f*-pine* [4]—of the original graph from which we can much more efficiently infer the backbone. This inference is performed by solving the Constrained Leaves Optimal subForest (CLOF) problem, which we present in Section 5.4. We summarize how these three puzzle pieces, being the BC, f-pines, and CLOF, fit together and form an effective method for inferring backbones in given graphs (whether metric or non-metric) in Section 5.5. We also show how we can regard our method as a facility location problem in networks, and discuss its advantages over existing such methods in Section 5.6. In Section 5.7, we qualitatively and quantitatively confirm the applicability, effectiveness, and scalability of our method for discovering backbones in a variety of graph-structured data, such as social networks, earthquake locations scattered across the Earth, and high-dimensional cell trajectory data. We furthermore focus on the use of our method for the particular case of CTI in Section 5.8. We show it is competitive with the state-of-the-art through a large scale analysis of 333 cell trajectory data sets, even though our method was not particularly designed for this task. We conclude upon the work in this chapter in Section 5.9.

Topological Signatures through Graph Approximations [Chapter 6] In this chapter, we build further upon our conclusions from Section 5.8. More specifically, in Section 5.8 we will show that accurately performing CTI is a difficult problem to this day, and that high-ranked state-of-the-art approaches still struggle to infer cell trajectories in many data sets, as well as to correctly infer the number of leaves, i.e., start or end states, of the model. The purpose of Chapter 6 is hence to provide theoretical and practical insights into why this is the case. However, the results in this chapter extend to any other application considering *metric trees* (Section 2.4.2).

In Section 6.2 we point out the difference between preserving geometrical and topological properties further, which we initially started in Section 2.3. Although preserving geometry is much stronger than preserving topology, the former allows some form of quantization under which we fail to do this, through the *Gromov-Hausdorff* distance (Definition 2.3.14). This means that we can empirically approximate the geometry of our model arbitrarily well, even when our inferred topology is incorrect. In Section 6.3, we build upon this fact and provide *topological signatures* for metric trees that capture the underlying topological properties of these models well whenever we compute these signatures from graph approximations that preserve the geometry well. In Section 6.4, we use these signatures to provide a charting of cell trajectory data sets with an underlying metric tree model, through which we confirm our observations in the field of CTI on both a theoretical and practical level, providing novel insights into this field. We furthermore explain how our signatures lead to a new method for quality control within the field of CTI. We conclude upon the work in this chapter in Section 6.5.

Topological Object Detection in Images [Chapter 7] This Chapter—which came to existence through a three month research visit to the Gevaert lab on multiscale data fusion at Stanford University School of Medicine—will be notably different from the previous chapters. We will be considering the problem of topological inference in images, more specifically for the particular task of object detection. This can still be regarded as a topological inference problem defined on graphs, as for object detection it suffices to restrict to *0-dimensional (persistent) homology* (Section 2.5). Nevertheless, we will not be inferring graph-structured models similar as in our previous chapters, but rather well-defined real world 2D shapes and objects.

In Section 7.2 we discuss how 0-dimensional persistent homology can be used for the task of object detection, as well as the difficulties that accompany this approach in case of real-world images. In Section 7.3, we introduce *topological image modification* (TIM), in order to overcome these difficulties and enhance the ability to extract relevant topological information from images. We present two topological image modifiers, namely *smoothing* and *border modification* for this purpose. In Section 7.4, we discuss how this topological information can be used to process images in a completely unsupervised way, as to enhance the ability of segmenting the objects from the images. We qualitatively and quantitatively confirm the effectiveness of our approach for improving three different generic and unsupervised methods for providing binary segmentations of biomedical skin lesion images in Section 7.5. We qualitatively confirm that our approach improves three other algorithms for oversegmentation, edge detection, and binary segmentation in Section 7.6. We conclude upon the work in this chapter in Section 7.7.

1.2 Visualizations

(Graph) visualizations will be tremendously important throughout this entire thesis, both for explaining our methods as well for qualitative analysis of our results. Indeed, given how difficult it is to mathematically formalize topological models (as we will discuss in Section 3.4), thoroughly illustrating their existence is necessary to understand these models. We therefore emphasize how these visualizations were obtained right from the start. For this, we distinguish between two cases.

- **Graphs with an extrinsic space** will correspond to proximity graphs (Rips or kNN) constructed from Euclidean point cloud data. We therefore always use the coordinates from the original data to provide the layout of the nodes. These are either given explicitly if the data lies in $\mathbb{R}^2$ or $\mathbb{R}^3$, or derived from a dimensionality reduction method. This will be clear from context, or specified in the main text and/or figure caption.

- **Graphs without an extrinsic space** correspond to graphs given at hand (e.g. social networks). These graphs will always be plotted using layout algorithms from the igraph library in R [14]. Unless the layout algorithm has been specified, we use the standard setting. This means that the algorithm is chosen through the layout_nicely function, a smart function that chooses the layouter based on the graph itself.

1.3 Code

R (Chapters 4 & 6), R and Python (Chapter 5), and Python (Chapter 7) code for this entire thesis organized with READMEs per chapter can be found on `https://github.com/aida-ugent/PhD_Code_Robin_Vandaele`.

2 Literature Overview: Topological Data Analysis and Graphs

2.1 Introduction

The emergent area of *topological data analysis* (TDA) aims to understand the *shape of data* [15, 16]. Its tools are increasingly being applied to supervised and unsupervised machine learning problems [17–21]. The most prominently applied and studied methods are *persistent homology* and the *Mapper* algorithm. In this chapter, we introduce these methods, the mathematical background on which they are founded, and other concepts and results that will be used throughout this thesis.

Note that not all of the material presented in this chapter will be equally essential for the rest of this thesis. A concise overview of the key observations and material per section is therefore provided in Section 1.1.

2.2 Introduction to Category Theory

Category Theory is the branch of mathematics that formalizes mathematical structures. Many mathematical structures, such as topological spaces, groups, and graphs, can be studied in a generic setting through category theory.

Categories contain *classes*, which are collections of mathematical objects that can be unambiguously defined by a property shared by its members. As not uncommon in mathematics, our notion of class is informal. Classes differ from *sets* in that the latter can be formally defined through the systematic axiomatic rules provided by Zermelo–Fraenkel set theory [22]. In contrast to this, classes are very generic in terms of the mathematical objects they may include.

Definition 2.2.1. *(Category). A* category *C consists of the following three mathematical entities:*

- *A class* $\mathrm{Obj}(C)$ *whose elements are called the* objects*;*
- *A class* $\hom(C)$ *of* morphisms*, also called* arrows *or* maps*. Each morphism f has a* source object $a \in \mathrm{Obj}(C)$ *and a* target object $b \in \mathrm{Obj}(C)$*. We write $f : a \to b$ and say that f is a* morphism from a to b*. We write* $\mathrm{Hom}_C(a, b)$*, or* $\mathrm{Hom}(a, b)$ *when C is clear from the context, to denote the* hom-class *of all morphisms from a to b.*
- *For every $a, b, c \in \mathrm{Obj}(C)$, a binary operation*

  $$\circ : \mathrm{Hom}(a, b) \times \mathrm{Hom}(b, c) \to \mathrm{Hom}(a, c) : (f, g) \mapsto \circ(f, g) =: g \circ f,$$

  (one may read 'g after f',) called composition of morphisms*, such that the following axioms hold:*

  1. *For all $x \in \mathrm{Obj}(C)$, there exists a morhpism $\mathbb{1}_x : x \to x$, called the* identity morphism *for x, such that for all $f \in \mathrm{Hom}(a, x)$, and $g \in \mathrm{Hom}(x, b)$, we have*

     $$\mathbb{1}_x \circ f = f \text{ and } g \circ \mathbb{1}_x = g \qquad \textit{(identity)}.$$

  2. *For all $f \in \mathrm{Hom}(a, b)$, $g \in \mathrm{Hom}(b, c)$, and $h \in \mathrm{Hom}(c, d)$, we have*

     $$h \circ (g \circ f) = (h \circ g) \circ f \qquad \textit{(associativity)}.$$

Isomorphism are formalized in category theory as the functions that preserve all mathematical structure of the considered spaces. They are the homeomorphisms in the category of topological spaces, the isometries in the category of metric spaces, and so on.

Definition 2.2.2. *(Isomorphism). Let C be a category and $f \in \mathrm{Hom}_C(a, b)$. f is called an* isomorphism *if there exists $g \in \mathrm{Hom}_C(b, a)$ such that $g \circ f = \mathbb{1}_a$ and $f \circ g = \mathbb{1}_b$. In this case, g is called the* inverse *of f, and is also denoted by f^{-1}.*

2.3 Basic Concepts of Topology

Topology is the branch of mathematics concerned with the properties of particular structured spaces, more specifically, *topological spaces*, that are preserved under continuous deformations, i.e., *homeomorphisms*. These deformations include stretching, twisting, crumpling and bending, but not tearing or gluing.

The focus of this section is to introduce the basic concepts of topology used in this thesis.

There are several equivalent definitions of a topological space. One may choose the axiomatisation that is best suited for the application. The choice will have no significant impact on this thesis. We will use the most commonly used definition in terms of open sets [23].

Definition 2.3.1. *(Topological Space). A* Topological Space *is an ordered set (X, τ), where X is a set and τ is a collection of subsets of X, i.e., $\tau \subseteq \mathcal{P}(X)$, satisfying the following axioms:*

1. *$\emptyset \in \tau \wedge X \in \tau$.*
2. *Any union of members of τ belongs to τ. This union can be taken over finitely or infinitely many sets.*
3. *The intersection of any finite number of members of τ belongs to τ.*

The elements of τ are called open sets, *and the collection τ is called a* topology *on X.*

If (X, τ) is a topological space, then for any subset $S \subset X$, τ straightforwardly induces a topology on S as follows.

Definition 2.3.2. *(Subspace Topology). Let (X, τ) be a topological space and $S \subseteq X$. Letting $\tau_S := \{S \cap U : U \in \tau\}$, the tuple (S, τ_S) forms a topological space, called the* subspace topology *on S.*

In a sense, topological spaces (X, τ) are some of the weakest structured spaces. E.g., they do not impose any proximity measure between the points contained in X. This leads to the rather famous and traditional joke that a topologist cannot distinguish a coffee mug from a donut (the variant of the pastry with the hole in the center). This is because either shape can be *continuously* deformed into the other.

Definition 2.3.3. *(Continuous Function). Let (X, τ_X) and (Y, τ_Y) be two topological spaces. A function $f : X \to Y$ is said to be* continuous, *if*

$$V \in \tau_Y \implies f^{-1}(V) := \{x \in X : f(x) \in V\} \in \tau_X.$$

Hence, f is said to be continuous if the inverse image $f^{-1}(V)$ of every open set V in Y is an open set in X.

Definition 2.3.4. *(Category of Topological Spaces). The category of topological spaces* **Top** *is the category whose objects are topological spaces and whose morphisms are continuous maps.*

Homeomorphisms are known as the isomorphisms for the category of topological spaces: they preserve *all* the topological structure of a given topological space. Due to their importance within the field of topology, we provide their definition—which also easily follows from Definition 2.2.2—separately.

Definition 2.3.5. *(Homeomorphism). Let (X, τ_X) and (Y, τ_Y) be two topological spaces. A function $f : X \to Y$ is said to be a* homeomorphism *if it has the following properties:*

- *f is a bijection.*
- *f is continuous.*
- *The inverse function $f^{-1} : Y \to X$ is continuous.*

Hence, the joke that a topologist cannot distinguish a coffee mug from a donut, is mathematically formalized through the fact that there exists a homeomorphism between these shapes.

One of the many topological properties preserved through such homeomorphisms, are the *connected components* of a topological space.

Definition 2.3.6. *(Connectedness in Topological Spaces). A topological space (X, τ) is said to be* connected *if X cannot be represented as the union of two or more disjoint nonempty open subsets in τ. The maximal connected subsets (in the sense of Definition 2.3.2) are called the* connected components *of X.*

Topologists are brilliant when it comes to finding counterintuitive examples that contradict what one might believe. Hence, contrary to what one may suspect, if (X, τ) is a connected topological space, then not necessarily every two points $x, y \in X$ are connected by a path. For this we have the separate notion of *path connectedness*, which can easily be shown a strictly stronger concept than connectedness.

Definition 2.3.7. *(Path Connectedness in Topological Spaces). Let (X, τ) be a topological space. A* path *in X is a continuous function $\varphi : [a, b] \to X$, for some $a \leq b \in \mathbb{R}$. Two points $x, y \in X$ are said to be* path connected, *denoted $x \sim y$, if there exists a path $\varphi : [a, b] \to X$ such that $\varphi(a) = x$ and $\varphi(b) = y$. (X, τ) is said to be* path connected *if every two points in X are path connected.*

The definition above requires that $\mathbb{R}$ can be regarded as a topological space. Indeed, it is a *metric space* for the metric $d : \mathbb{R} \times \mathbb{R} : (x, y) \mapsto |x - y|$. Below, we will show that metric spaces are a special type of topological spaces, that have additional structure imposed on them. Note that when working with data, we most often deal with metric spaces.

Definition 2.3.8. *(Metric Space). A metric space is an ordered pair (X, d) where X is a set and d is a metric on X, i.e., a function $d : X \times X \to \mathbb{R}$, such that for any $x, y, z \in X$, the following holds:*

$$\begin{aligned} d(x,y) = 0 \iff x = y &\qquad \textit{(identity of indiscernibles)} \\ d(x,y) = d(y,x) &\qquad \textit{(symmetry)} \\ d(x,z) \leq d(x,y) + d(x,z) &\qquad \textit{(triangle inequality).} \end{aligned}$$

Definition 2.3.9. *(Metric Map). Let (X, d_X) and (Y, d_Y) be two metric spaces. A* metric map *from X to Y is a function $f : X \to Y$ that does not increase any distance, i.e., for all $x_1, x_2 \in X$, we have*

$$d_Y\left(f(x_1), f(x_2)\right) \leq d_X(x_1, x_2).$$

Definition 2.3.10. *(Category of Metric Spaces). The category of metric spaces* $\mathbf{Met}$ *is the category whose objects are metric spaces and whose morphisms are metric maps.*

Isometries are the isomorphism of metric spaces (Definition 2.2.2), they preserve all distances between points. Hence, although a coffee mug and a donut may be topologically equivalent, they are not in the sense of metric spaces, i.e., isometries. Thankfully, for this very reason, humanity is unlikely to pour their morning drink into their breakfast.

The fact that metric spaces may be regarded topological spaces, is because they admit an intuitive concept of open sets.

Definition 2.3.11. *(Open Sets in Metric Spaces). Let (X, d) be a metric space. A subset $U \subseteq X$ is said to be* open*, if*

$$(\forall x \in U)(\exists \epsilon \in]0, \infty[)(\forall y \in X)(d(x,y) < \epsilon \implies y \in U).$$

By letting τ be the set of open sets according to this definition, one easily shows that (X, τ) forms a topological space.

In data science, we often deal with finite metric spaces, which we term point clouds.

Definition 2.3.12. *(Point cloud). A* point cloud *is a metric space (X, d) for which $|X| \in \mathbb{N}$.*

In practice, point clouds (X, d) often arise from some (unknown and infinite) metric space (M, d'). The restriction of d' to $X \times X$ must not necessarily be equal to d, and the purpose of d may be to provide an estimate of d' [6].

Remark 2.3.13. *According Definition 2.3.11, any two point clouds (X, d_X) and (Y, d_Y) for which $|X| = |Y|$ are topologically equivalent. By this, we mean there*

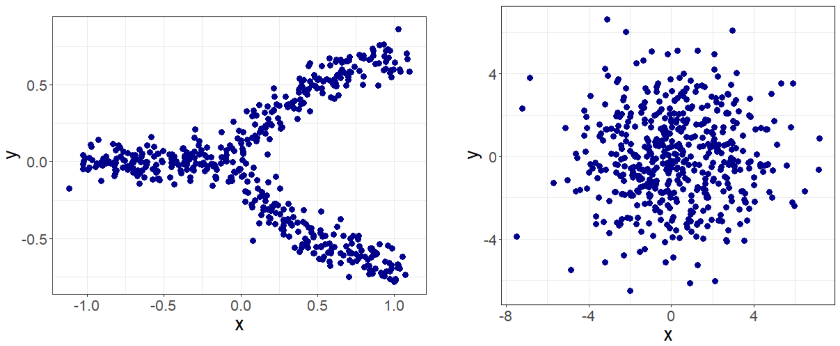


Figure 2.1: Two equally sized point cloud samples X (Left) and Y (Right) from the metric space $(\mathbb{R}^2, d_{\mathbb{R}^2})$, where for $d \geq 1$, we denote $d_{\mathbb{R}^d}$ for the Euclidean distance metric *in $\mathbb{R}^d$. Although X visually resembles a bifurcating structure, according to Definition 2.3.11 it cannot be topologically distinguished from Y (which resembles more of a 'blob').*

exists a homeomorphism between the topological spaces (X, τ_{d_X}) and (Y, τ_{d_Y}) induced by the metrics d_X and d_Y, respectively. Indeed, observe that for every $r \in \mathbb{R}^+$, τ_{d_X} contains the open ball

$$B_{d_X}(x, r) := \{z \in X : d_X(x, z) < r\}.$$

By taking $r = \epsilon_X := \min_{x \neq z \in X} d_X(x, z)$, for every $x \in X$, it follows that $B_{d_X}(x, \epsilon_X) = \{x\} \in \tau_{d_X}$ The second axiom of a topological space in Definition 2.3.1, now shows that $\mathcal{P}(X) \subseteq \tau_{d_X}$, so that necessarily $\mathcal{P}(X) = \tau_{d_X}$. Analogously $\mathcal{P}(Y) = \tau_{d_Y}$. According to Definitions 2.3.3 & 2.3.5, every bijection $f : X \to Y$ then trivially defines a homeomorphism between the topological spaces (X, τ_{d_X}) and (Y, τ_{d_Y}). Figure 2.1 shows that this is yet another example of how weak topology may be (e.g. in comparison to geometry). Indeed, although the two data sets in Figure 2.1 cannot be topologically distinguished according to the Euclidean distance metric, they can be clearly geometrically distinguished for the same metric (in the sense that there does not exist an isometry between the metric spaces). Note that it is important to say that these spaces cannot be topologically distinguished according to the metrics, *as one may define other topologies on the finite sets such that they can indeed be topologically distinguished. An example of this would be taking the topology τ_{d_X} induced by the metric d_X for the metric space (X, d_X), and the* trivial topology $\tau = \{\emptyset, Y\}$ *for the metric space (Y, d_Y).*

The *Gromov-Hausdorff distance* quantifies how 'close' two metric spaces are to being isometric.

Definition 2.3.14. *Let (X, d_X) and (Y, d_Y) be two metric spaces. A* correspondence *is a set $C \subseteq X \times Y$, such that for any $x \in X$, there exists $y \in Y$ such* *that $(x, y) \in C$, and vice versa. Given $\epsilon > 0$, a correspondence C is an ϵ-correspondence if $(x, y), (x', y') \in C$ implies that $|d_X(x, x') - d_Y(y, y')| \leq \epsilon$. The* Gromov-Hausdorff distance $d_{\mathrm{GH}}(X, Y)$ *is the infimum of the ϵ for which there exists an ϵ-correspondence between (X, d_X) and (Y, d_Y).*

Remark 2.3.15. *Below we will informally regard certain spaces (X, d) that define a metric space apart from the requirement that d is real-valued, as metric spaces as well. More specifically, we will consider metric spaces (X, d) for which we may have $d(x, y) = \infty$ for some $x, y \in X$. We then take $|\infty - \infty| = 0$ when this occurs in the definition of an ϵ-correspondence as a result.*

Finally, the definition of *totally bounded metric spaces* will be important for introducing the stability results in Section 2.5.

Definition 2.3.16. *A metric space (X, d) is called* totally bounded *if for every $\epsilon > 0$ there exists a finite collection of open balls in X of radius ϵ whose union contains X.*

2.4 Graphs, Proximity Graphs, and Metric Graphs

Graphs, also often called *networks*, will be the most important mathematical objects in this thesis. They are widely used for representing relations, information, and structure in various fields of science. Some examples include:

- gene regulation networks and protein interaction networks in biology [24, 25];
- co-authorship networks and social networks in social sciences [26, 27];
- road networks in geoinformatics [28];
- proximity graphs (Section 2.4.1) constructed from point cloud data, such as earthquake locations in geology [4], galaxies in space in astrophysics [29], or cell trajectory data in biology [9];
- graphs modeling partially structured data, such as images [30].

Figure 2.2 shows an example of a graph from social sciences.

As mathematical objects themselves, graphs are commonly studied in the fields of graph theory, combinatorics, and optimization [31]. They are formally defined as follows.

Definition 2.4.1. *(Graph). An undirected finite* graph *is a pair $G = (V, E)$, where V is a finite set whose elements are called the* vertices *of G, and*

$$E \subseteq \{S \in \mathcal{P}(V) : |S| = 2\}$$

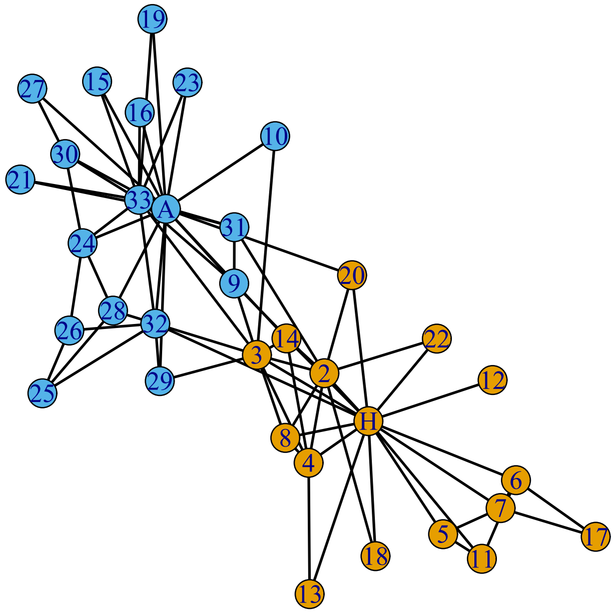


Figure 2.2: A well-known example of a graph is the Karate network *[12]. This graph models 34 members of a karate club, and connects members who interacted outside the club. A conflict between the administrator "John A" (A) and instructor "Mr. Hi" (H) led to splitting the club into two communities, corresponding to the blue and yellow nodes.*

is a set whose elements are called the edges *of* G*. Here,* $\mathcal{P}(V)$ *denotes the* power set *of* V*, i.e., the set of all subsets of* V*. We also denote* $V(G)$ *to refer to the set of vertices of* G*, and* $E(G)$ *to refer to the set of edges of* G*. We say that* G *is positively weighted, if we have a* weighting function $\omega : E(G) \to \mathbb{R}^+$*. For any* $v \in V$*, we call* $\delta(v) := \{e \in E : v \in e\}$ *the* degree *of* v*. We call* v *a* leaf *of* G *if* $\delta(v) = 1$*, and a* multifurcation *if* $\delta(v) \geq 3$.

When referring to any graph in this thesis, we mean in the sense of Definition 2.4.1. Hence, we always consider finite, undirected, positively weighted graphs, without selfloops (since $e \in E$ implies that e is a set for which $|e| = 2$) or parallel edges (since E is a set). In the case we have no explicit weighting function ω, i.e., for unweighted graphs, we implicitly let $\omega \equiv 1$, i.e., ω equals the constant function that maps every edge to weight 1. For both weighted and unweighted graphs, the weighting function will often remain unmentioned, unless specifically required so.

Similar to topological spaces, graphs have a notion of *connectedness* defined on them. This is closely related to the concept of *paths*.

Definition 2.4.2. *(Paths in Graphs). Let* $G = (V, E)$ *be a graph. A* walk *in* G

is sequence $v_0, e_1, v_1, \ldots, v_k$, $k \in \mathbb{N}$, of vertices $v_i \in V$, $0 \leq i \leq k$, and edges $e_i \in E$, $1 \leq i \leq k$, such that $e_i = \{v_{i-1}, v_i\}$. A walk $v_0, e_1, v_1, \ldots, v_k$, is called a path *(from v_0 to v_k) if all vertices $v_0, \ldots, v_k$ are distinct. Note that the edges $e_1, \ldots, e_k$ in a path $v_0, e_1, v_1, \ldots, v_k$, are all distinct, and we sometimes identify a path with the set of edges it contains. If there exists a path from u to v, then we say that u and v are* connected *in G. Note that by taking $k = 0$ in the definition of a path, it follows that each node is connected to itself. A walk $v_0, e_1, v_1, \ldots, v_k$, is called a* cycle, *if $k > 2$, $v_0 = v_k$, and all vertices $v_0, \ldots, v_{k-1}$ are distinct.*

Definition 2.4.3. *(Connectedness in Graphs). Denote by $u \sim v$ that there exists a path from u to v in G. Otherwise, we denote $u \nsim v$. Then $\sim$ defines an equivalence relation $V/\sim$ on V, whose equivalence classes are determined by the sets*

$$[v] := \{v \in V : u \sim v\},$$

for $v \in V$. The elements in $V/\sim$ are called the connected components of *G, and $[v] \in V/\sim$ is called the* connected component of *v. We denote $\beta_0(G) := |V/\sim|$ for the* number of connected components *of G. We say that G is* connected *if $\beta_0(G) = 1$, and* disconnected *if $\beta_0(G) > 1$. Note that for the empty graph $G = (\{\}, \{\})$, it holds that $\beta_0(G) = 0$.*

The concepts of paths and connectedness allow us to define a notion of distance between nodes in a graph as follows.

Definition 2.4.4. *(Shortest Path Distance). Let $G = (V, E)$ be a graph with weighting function ω, and $u, v \in V$. The* shortest path distance *between u and v is defined as*

$$d_G(u, v) = \begin{cases} \infty & \text{if } u \nsim v; \\ \min\{\mathcal{L}(P) : P \text{ is a path from } u \text{ to } v \text{ in } G\} & \text{if } u \sim v, \end{cases}$$

where

$$\mathcal{L}(P) := \sum_{e \in P} \omega(e),$$

is the length *of the path P. The tuple (V, d_G) is called the* metric space induced by *G. We also denote d_G^{unw} for the metric that is obtained on G by letting $\omega \equiv 1$.*

By letting $a + \infty = \infty$ for $a \in \mathbb{R}$, $\infty + \infty = \infty$, and $\infty \leq \infty$, the tuple $(V(G), d_G)$ satisfies the three axioms of a metric space given in Definition 2.3.8. However, d_G is not necessarily a real-valued function whenever G is disconnected, as formally required by this definition. Nevertheless, we will often regard the tuple $(V(G), d_G)$ as a metric space for any graph G. It would be unnecessary to formally introduce a separate definition for metric spaces with possibly infinite distances between points for this thesis. Note that this space does however formally defines a topological space $(V(G), \tau)$ in the sense of Definition 2.3.11. However, even though the presence of edges in the defining graph might make it apparent that we also impose some additional continuous segments in this space, this is not the case, and one would still not be able to distinguish this topological space from any other point cloud (X, d) for which $|X| = |V(G)|$, as explained in Remark 2.3.13. Hence, the notions of connectedness in G (Definition 2.4.3) and $(V(G), \tau)$ (Definition 2.3.6) do not coincide.

2.4.1 Proximity Graphs

For a large part of this thesis, we will study data which has an underlying graph-structured topology, which will be further clarified in Chapter 3. This data may come in the form of a graph itself, and the task may then be to infer a much simpler topological graph representation that models the data well. In other cases, we only have a point cloud available. The data may then resemble a graph when looking to the data 'from far away' (Figure 2.1 Left). However, as the topological space induced by the metric on the data (Definition 2.3.11), it cannot be distinguished from any arbitrary metric space containing an equal number of points, as also discussed in Remark 2.3.13.

Nevertheless, on a point cloud (X, d), we can impose at least some non-trivial notion of connectedness, i.e., in the sense of Definition 2.4.3, by constructing a *proximity graph* on X. These graphs are constructed according to particular *neighborhood rules*, which specify when two points in X should be connected by an edge in the resulting graph. The type of neighborhood rule defines the class of the considered proximity graph. We will consider two of the most commonly used proximity graphs, namely the *(Vietoris-)Rips graph*, also known as the *unit disk*

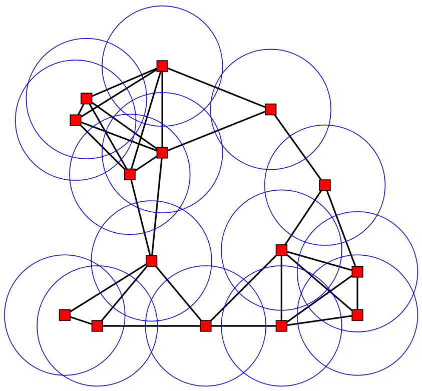


Figure 2.3: A collection of fixed radius closed ball neighborhoods, and the corresponding Rips graph. Image from [32].

graph [33], and the (undirected) *kNN graph* [34].

Definition 2.4.5. *(Rips graph). Let (X, d) be a finite metric space, and $\epsilon \in \mathbb{R}_{\geq 0}$. The Rips graph $\mathcal{R}_\epsilon(X)$ is defined as the graph $G = (V, E)$, for which $V = X$, $E = \{\{x, y\} \in X : 0 < d(x, y) \leq \epsilon\}$, and $\omega : E \to \mathbb{R}^+ : \{x, y\} \mapsto d(x, y)$.*

Figure 2.3 illustrates how a Rips graph is constructed. These graphs will be used extensively in Chapter 4 to infer graph-structured models underlying point cloud data, but find other applications such as representing the bonds between atoms in complex molecules as well [35].

Unlike the Rips graphs, where the 'scope' around each point is kept fixed, the kNN graph allows one to vary this scope according to the local data density.

Definition 2.4.6. *(kNN graph). Let (X, d) be a finite metric space, and $\epsilon \in \mathbb{R}_{\geq 0}$, and $k \in \mathbb{N}$. Suppose for each $x \in X$, its set of k closest neighbors (distinct from x) according to d, denoted $\mathcal{N}_k(x)$, is uniquely defined. The (undirected) kNN graph $\mathrm{NN}_k(X)$ is defined as the graph $G = (V, E)$, for which $V = X$, $E = \{\{x, y\} \subseteq X : y \in \mathcal{N}_k(x) \vee x \in \mathcal{N}_k(y)\}$, and $\omega : E \to \mathbb{R}^+ : \{x, y\} \mapsto d(x, y)$.*

It is a natural assumption that the set of k closest neighbors for any point is uniquely defined in practice. Nonzero distances between different pairs of points sampled from a continuous space are generally equal with zero probability.

Figure 2.4 illustrates a point cloud data set $X \subseteq \mathbb{R}^2$, that resembles a graph-structured topology (containing one edge and one isolated node), as well as both

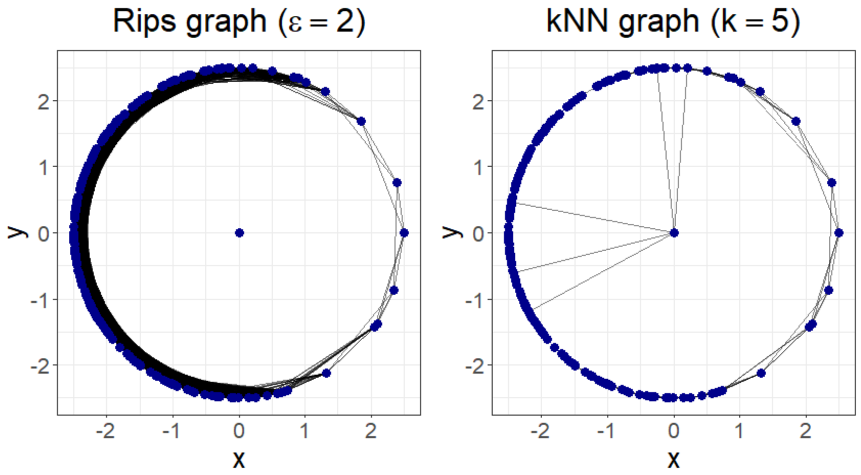


Figure 2.4: An example of a Rips graph (left) and a kNN graph (right). Rips graphs tend to deal better with outliers, while kNN graphs tend to deal better with non-uniform density, requiring less edges to truthfully represent the geometry of the model in this case.

a Rips graph ($\epsilon = 2$) and a kNN graph ($k = 5$) are constructed from X. From this example, one can immediately deduce important properties of these classes of proximity graphs. Rips graphs are more robust to outliers, but may include many edges in dense regions of the data. In contrast to this, kNN graphs often force outliers to connect to far away regions. However, they generally require less edges to truthfully represent dense regions, only depending on the local scale of the data.

By constructing a proximity graph G on a finite point cloud X, we essentially do not resolve our issue that X cannot be topologically distinguished from an arbitrary point cloud that consists of an equal number of points. Indeed, the identity map from X to itself (or even any permutation) induces a homeomorphism between the topological spaces induced my the metric spaces (X, d) and (X, d_G) (Definition 2.3.11). This is why the notions of connectedness in graphs and topological spaces are distinct. Nevertheless, proximity graphs serve as geometrical approximations of *metric graphs*, which will be introduced in the following section. This allows one to study relevant topological properties of continuous models through discrete proximity graph representations [6].

2.4.2 Metric Graphs

Metric graphs will be some of the most important topological spaces handled in this thesis. They are metric spaces that generally consist of an uncountable number of points, but can be modeled through a (finite) graph. They are formally defined as follows.

Definition 2.4.7. *(Metric Graph). A* metric graph *$\mathcal{M}$ in $\mathbb{R}^d$ is the union of the images of a set of smooth curves $\left\{\varphi_i : [a_i, b_i] \to \mathbb{R}^d\right\}_{i=1,\ldots,m}$, with the following properties*

- (join at endpoints) *for every $1 \leq i < j \leq m$, $\varphi_i(]a_i, b_i[) \cap \varphi_j(]a_j, b_j[) = \emptyset$ and the cardinality of $\varphi_i([a_i, b_i]) \cap \varphi_j([a_j, b_j])$ is either 0 or 1,*
- (no self-intersections) *for every $1 \leq i \leq m$, φ_i is injective.*

The points in $V(\mathcal{M}) := \bigcup_{i=1,\ldots,m}\{\varphi_i(a_i), \varphi_i(b_i)\}$ are called the nodes *of $\mathcal{M}$. Note that $\mathcal{M}$ may have isolated nodes, as our definition allows the necessary condition $a_i = b_i$ for some $1 \leq i \leq m$ for this to hold. We also sometimes simply identify $\mathcal{M}$ with the set $\left\{\varphi_i : [a_i, b_i] \to \mathbb{R}^d\right\}_{i=1,\ldots,m}$. We denote by $\Gamma_{\mathcal{M}}$ the set of maximal connected components of $\mathcal{M}$, where $\mathcal{M}$ is seen as the topological space induced by the standard topology on $\mathbb{R}^d$. We call $\mathcal{M}$* connected *if $|\Gamma_{\mathcal{M}}| = 1$. For a node $v \in V(\mathcal{M})$, we define the* degree of *v as $\delta(v) := |\{1 \leq i \leq m : \{v\} \subsetneq \mathrm{Im}(\varphi_i)\}|$. For $1 \leq i \leq m$ and a point $x \in \mathrm{Im}(\varphi_i) \backslash \{\varphi_i(a_i), \varphi_i(b_i)\}$, we define the* degree of *x as $\delta(x) := 2$. $x \in \mathcal{M}$ is called a* leaf *of $\mathcal{M}$ if $\delta(x) = 1$. We call x and y* connected, *denoted by $x \sim y$, if there exists a path from x to y in $\mathcal{M}$, otherwise* *we call x and y* disconnected, *denoted by $x \nsim y$. A path $\gamma : [a, b] \to \mathcal{M}$ is called a* cycle *if $b > a$, $\gamma(a) = \gamma(b)$, and γ is injective on $]a, b[$.*

If $\mathcal{M}$ is a metric graph, then each connected component $C \in \Gamma_{\mathcal{M}}$ defines a metric space $(C, d_{\mathcal{M}})$, where for $x, y \in C$, the *geodesic distance* $d_{\mathcal{M}}(x, y)$ equals the minimum of the arc lengths of all piecewise smooth curves $\psi : [a, b] \to C$ for which $\psi(a) = x$ and $\psi(b) = y$. We can extend these metrics to $\mathcal{M} \times \mathcal{M}$ by letting $d_{\mathcal{M}}(x, y) = \infty$ if x and y are not connected. Similar to the discussion above, it follows that $(\mathcal{M}, d_{\mathcal{M}})$ satisfies all axioms of a metric space, apart from the requirement that $d_{\mathcal{M}}$ is a real-valued function if $\mathcal{M}$ is not connected. Nevertheless, as we did for the metric spaces induced by graphs (Definition 2.4.4), we will regard these spaces informally as metric spaces as well, and directly identify these with the metric graphs themselves. Note that this metric always induces a topology on a $\mathcal{M}$ in the formal sense. Moreover, degrees, paths, connectedness, as well as cycles, are all preserved under homeomorphisms of $\mathcal{M}$, so that we regard these as topological properties of $\mathcal{M}$.

Having a distance measure defined on a metric graph $\mathcal{M}$, we are able to define the *diameter* and *radius* of $\mathcal{M}$ as follows.

Definition 2.4.8. *Let $\mathcal{M}$ be a metric graph. The* diameter *of a metric graph $\mathcal{M}$ is defined as* $\operatorname{diam}(\mathcal{M}) := \max_{x,y\in\mathcal{M}} d_{\mathcal{M}}(x, y) \in [0, \infty]$*, and its radius is defined as* $\operatorname{rad}(d_{\mathcal{M}}) := \min_{x\in\mathcal{M}} \max_{y\in X} d_{\mathcal{M}}(x, y) \in [0, \infty]$.

Note that the diameter and radius of a metric graph $\mathcal{M}$ are metric properties and generally are not preserved under homeomorphisms of $\mathcal{M}$.

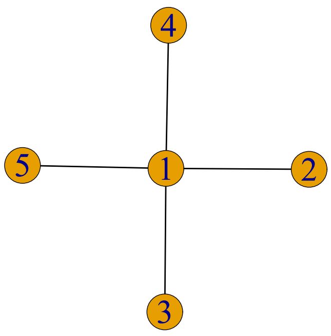
From our discussion above it easily follows that each connected component of a metric graph is path connected. This leads us to the definition of *metric trees*, which will be important in Chapter 6.

Definition 2.4.9. *(Metric Tree). A* metric tree *is a metric graph for which there is a unique path between every two points.*

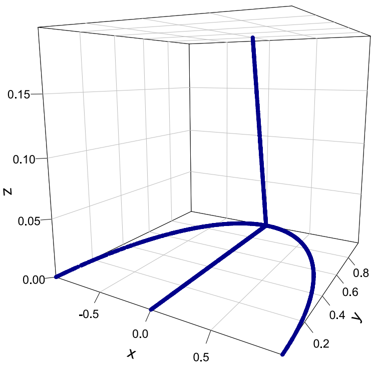
2.4.2.1 Geometric Realization of a Graph

Let $G = (V, E)$ be a graph, and suppose we have a function $\psi : V \to \mathbb{R}^d$ and a smooth curve $\varphi_e : [a_e, b_e] \to \mathbb{R}^d$ for each edge $e \in E$, such that the following properties are valid.

- ψ is injective.
- if $e = \{u, v\} \in E$, then $\{\psi(u), \psi(v)\} = \{\varphi_e(a_e), \varphi_e(b_e)\}$.
- for every $e \neq e' \in E$, $\varphi_e(]a_e, b_e[) \cap \varphi_{e'}(]a_{e'}, b_{e'}[) = \emptyset$.



(a) A graph G.



(b) A geometric realization $\mathcal{M}$ of G in $\mathbb{R}^3$.

Figure 2.5: A geometric realization $\mathcal{M}$ of a graph G. Whereas the metric space induced by G is a discrete space, $\mathcal{M}$ is a continuous space containing uncountable many points.

It can be easily verified that the union of the images of ψ and $\{\varphi_e\}_{e \in E}$ defines a metric graph $\mathcal{M}$. We call any such metric graph $\mathcal{M}$ a *geometric realization* of G. A geometric realization of a graph G can be seen as a 'drawing' of G in $\mathbb{R}^d$, such that no two edges intersect at interior points.

Conversely, every metric graph $\mathcal{M} = \left\{\varphi_i : [a_i, b_i] \to \mathbb{R}^d\right\}_{i=1,\ldots,m}$ defines a graph $G = (V, E)$ for which $\mathcal{M}$ is a geometric realization of G. This graph G can be constructing by letting V be the set of nodes of $\mathcal{M}$, and letting $\{u, v\} \in E$ iff $u \neq v$ and $\{u, v\} = \{\varphi_i(a_i), \varphi_i(b_i)\}$ for some $1 \leq i \leq m$. Graph theoretical 'topological' properties of G, such as connected components, leaves, and cycles, then coincide with the corresponding topological properties of a geometric realization $\mathcal{M}$ of G, and conversely.

It is known that every graph G has a geometric realization in $\mathbb{R}^d$ whenever $d \geq 3$ [36, Geometric Realization Theorem]. If $\mathcal{M}$ is a geometric realization of a graph G in $\mathbb{R}^2$, then $\mathcal{M}$ is called a *planar embedding* of G. G is called a *planar graph* whenever it has a planar embedding. Not every graph admits a planar embedding, and the problem whether a given graph G is planar is computationally easy to solve [31].

Figure 2.5 illustrates a graph G and a geometric realization $\mathcal{M}$ of G in $\mathbb{R}^3$. Note that G clearly also admits a geometric realization in $\mathbb{R}^2$ (as a matter of fact, the visualization in Figure 2.5a directly corresponds to such a realization), and hence, that G is planar.

2.5 Introduction to Topological Persistence

Persistent homology [37] can easily be said to the most profoundly used and studied tool in TDA. It is being increasingly and successfully applied to a wide variety of practical machine learning problems [38–43]. Nevertheless, it has a rather abstract mathematical foundation, most notably in the fields of category theory and algebraic topology [44]. For this reason, at the present day of writing, it is still considered more of a niche and lesser known topic within machine learning.

In the following sections, we will introduce the most important concepts of persistent homology and explain its purpose. As mentioned in the outline of this thesis, the first parts of this section will touch on the many theoretical aspects of (persistent) homology (Section 2.5.1 and the start of Section 2.5.2), and how these relate to its computation (Section 2.5.2.1). Although we ensured that these sections are sufficiently self-contained, they are—even though this might appear not to be the case at first sight—rather concise.[1] Someone unfamiliar with topics such as TDA and algebraic topology may therefore find them more challenging. However, we ensured that the following parts (Sections 2.5.2.2-2.5.2.4)—which are less abstract than the former sections—are sufficiently self-contained for one to comprehend the purpose of persistent homology and what it computes, i.e., *persistence diagrams*. One is therefore freely able to skip to these sections, which are more important for one to comprehend the rest of this thesis.

2.5.1 Simplicial Homology

Simplicial Homology is a way to study topological properties of topological spaces whose 'building blocks' are n-*simplices*, i.e., the n-dimensional analogs of triangles. These are spaces that—by definition—are homeomorphic to *simplicial complexes* (Figure 2.6). Such a homeomorphism is then also called a *triangulation*. Note that the majority of the topological models that occur in data science—including metric graphs—can be triangulated. We formalize these concepts below.

Definition 2.5.1. *(Abstract Simplicial Complex). A family of finite sets Δ is called an* abstract simplicial complex *if for every $\sigma \in \Delta$ and $\sigma' \subseteq \sigma$ it holds that $\sigma' \in \Delta$. The finite sets that belong to Δ are called* faces *or* simplices *of the complex. The* dimension *of a face $\sigma \in \Delta$ is defined as $|\sigma| - 1$, and σ is called a $(|\sigma| - 1)$-simplex. An* abstract simplicial k-complex *is a simplicial complex Δ where the largest dimension of any face in Δ equals k. The* vertex set $V(\Delta)$ *of an abstract simplicial complex Δ is the set of 0-simplices in Δ. Two abstract simplicial complexes Δ and Δ' are called* isomorphic *if there exists a bijection $\phi : V(\Delta) \to V(\Delta')$ for which $\sigma \in \Delta$ iff $\phi(\sigma) := \{\phi(v) : v \in \sigma\} \in \Delta'$.*

[1]For a more detailed description we recommend [44].

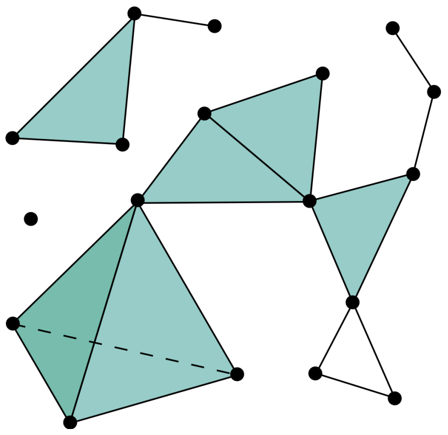


Figure 2.6: A simplicial 3-complex. Image from [45].

Any graph $G = (V, E)$ straightforwardly induces an abstract simplicial complex $V \cup E$, since $v \in e \in E \implies v \in V$. Hence, an abstract simplicial complex lies somewhere between a graph and a *hypergraph*. Similar to a graph, the latter is also a tuple $H = (V, E)$—where the elements in $E \subseteq \mathcal{P}(V)$ are called the *hyperedges* of H—but without the additional restriction that $e \in E \implies |e| = 2$. Hence, hypergraphs admit higher-order relations than graphs. Abstract simplicial complex can then be seen as hypergraphs, with the additional constraint that its hyperedges are closed under inclusion. Intuitively, we thus have the following class relationships

$$\{\text{graphs}\} \subsetneq \{\text{abstract simiplicial complexes}\} \subsetneq \{\text{hypergraphs}\}.$$

As the name suggests, there is also a 'less abstract' definition of simplicial complexes.

Definition 2.5.2. *(Simplicial Complex). A* k-simplex σ *is the convex hull*

$$\sigma = \left\{ \sum_{i=0}^{k} \lambda_i v_i : \sum_{i=0}^{k} \lambda_i = 1 \wedge \lambda_0, \ldots, \lambda_k \geq 0 \right\}$$

of $k + 1$ affinely independent points $v_0, \ldots, v_k \in \mathbb{R}^d$, $d \geq k$. k is then called the dimension *of σ. The convex hull of any nonempty subset of the $k + 1$ points that define a k-simplex is called a* face *of the simplex. A* simplicial complex K *is a set of simplices, satisfying the following conditions:*

1. *every face of a simplex in K is also in K;*
2. *the non-empty intersection of any two simplices $\sigma, \sigma' \in K$ is a face of both σ and σ'.*

A simplicial k-complex *is a simplicial complex K where the largest dimension of any simplex in K equals k.*

Let K be a simplicial k-complex. Denote $V(\sigma)$, for the finite set of points in $\mathbb{R}^d$, $d \geq k$, of which the convex hull equals σ. It can then be easily seen that the set $\Delta = \{V(\sigma) : \sigma \in K\}$ is an abstract simplicial complex. Conversely, it is known that every abstract simplicial k-complex Δ is isomorphic to an abstract simplicial complex $\Delta = \{V(\sigma) : \sigma \in K\}$ for some simplicial d-complex K for which $d \geq 2k + 1$ [36, Geometric Realization Theorem]. We then call $|\Delta| = K$ a *geometric realization* of Δ. Loosely speaking, an abstract simplicial complex is a simplicial complex without associated geometry. Note that this concept of geometric realization coincides with the one for graphs and metric graphs defined in Section 3.2.

The Euclidean space in which a simplicial complex (geometric realization) $|\Delta|$ is embedded, induces a topology on $|\Delta|$ (Definition 2.3.2). Most often, these are the topological properties we are interested in, and not the topological properties of the 'isolated points' in Δ (see also the discussion Section 2.4.1). However, in a computational setting—similar to (metric) graphs—topological properties of $|\Delta|$ are studied through their discrete counterparts of the finite representation Δ. We now formalize what these topological properties correspond to.

Suppose that Δ is a finite abstract simplicial complex. By definition, an *orientation* of a simplex $\sigma \in \Delta$ is given by an ordering $(v_0, \ldots, v_k)$ of the vertices in σ, with the convention that two orderings define the same orientation if they differ by an even permutation. This means that every simplex σ admits exactly two distinct orientations, and by convention, they are each other negatives. Note that we omit a more formal definition of an orientation, as the fact that simplices with an opposite orientation cancel each other out in summation will be more important than the exact choice which orientation is positive and which is negative. Intuitively, if σ is a 1-simplex, which can be visualized through a line segment, then choosing the orientation coincides with choosing one of the two possible directions to traverse this segment. If σ is a 2-simplex, which can be visualized through a triangle, then choosing an orientation coincides with what the choice between clockwise and counterclockwise means. Note that providing a similar intuitive meaning of choosing the orientation is rather difficult for higher-dimensional simplices.

Before we proceed with introducing the concept of *homology*, we require the introduction of a fundamental algebraic structure, known as a *ring*. As we will see below, fixing a ring will be important when computing topological properties from finite abstract simplicial complexes.

Definition 2.5.3. *A ring is a set R equipped with two binary operations $+$ and $\cdot$, respectively termed the* addition *and* multiplication*, satisfying the following axioms:*

1. *R is an* abelian group *under the addition, meaning that*
   - *for all $a, b, c \in R$, $(a+b)+c = a+(b+c)$, that is, $+$ is* associative*;*
   - *for all $a, b \in R$, $a+b = b+a$, that is, $+$ is* commutative*;*
   - *there exists an element $0 \in R$ such that $a+0 = a$ for all $a \in R$, that is, 0 is the additive* identity*;*
   - *for every $a \in R$ there exists $-a \in R$ such that $a+(-a) = 0$ for all $a \in R$, that is, $-a$ is the additive* inverse *of a;*
2. *R is* monoid *under multiplication, meaning that*
   - *for all $a, b, c \in R$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$, that is, $\cdot$ is* associative*;*
   - *there exists an element $1 \in R$ such that $a \cdot 1 = 1 \cdot a = a$ for all $a \in R$, that is, 1 is the multiplicative* identity*;*
3. *the multiplication is* distributive *with respect to the addition, meaning that*
   - *for all $a, b, c \in R$, $a \cdot (b+c) = (a \cdot b) + (a \cdot c)$, that is, $\cdot$ is* left distributive*;*
   - *for all $a, b, c \in R$, $(b+c) \cdot a = (b \cdot a) + (c \cdot a)$, that is, $\cdot$ is* right distributive.

We also sometimes denote $a - b$ for $a + (-b)$, and ab for $a \cdot b$. If furthermore the multiplication R is commutative, i.e., $a \cdot b = b \cdot a$ for all $a, b \in R$, then R is called a commutative ring. *Finally, if R is a commutative ring in which each $0 \neq a \in R$ has a* multiplicative inverse *$a^{-1} \in R$, i.e., for which it holds that $aa^{-1} = 1$, then R is called a* field.

The concept of a ring only plays a minor role in this thesis, and we mainly include its definition for completeness. Note that many known structures, such as $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ equipped with their standard addition and multiplication are (commutative) rings. However, the natural numbers $\mathbb{N}$ is not, as the addition generally has no inverse.

Choosing a commutative ring R is albeit important for (persistent) homology and its computation, and—as we will discuss below—the obtained results may depend on this choice. A popular choice in TDA is taking $R = \mathbb{F}_2$, i.e., the (smallest) Galois field of two elements $\{0, 1\}$ in which the addition and multiplication are performed modulo 2. The benefit of taking $R = \mathbb{F}_2$ is that one does not have to bother with orientations of simplices: positive and negative are then the same. As a matter of fact, for all persistent homology computations within this thesis we take $R = \mathbb{F}_2$. (At least) for all of our considered data, this choice will not matter. Furthermore, the fact that one requires choosing such ring to compute topological properties of a space defined independently from any ring may be rather confusing. One will therefore find that starting from Section 2.5.2.2, we will no longer talk about these rings.

Given a simplicial complex Δ in which each complex has an orientation, and a commutative ring R, a *simplicial k-chain* is a finite *formal sum*

$$\lambda_1\sigma_1 + \ldots + \lambda_N\sigma_N,$$

where each σ_i is an oriented k-simplex in Δ and $\lambda_i \in R$, $1 \leq i \leq N \in \mathbb{N}$, and (as discussed above) each orientation is declared equal to the negative of the simplex with the opposite orientation. By a formal sum we mean that even if it doesn't make sense to add things, we do it anyway. One should not bother too much with trying to interpret these sums. E.g, if $\sigma \neq \sigma'$ are two k-simplices in Δ, then $\sigma + \sigma'$ is just $\sigma + \sigma'$, and nothing else. It does not equal $\sigma \cup \sigma'$, nor does it equal $\tilde{\sigma}$ for any other $\tilde{\sigma} \in \Delta\backslash\{\sigma, \sigma'\}$. As some fictional Disney character might say: "let it go". Welcome to the wonderful world of abstract mathematics.

Nevertheless, the operations permitted by such formal sums are not much different from those by more well-known structures, such as vector spaces. Indeed, considering the basic vectors $e_1 = (1, 0)$ and $e_2 = (0, 1)$ in $\mathbb{R}^2$, there is no 'easier' way to write $e_1 + e_2$. Yet, there is 'another' way, i.e., $e_1 + e_2 = (1, 1)$, and one may use similar notations for formal sums as well. Furthermore, similar to vector spaces, operations such as $\sigma + \sigma = 2\sigma$, $\sigma - \sigma = 0$, $10 \cdot (42\sigma) = 420\sigma$ are all permitted, but operations such as multiplying two simplices σ_1 and σ_2 are not straightforwardly defined. The fact that such formal sums are very similar in behavior to vector spaces, is because the set of all k-chains with the formal addition over R forms an *R-module*, which is a weaker variant of a K-vector space (K being a field).

Definition 2.5.4. *(Module). Let R be a ring and 1 its multiplicative identity. A* left R-module M *consists of an abelian group $(M, +)$ and an operation $\cdot : R \times M \to M$, such that for all $r, s \in R$ and $x, y \in M$, we have*

1. $r \cdot (x + y) = r \cdot x + r \cdot y$;
2. $(r + s) \cdot x = r \cdot x + s \cdot x$;
3. $(rs) \cdot x = r \cdot (s \cdot x)$;
4. $1 \cdot x = x$.

For $r \in R$ and $x \in M$, we also sometimes denote rx for $r \cdot x$. The elements in R are called the scalars. *A* right R-module *is defined analogously, except that the ring acts on the right, i.e., $\cdot : M \times R \to M$, and the above axioms are written with the scalars on the right. If R is commutative, then left R-modules are the same as right R-modules, and are simply called R-modules.*

The set of all k-chains, denoted Δ_k, hence forms an R-module for the chosen commutative ring R (one may assume all the axioms in Definition 2.5.4 and their right hand counterparts to hold by definition). A natural set of *generators* of Δ_k—which are to be interpreted similarly as basis vectors of a vector space—is then the set of all k-simplices in Δ.

Similar to linear maps between vector spaces, we can have linear maps between modules, which are completely determined by the image of the generators. These are the *(homo)morphisms in the category of R-modules.*

Definition 2.5.5. *(Boundary Map). Given a simplicial complex Δ and a ring R, the* boundary map $\partial_k : \Delta_k \to \Delta_{k-1}$ *between the R-modules Δ_k and Δ_{k-1} is defined by letting for each simplex $\sigma = (x_0, \ldots, x_k)$,*

$$\partial_k(\sigma) = \sum_{i=0}^{k} (-1)^i (x_0, \ldots, \hat{x_i}, \ldots, x_k),$$

where $(x_0, \ldots, \hat{x_i}, \ldots, x_k)$ is the $(k-1)$-face of σ obtained by omitting the vertex x_i, while preserving the ordering of the remaining vertices. The boundary map extends linearly to all k-chains in Δ_k. By convention, we let $\partial_0 \equiv 0$. This coincides with taking 'empty sums' in the definition of ∂_0, and letting $\Delta_{-1} = 0$. If C is a k-chain in Δ_k, we also sometimes refer to $\partial_k(C)$ as the boundary *of C.*

Chain complexes are algebraic structures that consist of a sequence of abelian groups of modules and a sequence of homomorphisms between consecutive groups such that the image of each homomorphism is included in the *kernel* of the next. Note that throughout this section, by kernel we mean the algebraic structure that consists of all elements mapped to 0, i.e., the neutral element for the addition. Chain complexes can be defined and studied more generally in the field of category theory. As we have not—and will not require to have—specified (homo)morphisms in the case of (abelian) groups, we will only provide the definition for modules.

Definition 2.5.6. *(Chain Complex). A* chain complex $(A_\bullet, d_\bullet)$ *is a sequence of modules $\ldots, A_0, A_1, \ldots$, also called* boundary operators, *connected by homomorphisms $d_n : A_n \to A_{n-1}$, such that the composition of any two consecutive maps d_n and d_{n-1} is the zero map $d_{n-1} \circ d_n : A_n \to A_{n-2} : x \mapsto 0$.*

The collection of R-modules Δ_k connected by the boundary maps ∂_k forms a chain complex $(\Delta_\bullet, \partial_\bullet)$. This is easy to see whenever we take $R = \mathbb{F}_2$, as for any

$\sigma = (v_0, \ldots, v_k) \in \Delta$, we have

$$\begin{aligned}
\partial_{k-1}(\partial_k(\sigma)) &= \partial_{k-1}\left(\sum_{i=0}^{k}(v_0, \ldots, \hat{v}_i, \ldots, v_k)\right) \\
&= \sum_{i=0}^{k} \partial_{k-1}\left((v_0, \ldots, \hat{v}_i, \ldots, v_k)\right) \\
&= \sum_{i=0}^{k}\left(\sum_{j=0}^{i-1}(v_0, \ldots, \hat{v}_j, \ldots, \hat{v}_i, \ldots, v_k) \quad + \right. \\
&\qquad \left. \sum_{j=i+1}^{k}(v_0, \ldots, \hat{v}_i, \ldots, \hat{v}_j, \ldots, v_k)\right) \\
&= 2 \sum_{0 \le i < j \le k}(v_0, \ldots, \hat{v}_i, \ldots, \hat{v}_j, \ldots, v_k) = 0.
\end{aligned}$$

A more general proof for rings $R \neq \mathbb{F}_2$ can be found in [46, Lemma 2.7.]. As a consequence, it holds that $\mathrm{Im}(\partial_{k+1}) \subseteq \ker(\partial_k)$.

Definition 2.5.7. *Let Δ be an abstract simplicial complex, R a ring, and ∂_k the corresponding boundary maps. We call the kernel $\ker(\partial_k)$ the k-*th cycle module*, and the image $\mathrm{Im}(\partial_k)$ the k-*th boundary module.

The relation $\mathrm{Im}(\partial_{k+1}) \subseteq \ker(\partial_k)$ has a geometric interpretation. It implies that every $(k+1)$-*boundary*, which is a k-chain that is the image of some $(k+1)$-chain under ∂_{k+1}, has zero boundary under ∂_k, and hence, by definition, is a k-*cycle*. However, the reverse inclusion does generally not hold. In this case, there exist k-cycles that are not the boundary of any $(k+1)$-chain. These correspond to 'holes' in the simplicial complex, as illustrated by Figure 2.7.

Definition 2.5.8. *(Homology Module). Let Δ be an abstract simplicial complex, R a ring, and ∂_k the corresponding boundary maps. The k-*th homology module *of Δ is the R-module*

$$H_k := \ker(\partial_k)/\mathrm{Im}(\partial_{k+1}).$$

*The k-*th Betti number *of Δ, denoted β_k, is the rank of H_k, i.e., its number of generators.*

The k-th Betti-number β_k quantifies the extend to which the reverse inclusion $\ker(\partial_k) \subseteq \mathrm{Im}(\partial_{k+1})$ fails. Following the conventing in Definition 2.5.5, in case of the 0-th Betti-number, we find that

$$\beta_0 = \mathrm{rank}(H_0) = \mathrm{rank}(\ker(\partial_0)/\mathrm{Im}(\partial_1)) = \mathrm{rank}(\Delta_0/\mathrm{Im}(\partial_1)),$$

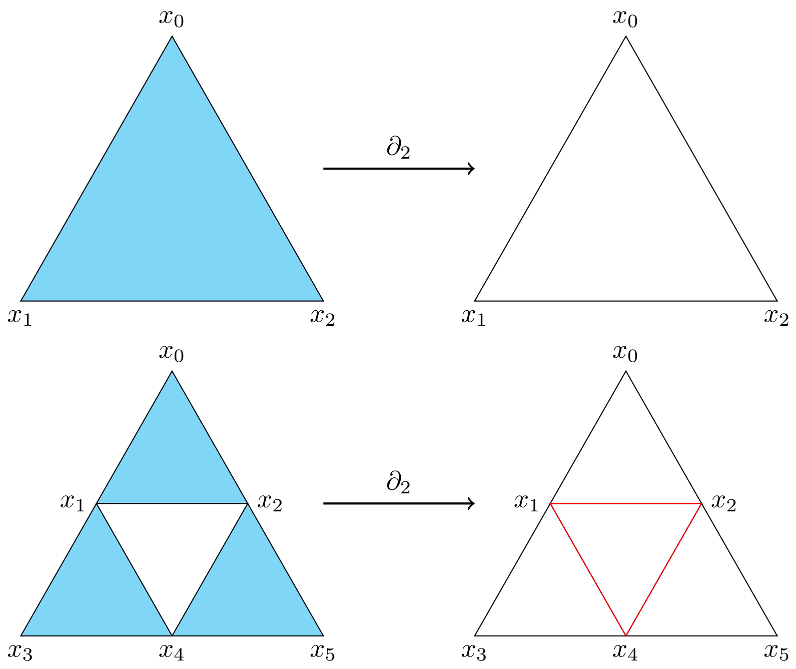


Figure 2.7: Geometric interpretation of the boundary operator. The boundary operator ∂_{k+1} maps a $(k+1)$-chain to a k-cycle. However, some k-cycles, such as the red chain in the bottom simplicial complex, are not the boundary of any $(k+1)$-chain. These correspond to 'holes' in our simplicial complex.

Now consider the graph that results from Δ after discarding all but the nodes (0-simplices) and edges (1-simplices), as well as discarding the orientation of all the edges. We call this graph the *1-skeleton* or *graph skeleton* of Δ, and call two nodes of Δ connected whenever they are connected in this graph, and identify (connected) components of this graph with connected components of Δ. Now take

$$x + \mathrm{Im}(\partial_1) = \lambda_1(v_1) + \dots \lambda_N(v_n) + \mathrm{Im}(\partial_1) \in \Delta_0/\mathrm{Im}(\partial_1).$$

Suppose that $1 \le i < j \le N$, v_i and v_j are connected through a path that goes consecutively through the points $v_i = u_0, u_1, \dots, u_{l-1}, u_l = v_j$ in the 1-skeleton of Δ. Each (u_k, u_{k+1}) and (u_{k+1}, u_k) is in Δ_1 for $0 \le k < l$. Hence, we find that

$$\begin{aligned}\lambda_i(v_i) + \mathrm{Im}(\partial_1) &= \lambda_i(v_i) + \lambda_i\partial_1\left((v_i, u_1) + \dots + (u_{l-1}, v_j)\right) + \mathrm{Im}(\partial_1)\\ &= \lambda_i(v_i) + \lambda_i\left((v_i) - (u_1) + \dots + (u_{l-1}) - (v_j)\right) + \mathrm{Im}(\partial_1)\\ &= -\lambda_i(v_j) + \mathrm{Im}(\partial_1).\end{aligned}$$

Hence, we are allowed to substitute the term $\lambda_i(v_i) + \lambda_j(v_j) + \mathrm{Im}(\partial_1)$ by the term $(\lambda_j - \lambda_i)(v_j) + \mathrm{Im}(\partial_1)$ in the expression for $x + \mathrm{Im}(\partial_1)$. By induction, it holds that

$$x + \mathrm{Im}(\partial_1) = \lambda'_{i_1}(v_{i_1}) + \ldots \lambda'_{i_M}(v_{i_M}) + \mathrm{Im}(\partial_1) \in \Delta_0/\mathrm{Im}(\partial_1),$$

where each v_{i_j} belongs to a different component of Δ, and there is no 'simpler' way to write $x + \mathrm{Im}(\partial_1)$. It follows that β_0 equals the number of connected components of Δ. If $k \geq 1$, then β_k expresses the number of *k-dimensional holes*. 1-dimensional holes correspond to 'circular' holes, whereas 2-dimensional holes coorespond to 'voids' or 'cavities'. These types holes can be straightforwardly visualized. In general, for $k \geq 1$, a k-dimensional hole is to be interpreted as the inside of a k-dimensional sphere. The core power of (persistent) homology is that it allows one to exactly quantify the number of such holes in $|\Delta|$, through its discrete representation Δ. E.g., in the bottom simplicial complex in Figure 2.7, there are essentially many cycles, in a graph theoretical sense. However, there is only one 1-dimensional hole in the topological space $|\Delta|$, so that $\beta_1(\Delta) = 1$.

Although we have only introduced *simplicial* homology, expressing holes (topological properties) in simplicial complexes, homology can be defined for any topological space. This is known as *singular* homology. Instead of considering simplices, one considers *singular simplices*, which are injective continuous functions from the standard d-simplex (the convex hull of the $d + 1$ unit vectors in $\mathbb{R}^{d+1}$) to a topological space. Boundary operators and chain complex are then defined analogously as for simplicial homology. Simplicial homology and singular homology agree for spaces that can be triangulated. This allows one to talk about the homology, and in following sections, filtrations, persistent homology, and persistence diagrams, of any topological space occurring in data science, whether these are geometric realizations of abstract simplicial complexes constructed from data, or topological models that underlie the observed data. For more information on this topic, we refer to Hatcher, 2020 [44].

As also commented above, at first reading one might consider it strange that Betti-numbers, which quantify topological properties, i.e., 'holes', depend on a choice of ring R, which may be completely unrelated to the actual topological space $|\Delta|$. Indeed, there exist topological spaces with different Betti-numbers when computed over different rings. An example is the *Klein bottle* K, for which we would find $\beta_1(K) = 2$ over $R = \mathbb{F}_2$, and $\beta_1(K) = 1$ over $R = \mathbb{F}_p$ for any prime number $p > 2$ [47]. This is related to the fact that the Klein bottle is an example of a non-orientable surface. This is beyond the scope of this thesis however, and we refer the interested reader to Hatcher, 2020 [44]. For our topological spaces of interest—most notably (metric) graphs—one may unambiguously interpret the Betti number β_k as the number of k-dimensional holes in the space, not requiring the ring R to be specified. Without further specification, all (persistent) homology computations in this thesis are performed over $R = \mathbb{F}_2$, as also mentioned above.

Finally, one should be aware that 'having the same Betti-numbers' is a strictly weaker concept than ‘having the same topology’. More formally, *Betti-numbers* are *topological invariants*, i.e., they are preserved under homeomorpishms of topological spaces (keeping the ring R fixed), but not conversely. E.g., a circle is not homeomorphic to a cylinder, but both topological spaces consist of one connected component, have one 1-cycle (more formally one equivalence class of such cycles), and do not include any higher dimensional cycles. One cannot discern between these spaces based on homology. It is important to realize this. As we discussed above, topology does not allow us to distinguish a coffee mug from a donut. Homology thus allows one to distinguish between even less spaces. However, it is known that there cannot exist an algorithm to determine when arbitrary topological spaces are homeomorphic [48]. Hence, in some sense, we are lucky that there exists something like (persistent) homology which we can actually compute, as will be discussed in the following section.

2.5.2 Persistent Homology

Persistent homology can be regarded as *simplicial homology at varying scales*. Instead of identifying the Betti-numbers of one given simplicial complex, we track the change in Betti-numbers across a varying sequence of simplicial complexes. This is formalized through the concept of a *filtration*.

Definition 2.5.9. *(Filtration). Let Δ be an abstract simplicial complex. A* filtration *is a sequence $\mathcal{F} = (\Delta^i)_{i\in\mathbb{N}}$ of subcomplexes of Δ, such that $\Delta^i \subseteq \Delta^j$ whenever $i \leq j$.*

The i-th simplicial complex Δ^i in a filtration $\mathcal{F}$ gives rise to its own chain complex $(\Delta^i_\bullet, \partial^i_\bullet)$. Now consider the inclusion maps $\iota^i : \Delta^i \hookrightarrow \Delta^{i+1}$ for a given filtration $\mathcal{F}$. Each inclusion map ι^i induces a *chain map* $(\iota^i_\bullet) = (\iota^i)_{i\in\mathbb{N}}$ between the chain complexes $(\Delta^i_\bullet, \partial^i_\bullet)$ and $(\Delta^{i+1}_\bullet, \partial^{i+1}_\bullet)$. More formally, for $x \in \Delta^i_k$ and $k \in \mathbb{N}$, it holds that

$$\partial^i_k \circ \iota^i(x) = \partial^i_k(x) = \iota^i \circ \partial^i_k(x).$$

This is written out in the *commutative diagram* in Figure 2.8.

Definition 2.5.10. *Let $\mathcal{F} = (\Delta^i)_{i\in\mathbb{N}}$ be a filtration defined on an abstract simplicial complex Δ. The sequence of chain complexes $\big((\Delta^i_\bullet, \partial^i_\bullet)\big)_{i\in\mathbb{N}}$ connected by the chain maps $\big((\iota^i_\bullet)\big)_{i\in\mathbb{N}}$ is called a* persistence complex.

Now let $\mathcal{F} = (\Delta^i)_{i\in\mathbb{N}}$ be a filtration defined on an abstract simplicial complex Δ, and take

$$x + \mathrm{Im}(\partial^i_{k+1}) \in H^i_k = \ker(\partial^i_k)/\mathrm{Im}(\partial^i_{k+1}).$$

It holds that $x \in \Delta^i \subseteq \Delta^{i+1}$, so that $\partial^{i+1}_k(x)$ is well-defined and must necessarily equal $\partial^{i+1}_k(x) = \partial^i_k(x) = 0$. It follows that $x \in \ker(\partial^{i+1}_k)$, so that

$$\eta^i_k(x + \mathrm{Im}(\partial^i_{k+1})) := x + \mathrm{Im}(\partial^{i+1}_{k+1}) \in H^{i+1}_k = \ker(\partial^{i+1}_k)/\mathrm{Im}(\partial^{i+1}_{k+1}).$$

$$\begin{tikzcd}
 & \vdots \arrow[d, "\partial^i_{k+2}"] & \vdots \arrow[d, "\partial^{i+1}_{k+2}"] & \\
\cdots \arrow[r, "\iota_{i-1}"] & \Delta^i_{k+1} \arrow[r, "\iota_i"] \arrow[d, "\partial^i_{k+1}"] & \Delta^{i+1}_{k+1} \arrow[r, "\iota_{i+1}"] \arrow[d, "\partial^{i+1}_{k+1}"] & \cdots \\
\cdots \arrow[r, "\iota_{i-1}"] & \Delta^i_k \arrow[r, "\iota_i"] \arrow[d, "\partial^i_k"] & \Delta^{i+1}_k \arrow[r, "\iota_{i+1}"] \arrow[d, "\partial^{i+1}_k"] & \cdots \\
 & \vdots & \vdots &
\end{tikzcd}$$

Figure 2.8: A persistence complex illustrated through a commutative diagram.

Hence, the inclusions $(\iota^i)_{i\in\mathbb{N}}$, induce module homomorphisms $\eta^i_k : H^i_k \to H^{i+1}_k$.

Definition 2.5.11. *Let $\mathcal{F} = (\Delta^i)_{i\in\mathbb{Z}}$ be a filtration defined on an abstract simplicial complex Δ. The k*th persistence module *$\mathcal{H}_k$ is the family of kth homology modules H^i_k together with the module homomorphisms $\eta^i_k : H^i_k \to H^{i+1}_k$. A persistence module is said to be of* finite type *if each component module is finitely generated and there exists some integer $z \in \mathbb{Z}$ such that the maps η^i_k are isomorphisms for all $i \geq z$.*

The finiteness of an abstract simplicial complex guarantees that the corresponding persistence module is of finite type [46]. Indeed, in this case we cannot indefinitely grow the simplicial complexes in $(\Delta_i)_{i\in\mathbb{N}}$, and at some point, the corresponding homology modules must remain fixed.

Similar to how simplicial homology generalizes to simplicial homology for arbitrary topological spaces, all concepts defined in this section generalize to arbitrary topological spaces as well. However, persistence modules of finite type—which are those we deal with in practice—allow for a convenient decomposition leading to the concept of a *persistence diagram.* First, we require some new definitions.

Definition 2.5.12. *(Graded Ring) A* graded ring *R is a ring with a direct sum decomposition into abelian groups*

$$R = \bigoplus_{i\in\mathbb{N}} R^i,$$

such that for each $x \in R^i$ and $y \in R^j$, it holds that $xy \in R^{i+j}$.

An example of a graded ring is the polynomial ring

$$\mathbb{F}[X] := \bigoplus_{i\in\mathbb{N}} X^i \cdot \mathbb{F},$$

where $\mathbb{F}$ is a field.

Definition 2.5.13. *A* left graded module *is a left module over a graded ring R, such that*

$$M = \bigoplus_{i\in\mathbb{N}} M^i,$$

such that for each $\lambda \in R^i$ and $m \in M^j$, it holds that $\lambda m \in M^{i+j}$.

A kth persistence module $\mathcal{H}_k$ can be identified with a graded module over the corresponding polynomial ring $R[X]$, i.e.,

$$\mathcal{H}_k = \bigoplus_{i\in\mathbb{N}} H_k^i,$$

where we let

$$X \cdot \sum_{i\in\mathbb{N}} m^i := \sum_{i\in\mathbb{N}} \eta_k^i(m^i),$$

which inductively and linearly extends to $\mathcal{H}_k$.

Theorem 2.5.14. *[46, Theorem 4.8] Let $\mathcal{H}_k$ be a persistence module of finite type over a polynomial ring $\mathbb{F}[X]$, where $\mathbb{F}$ is a field. Then there exists $M, N \in \mathbb{N}$, such that up to a module homomorphism,*

$$\mathcal{H}_k = \left(\bigoplus_{i=1}^{M} (X^{a_i})\right) \oplus \left(\bigoplus_{i=1}^{N} \left(\frac{X^{b_j}}{X^{d_j}}\right)\right).$$

If $\mathcal{F} = (\Delta^i)_{i\in\mathbb{Z}}$ is a filtration defined on an abstract simplicial complex Δ, then the powers a_i, b_j, d_j, in the decomposition of $\mathcal{H}_k$ in Theorem 2.5.14 capture the *birth* and *death* of k-dimensional holes across the sequence of simplical complexes $(|\Delta^i|)_{i\in\mathbb{Z}}$ [46]. The numbers b_j and d_j express that a k-dimensional hole that appeared at complex Δ_{b_j} disappears at complex Δ_{d_j}. The numbers a_i that a k-dimensional hole appeared at complex Δ_{a_i}, but never disappears. This forms the basic idea behind persistent homology and persistence: *k-dimensional holes that persist over long intervals represent relevant topological properties of the considered space*. This will be made clear in the following sections, which will visualize the information captured through persistent homology by means of *persistence diagrams*.

2.5.2.1 Computing Persistent Homology

In a practical setting, we are dealing with finite data. Corresponding simplicial complexes and filtrations will then be finite as well. Instead of indexing filtrations through integers, they are commonly indexed through an increasing sequence of reals $t_1 < \ldots < t_n$, called *times*. The purpose of computing persistent homology is then to obtain the times at which k-dimensional holes across a filtration

$$\mathcal{F} = \emptyset \subseteq \Delta_{t_1} \subseteq \ldots \subseteq \Delta_{t_n} = \Delta.$$

are *born* and at which times they *die*, for a given upper bound on the dimension k of the holes. Formally, this corresponds to indexing $\mathcal{F}$ through the integers $1 \leq i \leq n$, extending $\mathcal{F}$ indefinitely through $\Delta_{t_n} = \Delta_n \subseteq \Delta_{n+1} := \Delta \subseteq \Delta_{n+2} := \Delta \subseteq \ldots$, identifying the integer powers a_i, b_j, d_j in Theorem 2.5.14 for each of the considered dimension of holes, and mapping them back to the corresponding times $t_{a_i}, t_{b_j}, t_{d_j} \leq t_n$. For ease of explanation, we will assume that we are considering the fixed field of coefficients $\mathbb{F}_2$ over which we compute (persistent) homology (which is exactly our choice throughout this thesis).

First, we order the $m = |\bigcup \Delta|$ simplices $\{\sigma_1, \ldots, \sigma_m\}$ in Δ, such that $i < j$ whenever $t(\sigma_i) < t(\sigma_j)$, where $t(\sigma)$ denotes the *time of appearance* of σ, defined as

$$t(\sigma) = \min\{t_i : \sigma \in \Delta_{t_i}\}.$$

Naturally, we only include simplices of which the cardinality is at most the dimension of holes we are interested plus two. We then construct a binary matrix M of which the rows and columns correspond to the simplices in Δ, for which $M_{ij} = 1$ iff $\sigma_i \subseteq \sigma_j$ and $|\sigma_i| = |\sigma_j| - 1$. The birth and death times can then be obtained through simple linear and algebraic operations on M (modulo 2) [40, Algorithm 2.1].

The time complexity of this algorithm is cubic in the number of simplices m [40]. This is still one of the major drawbacks of (computing) persistent homology to this day. E.g., when one is interested in 1-dimensional holes, i.e., loops, one requires to build 2-simplices, resulting in the number of simplices itself being cubic in the number of data points. However, in case one is only interested in 0-dimensional holes, persistent homology can be computed in $\mathcal{O}(n\alpha(n))$ time—n being the number of vertices, i.e., 0-simplices, which equals the number of data points in practice—using a union-find structure [36]. Here $\alpha(\cdot)$ is the inverse of the Ackermann function, which for all practical purposes may be considered a constant no greater than 4 [49].

In general, efficiently computing or 'approximating' (through the *bottleneck distance* metric which will be defined in the following section) persistent homology is an active research topic at the time of writing [50, 51].

2.5.2.2 Introducing Persistent Homology through Euclidean Point Clouds

Although the fundamental results leading to *persistent homology* discussed in the previous sections are rather abstract, its purpose and what it computes are actually easy to visualize. As mentioned in the outline of this thesis, we will ensure that this and the following sections are sufficiently self-contained for one to understand the *purpose* of persistent homology, and the *persistence diagrams* it computes. For this reason, some of the terminology we will introduce in this section may already have been introduced (more formally) above.

A custom case in data science for illustrating how persistent homology works, is that we have a point cloud data set X embedded in a Euclidean space $\mathbb{R}^d$. In our case, we will take a look at the point cloud X that is visualized in left plot in Figure 2.9. The task is then to infer topological properties of the model underlying X, by means of persistent homology.

Looking at X (Figure 2.9 Left), the only topological information that can be deduced from it is that it is a set of points, since no two point clouds of the same size can be *topologically* distinguished as discussed in Remark 2.3.13 (at least according to any metric defined on them, which we assume to be the Euclidean distance metric here).

A partial solution to this can be obtained by constructing a *simplicial complex* from X, for which one often considers the *Vietoris-Rips complex*

$$VR_\epsilon^k(X) := \left\{ S \subset X : |X| \leq k+1 \wedge \max_{x,y \in S} \|x - y\| \leq \epsilon \right\}. \tag{2.1}$$

Each element σ in such a simplicial complex is called a *face* or *simplex*. If $|\sigma| = k+1$, it is also called a k-simplex, and k is called the *dimension* of σ. The *dimension* of a simplicial complex is the maximal dimension of its included simplices, and is constrained by k for the Vietoris-Rips complex $\mathcal{VR}_\epsilon^k(X)$. Here, ϵ is a parameter that specifies the upper bound on the diameter of subsets of X which should be included in the simplicial complex. Note that $\mathcal{VR}_\epsilon^1(X) = \mathcal{R}_\epsilon(X)$ (hence our chosen name for the Rips graph). $\mathcal{VR}_\epsilon^k(X)$ is then obtained through $\mathcal{R}_\epsilon(X)$ by 'filling in' all cliques up to cardinality $k+1$ in the graph through a simplex. Six examples of such complexes $\mathcal{VR}_\epsilon^2(X)$ (for varying ϵ) are shown in Figure 2.9.

In general, a simplicial complex can be seen as a generalization of a graph, where apart from nodes (0-simplices) and edges (1-simplices), it may also include triangles (2-simplices), tetrahedra (3-simplices), and so on. More specifically, the two defining property of simplicial complex Δ are that

- Δ is a set of finite subsets of some given set X;
- If $\sigma' \subseteq \sigma \in \Delta$, then $\sigma' \in \Delta$

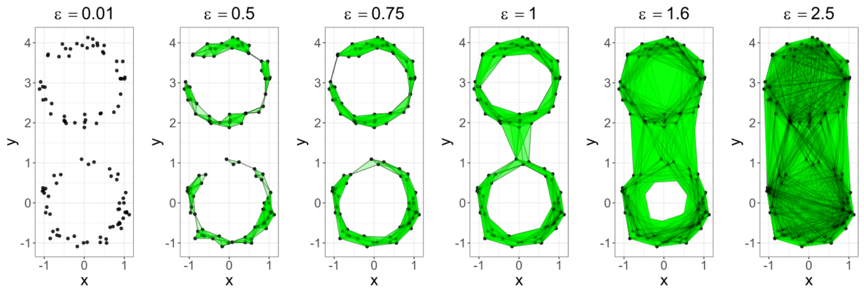


Figure 2.9: An example of six simplicial complexes $VR^2_\epsilon(X)$ in the Vietoris-Rips filtration of a point cloud data set X resembling two disconnected circles in the Euclidean plane.

However, one should note that technically this is the definition of an *abstract* simplicial complex, and the term simplicial complex actually refers to a *geometrical realization* of such a complex. Such realization can be seen as a continuous drawing of the complex in some Euclidean space $\mathbb{R}^d$, such that the intersection of two simplices σ and σ' corresponds to a common face of σ and σ'. This is similar to how a metric graph is a geometric realization of a graph, but we also 'fill in' the 2-simplices, 3-simplices, ..., which is usually how we visualize (abstract) simplicial complexes (such as in Figure 2.9). Sometimes the term 'simplex' is only used to refer to geometric realizations of their discrete counterparts, and the discrete counterparts are only referred to as 'faces'. However, throughout the rest of this thesis, it will not really be a problem for one to mix up this terminology, i.e., to identify simplicial complexes with their discrete counterparts, and we will often proceed to do this. The most important thing to be aware of, is that (persistent) homology is concerned with topological properties of the 'continuous versions' (geometric realizations) of simplicial complexes, which are computed through their 'discrete' (abstract) counterparts. Compare this to how the connectedness of a graph (as a graph) determines the connectedness of any of its geometric realizations (as a topological space).

Once we have constructed a (Vietoris-Rips) complex $\mathcal{VR}^k_\epsilon(X)$ from X, we can now infer more interesting topological properties. E.g., in Figure 2.9, we see that $\mathcal{VR}^2_{0.75}(X)$ captures the two most important topological properties of the model underlying X, which consists of two disconnected circles. *Homology* exactly captures such information by associating *Betti numbers* β_k to our simplicial complex. β_k corresponds to how many k*-dimensional holes* there are in the complex. In this sense, a 0-dimensional hole correspond to a gap between two components, and β_0 equals the number of connected components. A 1-dimensional hole corresponds to a loop (which can be seen as a circle, ring, or a handle of a coffee mug), and a 2-dimensional hole corresponds to a void (which can be seen as the inside of a balloon). In general, an n-dimensional hole corresponds to the interior of a n-sphere in $\mathbb{R}^{n+1}$. Note that no n-dimensional holes can occur in the Euclidean space $\mathbb{R}^d$ whenever $d \leq n$, and therefore (as well as for computational purposes) one often restricts the dimension k of the simplicial complexes that are constructed from the data. E.g., in Figure 2.9, we restrict to the case $k = d = 2$.

The difficulty lies in pinpointing an exact value for ϵ for which $\mathcal{VR}^k_\epsilon(X)$ truthfully captures the topological properties of the model $\mathcal{M}$ underlying X. E.g., $\mathcal{VR}^2_{0.5}(X)$ correctly captures that there are two components in the underlying model, but not that there are two loops (Figure 2.9). $\mathcal{VR}^2_1(X)$ does correctly captures that there are two loops in the underlying model, but then captures that there is only one connected component. This is where 'persistent' homology comes into play. Rather than inferring these topological properties (holes) for one particular simplicial complex, the task of persistent homology is to track the change of these topological properties across a varying sequence of simplicial complexes

$$\Delta_0 \subseteq \Delta_1 \subseteq \ldots \subseteq \Delta_n,$$

called a *filtration*. A commonly used example for Euclidean point cloud data is the *Vietoris-Rips filtration*

$$\mathcal{VR}^k(X) := \left(\mathcal{VR}^k_\epsilon(X)\right)_\epsilon,$$

where ϵ is considered a *time* parameter that parameterizes the filtration. Note that for a point cloud X, $\mathcal{VR}^k_\epsilon(X)$ can only change at finitely many times ϵ, so that we may indeed regard this filtration to be of the (finite) form $\Delta_0 \subseteq \Delta_1 \subseteq \ldots \subseteq \Delta_n$.

The information captured by persistent homology is often visualized by means of a *persistence diagram*, which is a set

$$\mathrm{Dgm}_k = \{(t_{a_i}, \infty) : 1 \leq i \leq N\} \cup \left\{(t_{b_j}, t_{d_j}) : 1 \leq j \leq M\right\} \cup \{(x, x) : x \in \mathbb{R}\},$$

where $t_{a_i}, t_{b_j}, t_{d_j}$, $1 \leq i \leq N$, $1 \leq j \leq N$, correspond to the birth and death times of k-dimensional holes across the filtration.[2] Points (t_{a_i}, ∞) are usually displayed on top of the diagram. They correspond to holes that never die across the (Vietoris-Rips) filtration.

Figure 2.10 shows the persistence diagrams Dgm_0 and Dgm_1 of the Vietoris-Rips filtration for our considered point cloud data X (on top of each other, which is common in TDA). Note that all points in Dgm_0 have 0 as a first coordinate. This corresponds to the fact that all connected components are born at time $\epsilon = 0$ in the filtration, at which time all points are added but no edges between them. At around $\epsilon = 0.5$, all but two components have died, i.e., merged with previously existing components. One can deduce from Dgm_0 that there are still two components alive at this time that merge together at around $\epsilon = 0.8$, after which one

[2]See also Theorem 2.5.14 and Section 2.5.2.1)

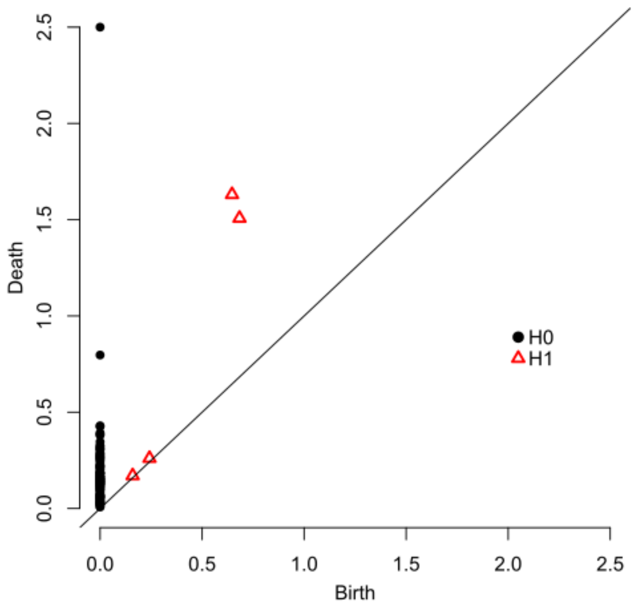


Figure 2.10: The diagrams Dgm_0 *and* Dgm_1 *for the Vietoris-Rips filtration of the point cloud data set in Figure 2.9. The four highly elevated points in the persistence diagram identify the presence of two connected components (H0) and two cycles (H1).*

component persists indefinitely. This is because once the underlying Rips graph $\mathcal{R}_\epsilon(X)$ of the Vietoris-Rips complex connects all points through paths, all points will naturally remain connected whenever we add more simplices to the complex. Observe that this is consistent with the visualization in Figure 2.9. Dgm_1 can be read analogously, but this time for loops rather than components. Furthermore, these loops are all born at different and nonzero times ϵ, since edges are necessary for loops to be present. They all have finite death-times, since (unless we impose an upper bound on ϵ which is sometimes done for computational purposes) they must be filled in at least at some point in time.

Note that for interpreting these persistence diagrams, it is important to be aware of the *elder rule*. This rules states that when two *homology classes* are merged, by convention, the youngest of them, i.e., the one with the greatest birth-time, dies. The term 'homology classes' is formally defined in Section 2.5.1, but not really important for the rest of this thesis, and—for what will be our purpose—one may freely replace this term with 'connected components' in the previous sentence. More specifically, this rule for connected components will play an important role in Chapter 7. In our current case for Dgm_0, all components are born at the same time, and the ordering of birth times are defined by the ordering of the data. However, different orderings would not affect the visualized diagram in Figure 2.10.

From Figure 2.10, we observe that the 'most persisting holes', which correspond to the most elevated points in the diagrams relative to the diagonal, i.e., the points (b, d) for which $b - d$ is large, capture the topological properties of the topological model $\mathcal{M}$ underlying X. The two highly elevated points in Dgm_0 identify the two connected components in $\mathcal{M}$, one of which has an infinite death time, as is custom for Vietoris-Rips filtrations. We also observe two highly elevated points in the diagram Dgm_1 of $\mathcal{VR}^2(X)$. These correspond to the two cycles in $\mathcal{M}$.

The fact that the persistence diagrams in Figure 2.10 capture topological properties of the underlying model $\mathcal{M}$ well, may be quantified through the *bottleneck distance*.

Definition 2.5.15. *(Bottleneck Distance). Let* Dgm *and* Dgm' *be two persistence diagrams. The* bottleneck distance *between them is defined as*

$$d_\mathrm{b}\left(\mathrm{Dgm}, \mathrm{Dgm}'\right) := \inf_{\varphi} \sup_{x} \|x - \varphi(x)\|_\infty \in \mathbb{R} \cup \{\infty\},$$

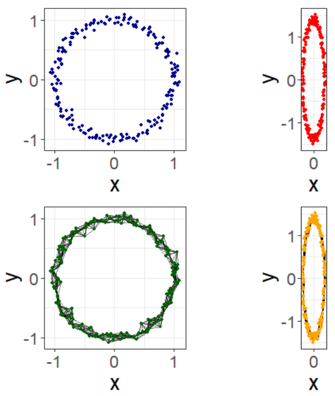
where φ *ranges over all bijections from* Dgm *to* Dgm'*, and* x *ranges over all points in* Dgm*. By convention, we let* $\infty - \infty = 0$ *when calculating the distance between two diagram points. Since persistence diagrams include the diagonal by definition,* $|\mathrm{Dgm}| = |\mathrm{Dgm}'| = |\mathbb{R}|$*. Thus,* $d_\mathrm{b}\left(\mathrm{Dgm}, \mathrm{Dgm}'\right)$ *is well-defined.*

As we will see in the following section, Vietoris-Rips filtrations generalize easily to arbitrary (non-Euclidean) metric spaces. For general metric (and hence, Euclidean) spaces, we may derive a *stability* result (Theorem 2.5.16), which states that if two spaces are geometrically close in terms of the Gromov-Hausdorff distance, so are their resulting persistence diagrams.

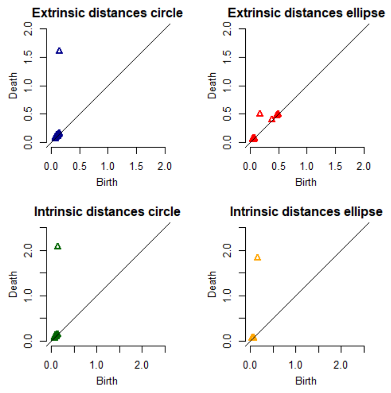
2.5.2.3 Persistent Homology of Metric Spaces

The Vietoris-Rips filtration can be defined for arbitrary metric spaces, where the metric must not necessarily be Euclidean: one may just replace the metric $(x, y) \mapsto \|x - y\|$ in (2.1) by any arbitrary metric $(x, y) \mapsto d(x, y)$. This is especially useful when we are considering a data set X, for which we have a (possibly approximated) intrinsic metric, which is quite distinct form the (usually Euclidean) extrinsic metric on the space X is embedded in.

This is illustrated in Figure 2.11, where we consider a data set X representing a circle in $\mathbb{R}^2$, and a data set Y representing a narrow ellipse in $\mathbb{R}^2$. When we construct the Vietoris-Rips filtration from X through the Euclidean distances, we easily infer the presence of the 1-dimensional hole from the 1-st dimensional persistence diagram. This hole is more difficult to infer through the analogous diagram for Y, since it is much more narrow, and therefore, will be filled in quite early in the Vietoris-Rips filtration.



(a) (Top) Two point clouds sampled from a circle and ellipse and (Bottom) 10NN graphs constructed from them.



(b) The 1-dimensional persistence diagrams obtained from the (Top) Euclidean and (Bottom) empirical geodesic distances.

Figure 2.11: 1-dimensional persistent homology of cyclic models through the Vietoris-Rips filtration constructed from the Euclidean and empirical geodesic distances. The hole in the ellipse is less apparent from the persistence diagram computed through the Euclidean distances. This is resolved by approximating the intrinsic distances of the ellipse through the geodesic distances of a proximity graph representation.

To overcome this issue, we proceed in the following way, which is illustrated in the bottom row of Figure 2.11a. First, through the ordinary (Euclidean) distances, we construct proximity graphs (in this case 10NN graphs, see Definition 2.4.6) from the data. The shortest path distances on these graphs are used to approximate the true (geodesic) distances on the underlying models [6]. When we now construct the Vietoris-Rips filtrations from these (approximated) intrinsic distances instead, the presence of the holes can be easily inferred from both of the resulting persistence diagrams (Figure 2.11b).

This is a formal consequence of the following *stability* result.

Theorem 2.5.16. *[40, Theorem 7.5] Let (X, d_X) and (Y, d_Y) be totally bounded metric spaces. Denote $\mathrm{Dgm}_l(\mathcal{VR}^k(M, d_M))$ for the l-th persistence diagram of the Vietoris-Rips filtration $\mathcal{VR}^k(M, d_M)$ of a metric space (M, d_M). It holds that*

$$d_{\mathrm{b}}\left(\mathrm{Dgm}_l(\mathcal{VR}^k(X, d_X)), \mathrm{Dgm}_l(\mathcal{VR}^k(Y, d_Y))\right) \leq 2d_{\mathrm{GH}}((X, d_X), (Y, d_Y)).$$

The Gromov-Hausdorff distance does not care much about cardinality. Hence, a point cloud X and its underlying model $\mathcal{M}$ can be close according to the Gromov-Hausdorff distance, even though X is finite and $\mathcal{M}$ may consist of an uncountable amount of points. Theorem 2.5.16 then guarantees that the empirical persistence diagram of the Vietoris-Rips filtration constructed from X, will be close to the true persistence diagram of the Vietoris-Rips filtration constructed from $\mathcal{M}$.[3]

Theorem 2.5.16 shows that we can compare metric, and hence, topological, properties between two data sets X and Y through the persistence diagrams of their Vietoris-Rips filtrations. For this as well as more general reasons (Theorem 2.5.18), persistence diagrams are often called *topological signatures*, as they encode topological information through a representation which is independent of the cardinality, dimension, or type of the data considered. Furthermore, computing the Gromov-Hausdorff distance is NP-hard [40], whereas the bottleneck distance between two persistence diagrams with at most n points off the diagonal can be computed in $\mathcal{O}(n^{1.5}\log n)$ time [52, Theorem 3.1]. Given that the computation of the persistence diagram itself is polynomial in time (Section 2.5.2.1), persistence homology leads to a more practical tool for quantifying topological information in metric spaces.

According to Theorem 2.5.16, if two spaces are close to being isometric, then their persistence diagrams are close as well. Unfortunately, the converse is not true. E.g., the Gromov-Hausdorff distance between a cylinder and circle of radius r can be made arbitrarily large, by increasing the height of the cylinder. Nevertheless, we would be unable to distinguish between these spaces based on the persistence diagrams of their Vietoris-Rips filtrations: both spaces are characterized by one connected component, one loop, and no higher-dimensional holes.

2.5.2.4 Persistent Homology of Sublevel Filtrations

The Vietoris-Rips filtration on a finite metric space (X, d) corresponds to a filtration on the simplicial complex that is obtained by filling all cliques of the Rips graph $\mathcal{R}_{d^{<\infty}_{X,\max}}(X)$ through simplices, where

$$d^{<\infty}_{X,\max} := \max\{d_X(x,y) : x, y \in X \wedge d_X(x,y) < \infty\}.$$

Hence, when computing persistent homology of metric spaces, we implicitly impose an 'initial' simplicial complex constructed from our data, which includes all simplicial complexes across our filtration.

In other cases, we have an initial simplicial complex that is implicitly or explicitly induced by the type of data X we are considering. Ideal examples are graphs and images. We then have a function $f : X \to \mathbb{R}$ characterizing our data, and we wish to capture 'topological information' of f. We illustrate this through the example image in Figure 2.12.

Figure 2.12 (Top Left) shows a grayscale image I. We may identify the image I with a function

$$f_I : \Delta_I \to \mathbb{R} : \sigma \mapsto \max_{p \in P} I(p),$$

[3]Note that the latter filtration does not have a countable indexing, as we required—for simplicity—in Definition 2.5.9. However, filtrations, as well as their corresponding persistence diagrams, can be more generally defined over an uncountable set of indices [40].

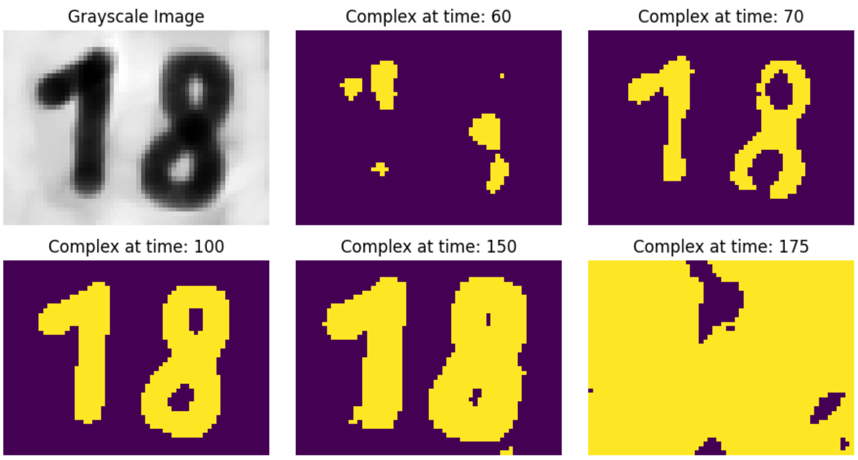


Figure 2.12: A filtration constructed from an image I. Pixels in the complex at a particular time step are marked in yellow.

where Δ_I is a simplicial complex that is obtained by letting the 0-simplices be the set of pixels—or more formally, their coordinates—of the image I, the 1-simplices the edges connecting all the neighboring pixels in I (vertically, horizontally, and diagonally), and the 2-simplices all the cliques (triangles) of the resulting graph. For a pixel p, i.e., a 0-simplex in Δ_I, $I(p)$ then equals the color—in this case, the grayscale—value of the pixel in the image I.

We may then construct the *sublevel filtration* on the simplicial complex Δ_I,

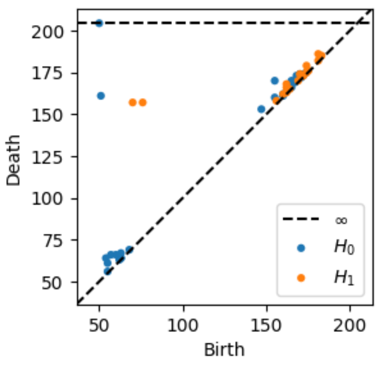


Figure 2.13: Persistent homology of a grayscale image I. Two 'outlying' lifetimes for 0-dimensional holes (H_0) represent the two components of I (the '1' and '8'). Similarly, the two 'outlying' lifetimes for 1-dimensional holes (H_1) represent the two holes in I (the holes in the '8').

defined as

$$\bar{\mathcal{F}}(f_I) := \left(\bar{\mathcal{F}}_t(f_I) := \{\sigma \subset \Delta_I : f_I(p) \leq t\}\right)_{t\in\mathbb{R}}.$$

Similar to Vietoris-Rips filtration constructed from finite metric spaces, the filtration $\bar{\mathcal{F}}(f_I)$ only changes at finitely many steps. Figure 2.12 illustrates the sublevel filtration for our example image.

Figure 2.13 shows the resulting 0-th and 1-st dimensional persistence diagrams of the sublevel filtration $\bar{\mathcal{F}}(f_I)$. We observe that the persistence diagram captures topological information of the objects displayed by the image I. In this case, this corresponds to the two 'objects' displayed by the image I (the '1' and the '8') in case of the 0-th dimensional persistence diagram, and the two cycles in the '8' in case of the 1-st dimensional persistence diagram.

The fact that persistent homology allows one to effectively capture topological information encoded through the sublevel filtrations of real-valued functions, is a consequence of the stability result below (Theorem 2.5.18), for which we first require another definition. Note that the following definition may be unclear without having read the start of Section 2.5.2, but one should not bother much with it for the purpose of this thesis. More specifically, one may assume that all functions we consider within this thesis (and also the majority of those considered in data science) are defined on a *triangulable topological space*, and satisfy the following condition.

Definition 2.5.17. *A real-valued function f on a topological space is called* tame *if it has a finite number of* homological critical values *(the times t at which the homology modules of $\bar{\mathcal{F}}(f)$ change), and $\mathcal{H}_k\left(\bar{\mathcal{F}}_t(f)\right)$ is finitely generated for all $k \in \mathbb{N}$ and $t \in \mathbb{R}$.*

Theorem 2.5.18. *[53, Main Theorem] Let (X, τ) be a triangulable topological space with continuous tame functions $f, g : X \to \mathbb{R}$. Denoting $\mathrm{Dgm}_l\left(\bar{\mathcal{F}}(h)\right)$ for the l-th persistence diagram of the sublevel filtration $\bar{\mathcal{F}}(h)$ of a function h, it holds that*

$$d_{\mathrm{b}}\left(\mathrm{Dgm}_l\left(\bar{\mathcal{F}}(f)\right), \mathrm{Dgm}_l\left(\bar{\mathcal{F}}(g)\right)\right) \leq \|f - g\|_\infty.$$

It is important to note the occurrence of the maximum norm $\|\cdot\|_\infty$ in the stability result of Theorem 2.5.18. This implies that when we conduct many but small perturbations on the functional values of a function f, the persistence diagram of its sublevel filtration would hardly change in terms of the bottleneck distance. However, if one would significantly alter one function value of f, then the persistence diagram may significantly change as well. In the case of our example image I, this can be interpreted as follows. If we would add a small amount of uniform or Gaussian noise to our image, we would still be able to infer the presence of two components and two cycles from the resulting persistence diagram. However, if we would significantly darken the single top left pixel in I, then we would suddenly infer the presence of three connected components.

Finally, observe that persistent homology of metric spaces can also be regarded as persistent homology of sublevel filtrations. Indeed, we can define a function f that maps each subset S of a finite metric space (X, d) to the diameter of S, i.e., $\max_{x,y\in S} d(x, y)$. The Vietoris-Rips filtration on X then coincides with the sublevel filtration on the abstract simplical complex Δ that contains all subsets in X of finite diameter (possibly constrained by some specified dimension). Hence, in this sense, persistent homology through sublevel filtrations is the most general formulation of all three presented in Sections 2.5.2.2-2.5.2.4.

2.6 Mapper

In this section, we present the *mapper* algorithm [17], which is a well-known tool from the field of TDA that is closely connected to graphs.

Consider a point cloud data set $X \subseteq \mathbb{R}^d$. The first step of the mapper al-

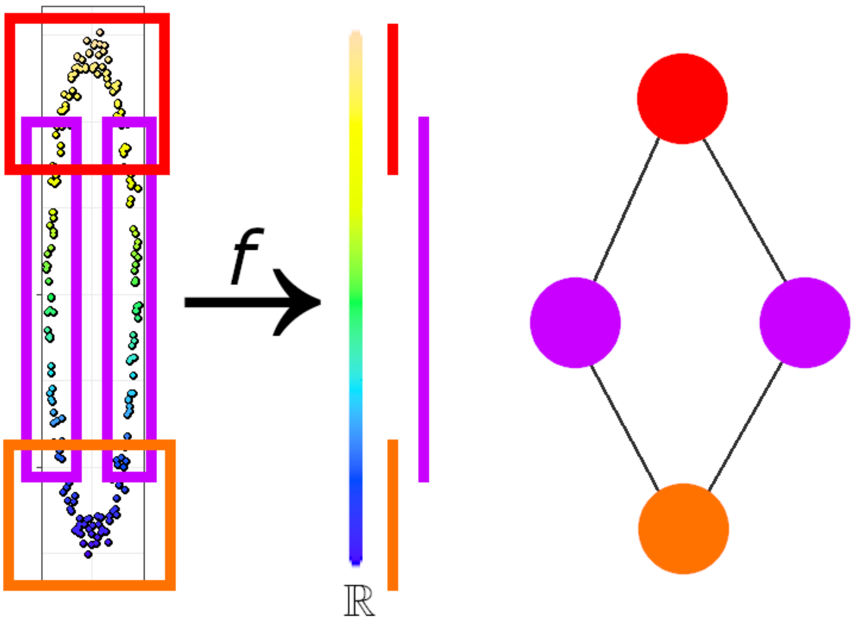


Figure 2.14: The mapper algorithm for an example data set X resembling an ellipse in $\mathbb{R}^2$. First, X is mapped to $\mathbb{R}$ by the filter f (in this case the height function on X). Next, an overlapping collection of three bins is constructed in $\mathbb{R}$. The preimage of f is then clustered separately for each of these bins. The final output of the Mapper algorithm is a graph of which the vertices equal all the obtained clusters, and where two distinct clusters are connected by an edge if they have a nonempty intersection. The mapper algorithm represents the topological structure underlying X well in this example.

gorithm is to fix a *filter* function $f : X \to \mathbb{R}^{d'}$ (also known as a *lens*), where usually $d' \in \{1, 2\}$. These filters often correspond to standard dimensionality reduction methods, such as PCA projections. Next, a collection of overlapping d'-dimensional cubes $(U_1, \ldots, U_m)$, also called *bins*, covering $\mathrm{Im}(f)$ is constructed in $\mathbb{R}^{d'}$. A standard clustering algorithm, such as single linkage clustering [54], is then applied to each preimage $f^{-1}(U_i)$, $1 \leq i \leq m$, resulting in k_i clusters $C^i_1, C^i_2, \ldots, C^i_{i_{k_i}} \subseteq X$. This results in a total of $K = \sum_{i=1}^{m} k_i$ clusters $C'_1, \ldots, C'_K$. The mapper algorithm then outputs an abstract simplicial complex $\mathbb{M}$, which contains a vertex (0-simplex) $\{v_i\}$ for each index $i \in \{1, \ldots, K\}$, as well as a k-simplex spanned by the distinct indices $i_0, \ldots, i_k$, $k \geq 1$, whenever $U_{i_0} \cap \ldots \cap U_{i_k} \neq \emptyset$. Mapper mainly serves as a visualization tool for point cloud data, so that one often restricts the simplicial complex to 0- and 1-simplices (the output is then a graph) . Figure 2.14 illustrates the Mapper algorithm through an example data set X, for which we used the height function, i.e., the y-coordinate, as the filter f.

For completeness, we include the way the mapper algorithm is formalized below. Note that this is not important for the rest of this thesis however.

Definition 2.6.1. *(Nerve). Let (X, τ) be a topological space and $\mathcal{U} = \{U_i\}_{i \in I}$ an open covering of X. The* nerve *of $\mathcal{U}$ is defined as the abstract simplicial complex $\check{N}(\mathcal{U})$, which contains a vertex 0-simplex $\{v_i\}$ for each $i \in I$, and a k-simplex $\{v_{i_0}, \ldots, v_{i_k}\}$, whenever $U_{i_0} \cap \ldots \cap U_{i_k} \neq \emptyset$ for distinct indices $i_0, \ldots, i_k$, $k \geq 1$.*

Definition 2.6.2. *(Pullback). Let (X, τ_X) and (Y, τ_Y) be topological spaces, and $f : X \to Y$ a continuous function (in the context of mapper, these functions are called filters or lenses). Let $\mathcal{U} = (U_1, \ldots, U_m)$ be an* open covering *of Y, i.e., a collection of open sets in Y such that $\bigcup_{i=1,\ldots,m} U_i = Y$. Then $f^\star\mathcal{U} := (f^{-1}(U_1), \ldots, f^{-1}(U_m))$ is an open covering of X, called the* pullback *of $\mathcal{U}$ under f.*

Remark 2.6.3. *Note that the pullback cover $f^\star\mathcal{U}$ in the definition above is indeed an open covering of X. First, it consists of open sets by Definition 2.3.3. Second, for every $x \in X$, we must have $f(x) \in U_i$ for at least one $1 \leq i \leq m$, so that $x \in f^{-1}(U_i) = \{z \in X : f(x) \in U_i\}$. This shows that $\bigcup_{i=1,\ldots,m} f^{-1}(U_i) = X$.*

Definition 2.6.4. *(Cluster Function). Let (X, τ) be a topological space. A function*

$$\pi : \tau \to \mathcal{P}(\tau) : U \mapsto \pi(U),$$

with the following properties:

- *for each $U \in \tau$, it holds that $\bigcup_{V \in \pi(U)} V = U$,*
- *for each $U \in \tau$, it holds that $V \cap V' = \emptyset$ whenever $V, V' \in \pi(U)$ and $V \neq V'$,*

is called a cluster function *or* cluster algorithm. *Given a family $\mathcal{U} = \{U_i\}_{i\in I}$ of open sets in X, we define another family of open sets $\pi_\star(\mathcal{U}) := \bigcup_{i\in I}\pi(U_i)$.*

Definition 2.6.5. *(Mapper). Let (X, τ_X) be a topological space. Suppose π is a cluster algorithm on X, $f : X \to Y$ a continuous function to a topological space (Y, τ_Y), and $\mathcal{U} = (U_1, \dots, U_m)$ an open covering of Y. The result of mapper applied to this triple is the simplicial complex*

$$\mathbb{M}(\pi, f, \mathcal{U}) := \check{N}(\pi_\star(f^\star\mathcal{U})).$$

Definition 2.6.5 can both be regarded in a practical setting, where X is a finite point cloud data set, or in a setting where X defines a (underlying) continuous topological model. In the first case, every subset of X is open (according to the topology induced by the metric space X is embedded in). A clustering algorithm (in the usual context) that joins distinct points together is then necessary. If X defines a (continuous) topological space (X, τ), e.g., the topological model underlying our observed data, then one may naturally take the clustering function π to map each $U \in \tau$ to the set of connected components of the subtopology induced on U.

The Mapper algorithm allows for a lot of flexibility to visualize data. Unfortunately, this is accompanied by a sensitivity to the used parameters, as the Mapper algorithm lacks robustness against the choice of filter, the amount of overlap of bins, as well as the clustering method in the original space. Any change in these choices can lead to a major change of the output of the algorithm [55]. A potential solution to this issue was introduced in [56], under the name of *multiscale mapper*. Here, one studies the resulting simplicial structures and maps between them, when the mapper algorithm is conducted for varying coverings of the image of the filter.

2.7 Merge Trees

We end this chapter by presenting the concept of *merge trees* [57], which will be used to conduct one of our proofs in Chapter 6. First, we require to introduce the concept of a *Reeb graph*.

Definition 2.7.1. *(Quotient Space). Let (X, τ_X) be a topological space and $\sim$ an equivalence relation on X. The* quotient set *$Y = X/\sim$ is the set of equivalence classes of elements of X. We denote the equivalence class of $x \in X$ as $[x]$. The* quotient space *under $\sim$ is the set Y equipped with the* quotient topology

$$\tau_Y := \{U \subseteq Y : \{x \in X : [x] \in U\} \in \tau_X\}.$$

Definition 2.7.2. *(Reeb Graph). Let (X, τ) be a connected topological space, and f a continuous scalar function $f : X \to \mathbb{R}$. We define an equivalence relation $\sim$*

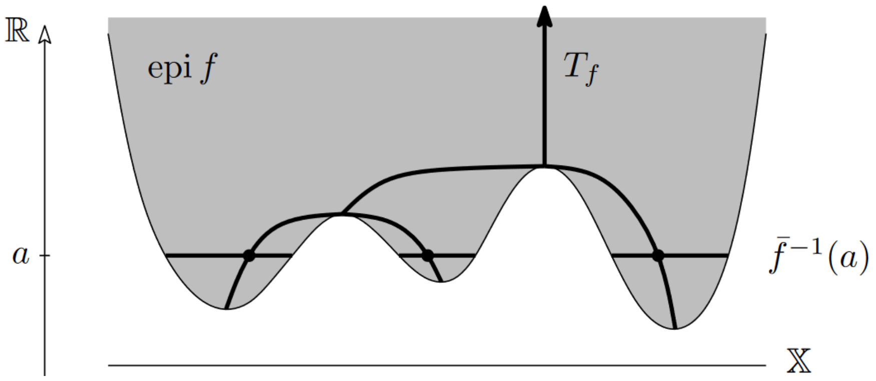


Figure 2.15: The graph of a function f together with its epigraph epi$f$ *and merge tree T_f. Three components of a levelset $\overline{f}^{-1}(a)$ of the projection $\overline{f}$: epi$f \to \mathbb{R}$ are highlighted in bold, together with their representative points on T_f. Image adapted with permission from [57].*

on X, such that for $x, y \in X$, $x \sim y$ iff there exists $a \in \mathbb{R}$ such that x and y belong to the same connected component of the level set $f^{-1}(a) := \{z \in X : f(z) = a\}$. *The* Reeb graph *is the quotient space $X/\sim$ endowed with the quotient topology.*

Note that the Reep graph is technically not a graph. However, the term is quite common in the TDA literature.

Definition 2.7.3. *(Merge Tree [57]). Let (X, τ) be a connected topological space, and f a continuous scalar function $f : X \to \mathbb{R}$. The* epigraph *of f is defined as*

$$\text{epi}f := \{(x, y) \in X \times \mathbb{R} : f(x) \leq y\}.$$

The projection from the epigraph onto $\mathbb{R}$ is defined as

$$\overline{f} : \text{epi}f \to \mathbb{R} : (x, y) \mapsto y.$$

The merge tree *of f, denoted T_f, is the Reeb graph of $\overline{f}$. We denote by $\hat{f}$ the function*

$$\hat{f} : T_f \to \mathbb{R} : [(x, y)] \mapsto y.$$

Note that $\hat{f}$ is a well-defined function. Indeed, if $[(x, y)] = [(x', y')] \in T_f$, then (x, y) and (x', y') belong to the same connected component of some level set $\overline{f}^{-1}(a)$. This means that $y = a = y'$. Figure 2.15 illustrates an example of a merge tree for a real-valued scalar function f.

For a projection $\overline{f}$: epi$f \to \mathbb{R}$, $a \in \mathbb{R}$, and $\epsilon > 0$, we have $\bar{\mathcal{F}}_a\left(\overline{f}\right) \subseteq \bar{\mathcal{F}}_{a+\epsilon}\left(\overline{f}\right)$, where $\bar{\mathcal{F}}_a\left(\overline{f}\right)$ is the sublevel set

$$\bar{\mathcal{F}}_a\left(\overline{f}\right) := \left\{(x, y) \in \text{epi}f : \overline{f}((x, y)) \leq a\right\}$$

of $\overline{f}$. Hence, any connected component in $\bar{\mathcal{F}}_a\left(\overline{f}\right)$ maps into a connected component of $\bar{\mathcal{F}}_{a+\epsilon}\left(\overline{f}\right)$.

Similar to the bottleneck distance between the persistence diagrams obtained through the sublevel filtrations of functions (Section 2.5.2.4), we can define a 'topological' distance between functions through their merge trees. This is formalized through the concept of *shift maps*.

Definition 2.7.4. *(Shift Map [57]). Let T_f be a merge tree. The* ϵ-shift map *$\iota^\epsilon : T_f \rightarrow T_f$ is defined by mapping $x \in T_f$ with $\hat{f}(x) = a$ representing a connected component $X \subseteq \bar{\mathcal{F}}_a\left(\overline{f}\right)$, to the point $\iota^\epsilon(x) := y \in T_f$ which represents the connected component in $\bar{\mathcal{F}}_{a+\epsilon}\left(\overline{f}\right)$ that includes X. In other words, $\iota^\epsilon(x)$ is obtained by following the path from x to the root of T_f until we reach a point y for which $\hat{f}(y) = a + \epsilon$.*

Definition 2.7.5. *(Interleaving Distance between Merge Trees [57]). Let T_f and T_g be merge trees. Two continuous maps $\alpha : T_f \rightarrow T_g$ and $\beta : T_g \rightarrow T_f$ are said to be* ϵ-compatible *for some $\epsilon \geq 0$, if*

$$\begin{aligned} \hat{g}(\alpha(x)) &= \hat{f}(x) + \epsilon, & \hat{f}(\beta(y)) &= \hat{g}(y) + \epsilon, \\ \beta \circ \alpha &= \iota^{2\epsilon}_{T_f}, & \alpha \circ \beta &= \iota^{2\epsilon}_{T_g}, \end{aligned}$$

where $\iota^{2\epsilon}_{T_f}$ and $\iota^{2\epsilon}_{T_g}$ denote the 2ϵ-shift maps in the respective merge trees. The interleaving distance *$d_{\mathrm{I}}(T_f, T_g)$ is defined as the infimum of ϵ for which there are ϵ-compatible $\alpha : T_f \rightarrow T_g$ and $\beta : T_g \rightarrow T_f$.*

The interleaving distance is no less sensitive than the bottleneck distance between 0-dimensional persistence diagrams, which is formalized through the following theorem.

Theorem 2.7.6. *[57, Theorem 3]. Given two tame functions $f : X \rightarrow \mathbb{R}$ and $y : Y \rightarrow \mathbb{R}$, it holds that*

$$d_{\mathrm{b}}\left(\mathrm{Dgm}_0\left(\bar{\mathcal{F}}(f)\right), \mathrm{Dgm}_0\left(\bar{\mathcal{F}}(g)\right)\right) \leq d_{\mathrm{I}}(T_f, T_g).$$

3 Introduction to Topological Models in Graphs

3.1 Introduction

The topic of this entire thesis is to study and perform topological inference *in* graphs. Indeed, for our purpose in Chapter 7, i.e., (topological) object detection in images, we will not require the inclusion of any higher-dimensional simplices than nodes and edges in the simplicial complex identified with the image (Section 2.5.2.4), so that they may be regarded as (functions defined on) graphs. Furthermore, in case of point cloud data, we will always construct an intermediate proximity graph, which we will use to perform topological inference or analysis.

With the only exception being object detection in images, we will also study and perform topological inference *of* graphs. In this chapter, *we will present this problem on an intuitive level*. Note that we will not provide a concrete theoretical formalization in this chapter (nor in this thesis) of what a topological model in a graph is. Nevertheless, throughout this thesis, we will show that such models are clearly present in many different graphs, whether given or derived from point cloud data.

In Section 3.2, we illustrate a first example of a *non-causal* topological model in graphs. In Section 3.3, we introduce the problem of *cell trajectory inference*, which will play an important role in Chapters 4-6. Similar to the earthquakes data set illustrated on the cover of this dissertation, cell trajectory data leads to *causal* topological models in graphs. Both types of models will be of interest throughout this thesis. Yet, this inconsistency in the dependency between models and graphs is one of the reasons why it is difficult to provide a mathematical formalization that applies to the majority of graphs. We discuss this further in Section 3.4.

3.2 A Topological Model in the Karate Network

We start by illustrating a first example of a (graph-structured) topological model in a graph. Consider the Karate network G in Figure 3.1a. According to the ground truth (see also the caption of Figure 2.2), the administrator John A and instructor Mr. Hi play important roles in the network. This can also be visually deduced from Figure 3.1a, as all members (nodes) in G lie close to (or more specifically, within at most two hops from) either John A or Mr. Hi. The ground truth separates two communities in the network, of which John A and Mr. Hi are the core members, defining important 'landmarks' in G. These communities are identified through the coloring of the nodes in Figure 3.1a.

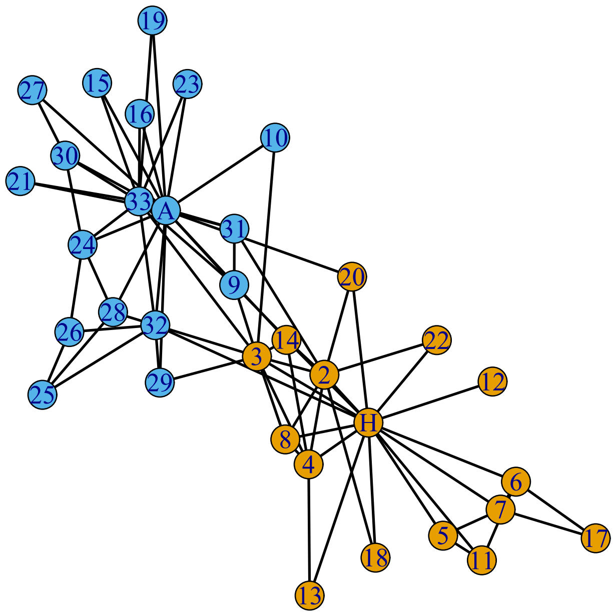
The difference between topological model inference and community detection in graphs lies in the fact that not only are we interested in identifying important landmarks or communities in the graph, but also in the present 'flow' between them. Indeed, although the two separate nodes for John A and Mr. Hi might model the karate network relatively well in some sense (e.g., in terms of distances to all other nodes), this would not model that there exists a connecting path or flow between them. In general, this flow induces connections between different landmarks in our graph, and the landmarks together with the connections between them then define a new graph B. The topology of B—which we identify with the topology of one of its geometric realizations—then models the *underlying topology* of the original graph.

We can now proceed in two ways to model the underlying topology of a graph (such as the Karate network) G, both leading to the same model up to a homeomorphism.

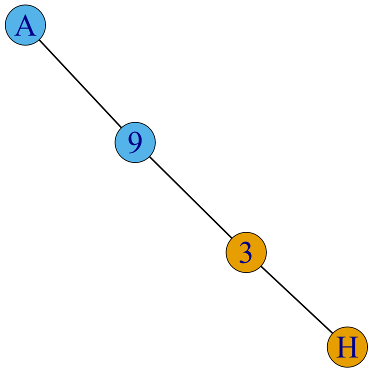
- **Subgraph models** model the flow between landmarks in G through paths in G itself (Figure 3.1b). The result is always a subgraph B of G.

- **Non-subgraph models** model the flow between landmarks in G through direct links between them (Figure 3.1c). This results in an entire new graph B, which has the landmarks in G as it nodes, and a link between two landmarks whenever there is a flow between them that does not pass through any other landmark.

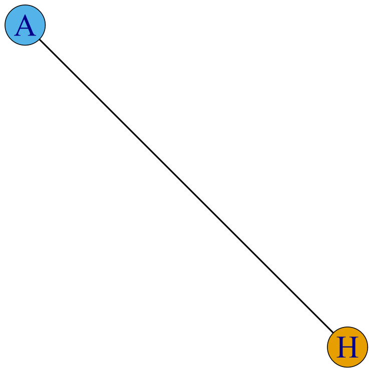
Naturally, subgraph models—which we also call *backbones*—encode more information. Instead of just knowing that there exists a link between two landmarks, we know an exact path between them. However, the difficulty lies in deciding on a



(a) The karate network.



(b) A subgraph topological model of the karate network.



(c) A non-subgraph topological model of the karate network.

Figure 3.1: Two topological models for the Karate network.

'good' path to model the flow. Inferring such models for given graphs will be the topic of Chapter 5. *Reconstruction* methods that only model links between landmarks, and hence, result in non-subgraph models, will be discussed in Chapter 4, in which we target point cloud data.

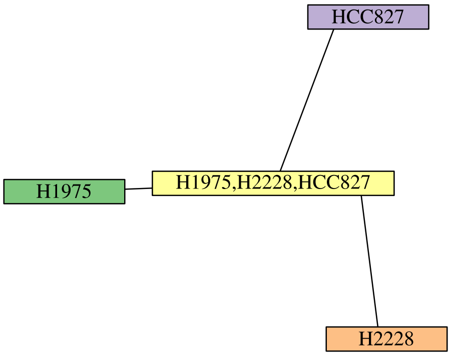
The Karate network is a good example of a *non-causal* topological model in graphs: the (linear) model is derived from the graph (data) itself, and the model did not generate the data. Intuitively, the entities (members of the karate club) existed prior to the model.

3.3 Introduction to Cell Trajectory Inference

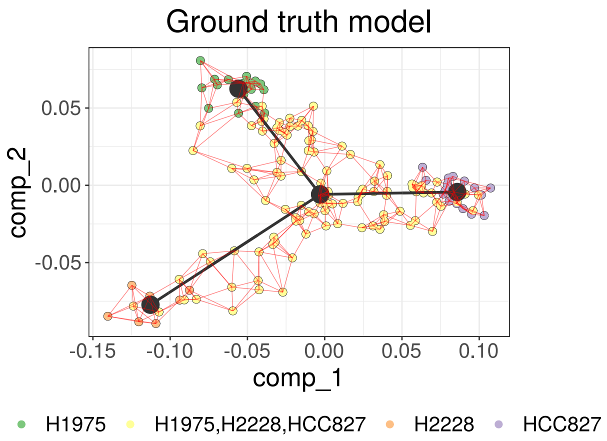
Cell trajectory inference might currently be considered one of the most crucial applications of (graph-structured) topological model inference in graphs, finding usage within domains such as cancer research and immunology [58–62]. One will therefore find we consider this problem quite a few times throughout this thesis (mainly in Chapters 4-6). Note that some methods for cell trajectory inference might not treat this as a topological inference problem in graphs, but may use the extrinsic Euclidean space in which the data is embedded [63]. However, throughout this thesis—and similar for all point cloud data we will consider—our first step will be constructing a proximity graph from the cell trajectory data, so that we may indeed treat this as a topological inference problem in graphs.

A *cell trajectory data set* X may be regarded as a data matrix in $\mathbb{R}^{n \times d}$. Each of the n rows in X corresponds to one particular biological cell, from which expression data is gathered through one of various possible *single cell sequencing methods* [64]. This expression data captures how much a particular gene or protein is expressed by a particular cell. The d columns of X thus represent the genes or proteins for which measurements are obtained. Hence, each entry X_{ij}, $1 \leq i \leq n$, $1 \leq j \leq d$, quantifies how much gene/protein j is expressed by cell i. Since often measurements of hundreds to thousands of genes or proteins are obtained at once, d is usually large, resulting in high-dimensional point cloud data in $\mathbb{R}^d$. Raw expression data generally requires a tremendous amount of pre-processing, such as quality control, normalization, data correction, feature selection, and dimensionality reduction, prior to its downstream analysis [9]. However, throughout this thesis we will only make use of dimensionality reductions, as the expression data we will consider have already undergone the various other steps [65].

In cell trajectory inference, the expression data X is gathered from *differentiating cells*, i.e., biological cells that evolve from one state into another. Each cell in X is then at some particular point during this biological *differentiation process*, which may be modeled through a graph G. During differentiation, functional changes of a cell correspond to changes in its expression profile. These changes may be regarded continuous for our purpose. This results in the point cloud data



(a) The ground truth model (a graph G) underlying the cell trajectory data.



(b) The ground truth model (a metric graph $\mathcal{M}$ in black) underlying the cell trajectory data, as well as the data itself, visualized through a diffusion map embedding in $\mathbb{R}^2$. A 5NN graph (Definition 2.4.6, edges in red) is constructed from this embedding, and we consider the ground-truth models of the point cloud data and graph constructed thereof to be equal.

Figure 3.2: An example of a real cell trajectory data set X, along with its underlying models represented as both a graph G and metric graph $\mathcal{M}$.

$X \in \mathbb{R}^{n \times d}$ approximating a metric graph $\mathcal{M}$—which is a geometric realization of G—in $\mathbb{R}^d$.

Figure 3.2 shows an example of a cell trajectory data set X. The underlying graph model G, which displays the differentiation process the cells undergo (each node represents one 'milestone' of this process), is shown in Figure 3.2a. Figure 3.2b shows the geometric realization $\mathcal{M}$ of G, as well as its location within the data set X. However, as the original data X is high (1770-)dimensional, we visualized both the data as well as its ground-truth (metric graph) model in Figure 3.2b through a two-dimensional diffusion map embedding [66] of X. The 2D coordinates resulting from this embedding correspond to the 'comp_1' and 'comp_2' axes in Figure 3.2b.

The main purpose of cell trajectory inference is to infer the model $\mathcal{M}$ from the expression data X. As discussed in Section 2.4.2, topological information of G corresponds to topological information of $\mathcal{M}$ and vice versa. The difference is that knowledge of $\mathcal{M}$ allows one to map G, i.e., place the cell differentiation network, into the original data X, which is something cell trajectory inference is also concerned with (e.g., for *pseudotime analysis* [9]).

Cell trajectory data, and therefore proximity graphs constructed thereof (an example is shown in Figure 3.2b), are good examples of *causal* relationships between the topological models and the (proximity) graphs derived from the data. More specifically, changes in gene or protein expression during cell differentiation may be modeled using ordinary differential equations [67], which define a metric graph in the high-dimensional expression space to which the observed cells are—accounting for the presence of noise which is quite common for this type of data—-expected to lie close to. Intuitively, it is the cell differentiation process that determines the positioning of cells, resulting in the observed data. Similarly, the union of the boundaries of the tectonic plates can be regarded as a causal model for (a proximity derived from) the earthquakes data set on the cover of this dissertation, as the friction between them is one of the main causes for the occurrence of earthquakes.

3.4 On Formalizing Underlying Models in Graphs

Up to this point, we have seen various examples of graph-structured topological models in graphs. In this section, we will describe these models further on an *intuitive level*, and discuss both the possibilities and difficulties when it comes to mathematically formalizing these models. Note that we will not provide such exact mathematical formalization of these models in this section (hence the 'On' in this section's title), nor in this thesis. We will therefore put a lot of emphasis on qualitative analysis and visualization to understand these models.

In Section 3.4.1 we discuss the intuitive properties we seek for in topological models of graphs. Finally, in Section 3.4.2 we discuss the various difficulties when it comes to expressing the mathematical relationship between the topological models and graphs.

3.4.1 Properties of a Topological Model in a Graph

We have seen that graph-structured models may occur both in *given* graphs, such as in the Harry Potter or Karate network, or in graphs *derived* from point clouds, such as earthquake locations or cell trajectory data. For point clouds with an extrinsic space, the model can be both interpreted in a graph-theoretical sense, as well as in a more 'continuous' topological sense, i.e., as a metric graphs. Graph-theoretical 'topological' properties (such as paths, components, cycles, ...) then coincide with their formal topological counterparts for these models. Since given graphs may not have such extrinsic space or model, we will describe our models as graphs themselves in all cases. Despite this confusing terminology, i.e., the original graph (data) is essentially a graph-structured topological model itself, all graphs we have seen above may be *characterized well* through a more *simplified model*. By 'characterized well', we mean that

1. the majority of nodes lie close to the model,
2. the model preserves the geometry of the original graph well,

whereas by 'simplified model' we mean that

3. the size of the model, i.e., in terms of number of vertices and/or edges it includes, is much smaller than that of the original graph,
4. the model is significantly less *topologically complex*, i.e., has less leaves, multifurcations, and cycles, than the original graph.

Note that essentially 4. is the only *topological* characteristic. However, throughout this thesis, properties 1.-4. will be the ones we seek for in topological models of graphs. Thus, apart from a model that only displays the topological properties, we aim for a model that describes the entire original data well. E.g. Figure 4.4 illustrates well why this may be different.

Each one of these properties admits direct, and often multiple ways to be mathematically formalized. Yet, aiming to optimize each one of these may lead to generalization difficulties, which we will further clarify in Section 5.7.4. E.g., for 1., one might consider a cost that equals a sum, average, mean, or maximum of the distances (either in the graph or in the extrinsic space) of all nodes in the original graph (data) to the model, aiming to achieve a low such cost. However, as we will see in Chapter 5, such optimization may lead to ineffective results when outliers are present, or when the data has a nonuniform density. Furthermore, clearly the original graph (data) will achieve the lowest cost for 1., and we would need to impose that the additional simplification properties (3.-4.) are satisfied. However, the fact that we want our models to simultaneously satisfy properties 1.-4., that we want to accommodate for outliers or nonuniform density, that infinitely many topologies may be associated with (geometric realizations of) finite graphs, and that we consider such models in both given graphs and graphs derived from point cloud data, makes it complicated for one to write down one simple mathematical expression (or even a class of such expressions) associating a cost—to be optimized—to a model, that effectively applies to the majority of graphs.

3.4.2 Expressing the Relationship between the Model and Data: the Difficulties

Properties 1.-4. in Section 3.4.1 intuitively capture what we want a topological model in a graph to satisfy. Yet, they do not tell us what a topological model in a graph exactly is. More specifically, they do not specify how such model relates to the original graph, e.g., in terms of dependencies, causality, or a generating process. In this section, we discuss the difficulties when it comes to providing such formalization that consistently applies to the majority of graphs occurring in data science.

Indeed, one may consider mathematical expressions that specify how the graph (data) is the probabilistic result of some (topological) model. This may be directly possible for graphs derived from point cloud data where a ground-truth model (metric graph) is present, as has been partially addressed in [6]. For given graphs, one may consider *random graphs* [68, 69], which are obtained by starting with a set of n nodes and adding successive edges between them at random. These graphs are described by a probability distribution, or by a random process which generates them. Hence, one might capture the existence of a ground-truth topological model within the mathematical formulas expressing the corresponding probabilities or data generating process. To the best of our knowledge, random graphs have not yet been characterized in terms of ground-truth topological models in such a way that captures properties 1.-4. in Section 3.4.1. Furthermore, another question that would arise from such formalization through random graphs is whether one would treat given graphs and those derived from point clouds separately, or whether one would aim for a formalization that is consistent for both (and prove that this is indeed the case, e.g., starting from the results in [6]). Other difficulties that further complicate providing a universal formalization of topological models through random graphs—even when graphs derived from point cloud data would be treated separately—are summarized below.

Causal vs. non-causal The question whether there is a causal relationship between the topological model and the original graph relates to the question whether there is a ground-truth topological model. Nevertheless, the answers to these questions may be different. E.g., in case of point cloud data (and hence, proximity graphs derived thereof), there is often a causal relationship between the observed data and the underlying ground-truth model, such as we discussed for the earthquake locations and cell trajectory data. However, there may not always be an analogous interpretation for every type of graph considered. A clear example of this is the Karate network, where the entities existed prior to the topological model we derived (Section 3.2). However, even though there might not be a causal relationship between the model and data in this case, our inferred linear model does fit the ground-truth metadata, i.e., the given community classes well (Figure 3.1). Then there are other examples for which the existence of such causal relationship may be ambiguous. E.g., in case of the Harry Potter network, J.K. Rowling (the author) might have first thought of a 'ground-truth' model, which captures that there are two communities (nodes), representing the 'good' and 'evil' characters, and some possible connection between them. A (non-subgraph) model similar to the one in Figure 3.1b may hence capture the ground truth of the Harry Potter network well. One may then argue whether the actual characters were derived from this model. Even so, providing a mathematical formalization that expresses how J.K. Rowling proceeded to do this will be likely tedious.

We summarize these difficulties—through which what we consider one of the most important phrases in this entire thesis—as follows.

simplified graph-structured models occur naturally in many real-world graphs, but conversely, many graphs are not the causal result of such model.

This also complicates providing a universal formalization of topological models through random graphs, where the corresponding probabilistic rules would express the dependency between the model and data.

Small-world vs. non-small-world The Karate network is an ideal example of one of the many real-world graphs that satisfy the *small-world network* model [70], which states that the number of highly connected nodes—termed *hubs*—is much smaller than the number of low degree nodes. This means that most nodes in small-world networks are not neighbors of each other, but are likely to have common neighbors. More formally, a small-world network is defined as a network G where the expected number of hops between two randomly chosen nodes u and v grows proportionally to the number of nodes in G, i.e.,

$$\mathrm{E}\left[d^{\mathrm{unw}_G}(u,v)\right] \propto \log|V(G)|.$$

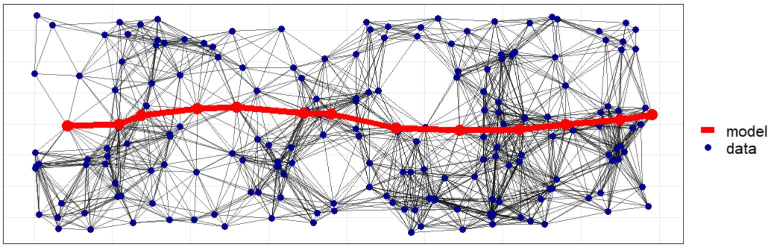


Figure 3.3: A graph G (nodes in blue) and a subgraph B (red) of G which truthfully represents the linear topology of G.

The *six degrees of separation* is an idea that is also based on the concept of small-world networks, and states that the set of all people (alive at one time) are six, or fewer, social connections away from each other.

One might deduce that the model underlying a graph G is always induced by the hubs of G and the flow between them (Section 3.2). However, this is generally not true, as there are many examples of non-small-world networks in which simple graph-structured models are present as well, which do not even include hubs. Intuitive examples are road networks, where nodes correspond to intersections and edges to streets between them (we should be rather grateful that no hubs are present in these networks). Given two arbitrary locations, there are likely many roads between them, but you are likely to spend most of your time traversing only the highways (which make up the model underlying the road network) between them. Another example we have already seen are earthquake locations. There is no location on Earth where many boundaries of tectonic plates meet, and the highest degree of the corresponding model is small. In general, proximity graphs constructed from low-dimensional point cloud data are examples of non-small-world networks. Examples other than the earthquake locations with an underlying (simplified) graph-structured model include galaxy locations distributed in space [13,29], or low-dimensional embeddings of high-dimensional cell trajectory data [9]. An illustrative example is given in Figure 3.3, which shows a Rips graph G constructed on a noisy point cloud sampled from a line segment in $\mathbb{R}^2$. Clearly, there is no strong correlation between the degree of a node and its closeness to the underlying linear model centered in G.

This suggests that when formalizing topological models in graphs through random graphs, one may need to consider different ways to express how the connections in the graph depend on the nodes in the model.

What nodes in the model represent Nodes—and edges between them—within the topological model of graphs have an intuitive meaning on a *global* level: they represent the existence of important connections in the underlying topology. Yet,

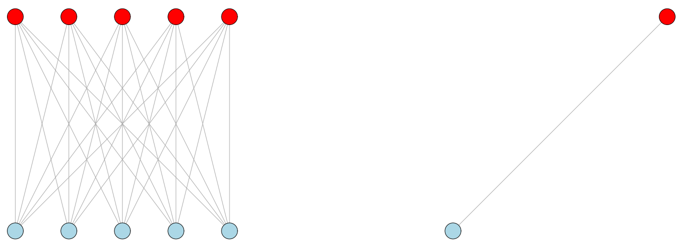

(a) A complete bipartite graph G. The layout is obtained through a bipartite layout algorithm. *(b) A possible topological representation of G.*

Figure 3.4: A possible topological model in a complete bipartite graph $G = (V = A \sqcup B, E)$. The topological model captures the flow from A to B. Yet, the nodes in G represented by the same node in the model are more distant to each other.

they have a less consistent meaning on a *local* level. I.e., in case of the models we discussed up to this point, the nodes in the original graph represented by those in the model—such as Harry Potter who represents Ron or Hermione in the Harry Potter Network (Figure 2)—share a common property. In all these cases, nodes in the original graph represented by the same node in the model can be regarded close to each other. This is similar to the model in the Karate network (Section 3.2). However, such interpretation may be invalid for other types of graphs. To see this, consider an unweighted *complete bipartite graph* $G = (V, E)$, where V is the union of two nonempty disjoint sets A and B, i.e., $V = A \sqcup B$, and $\{u, v\} \in E$ iff $u \in A$ and $v \in B$. An example of such graph is shown in Figure 3.4. One meaningful topological model may be obtained my modeling the flow from A to B, such as in Figure 3.4a. In this case, all nodes in A are represented by the same node in the model, yet, are more distant to each other than they are to nodes in B.

Hence, when one would formalize topological models in graphs through random graphs, one may need to decide what nodes in the topological model actually represent, and account for the possibility that nodes represented by the same node in the model must not be close to each other.

3.5 Discussion and Conclusion

In this chapter, we provided a first introduction to topological models in graphs. Both the Karate network and our considered cell trajectory data set (and hence, a proximity graph derived thereof, which has the same underlying model by assumption) could be modeled well by a simple graph-structured model. Despite the confusion terminology, i.e., both the original graph and model can be considered graph-structured models, these concepts were easy to understand through these visual examples.

We furthermore expressed the properties we seek for in topological models of graphs in Section 3.4.1. Furthermore, we discussed that each of these properties admits direct, and often multiple ways to be mathematically formalized, possibly resulting in a cost function to optimized. Although this may lead to well-defined optimization problems for inferring topological models in graphs, we discussed why it may be rather complicated for one to write down one simple mathematical expression (or even a class of such expressions) to effectively infer these models through an associated cost. We will elaborate on this in Chapter 5.

Unfortunately, things do not become easier when it comes to mathematically formalizing the relationship between graphs and their topological model. The wide variety in topological behavior of graphs makes it difficult to provide a universal approach that expresses these relationships in a way that applies to even the majority of graphs. Graphs commonly differ in terms of whether they are given or derived from point cloud data, whether there is a causal relationship between them and the model, whether they satisfy the small-world network model, or how nodes in the model relate to the nodes they represent.

Nevertheless, as we have seen in this chapter and will extensively see in the following chapters, topological models can occur in all of such graphs. Furthermore, although the methods we develop in the following chapters might only make sense under the assumption that these models are present, they will also allow us to confirm that this is indeed the case. Hence, despite—and, perhaps even more interestingly, given—all the various difficulties discussed above, how to mathematically formalize the concept of topological models in graphs is one of the most important open problems resulting from this thesis. We will discuss this further in Section 8.2.

4 Methods from Local Topological Data Analysis

This chapter is based on the following publication.

- Robin Vandaele, Tijl De Bie, and Yvan Saeys. *Local Topological Data Analysis to Uncover the Global Structure of Data Approaching Graph-Structured Topologies*. In Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, and Georgiana Ifrim, editors, Machine Learning and Knowledge Discovery in Databases, pages 19–36, Cham, 2019. Springer International Publishing. [3]

4.1 Introduction

In this chapter we discuss *reconstruction methods* that build a non-subgraph topological model of a given metric space, based on *local topological information* [3, 13]. These metric spaces will be defined through proximity graphs constructed from Euclidean point cloud data approximating a metric graph. More specifically, reconstruction methods target Rips graphs constructed from local neighborhoods of proximity graphs, but generalize rather poorly to arbitrary given graphs.

In Section 4.2 we briefly discuss how these reconstruction methods relate to *persistent local homology*, through which many of the concepts introduced in this chapter can be mathematically formalized. Next, in Section 4.3 we discuss how multifurcations can be inferred through local topological data analysis, followed by cycles in Section 4.4. In Section 4.5 we discuss how this topological information on multifurcations and cycles can be used to reconstruct the global model from the data. Finally, in Section 4.6 we conclude upon the effectiveness of reconstruction methods for inferring graph-structured models, and discuss further directions for improving them.

We refer to the general method that deals with the analysis and inference of local topological properties in data as *local topological data analysis* (LTDA).

4.2 Relation to Persistent Local Homology

The idea of *persistent local homology* [71–73] is to apply persistent homology after discarding a local neighborhood $B_d(x,r)$ of a data point x from a given finite metric space (data set) (X,d). As we will see in the following sections, this is very similar to how local topological information is deduced for graph reconstruction methods. However, unlike these reconstruction methods, persistent local homology—similar to regular persistent homology—does not infer an exact topological property, but quantifies topological properties at many scales. On the one hand, we have the parameter α (such as the time t, distance ϵ, $\dots$) which parameterizes the filtration after having discarded a fixed local neighborhood $B_d(x,r)$. On the other hand, we may also vary the radius parameter r. This leads to a 1-parameter family of persistence diagrams, parameterized by r, also known as *persistence vineyard* or $(\alpha|r)$*-vineyard* [74]. These persistence vineyards can be seen as lines in the 3D space, where each line tracks the evolution of a particular diagram point according to the increasing radius parameter r.

Unfortunately, graph reconstruction methods currently require exact local topological information near every data point x. E.g., they require a 'yes or no' answer to the question if the local neighborhood of x corresponds to a multifurcation or leaf, instead of a topological summary from which this might be deduced. Furthermore, it is still unclear how this one may actually deduce exact topological information, i.e., one precise number of interest, from persistence vineyards. For this reason, we will discuss reconstruction methods based on local topological information inferred at *fixed scales*, which can also be considered their main limitations.

4.3 Locating Multifurcations in Metric Data

In this section, we consider the following problem. We are given a metric space (X,d), and a point $x \in X$. The underlying topology of X is known to be metric graph $\mathcal{M}$. In practice, X will be a (Euclidean) point cloud data set, and d will be obtained through the shortest path distances in a proximity graph G constructed

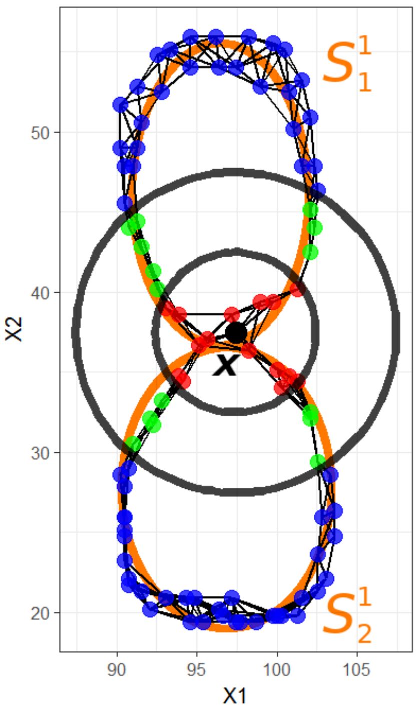


Figure 4.1: The idea behind LTDA for data that approaches a metric graph $\mathcal{M} = \mathcal{S}_1^1 \cup \mathcal{S}_2^1$. For appropriate proximity graphs, one finds the underlying degree of a data point x (black) by counting the connected components in the graph induced by the intersection of a spherical shell and the data (green points), representing branches emerging from x.

on X. The problem we then consider is *"to characterize the underlying topology of X near x*. Figure 4.1 illustrates how we proceed to this end. First we consider a ball $B_d(x, r)$ around X. By nature, the points within $B_d(x, r) \cap X$ (the red and green points in Figure 4.1) will then resemble one connected underlying topology $\mathcal{S} \subseteq \mathcal{M}$. If r is small enough, then furthermore S will be a 'star'-shaped topology. This means that S is equivalent to a topological space that is obtained by drawing $k \in \mathbb{N}$ branches away from a center point m, accounting for the fact S may be the isolated point m, i.e., $k = 0$ (we informally use the word 'branch' to refer to the continuous counterpart of an edge). m is the representative point for x in $\mathcal{M}$, and the local topology of X near x, i.e., $\mathcal{S}$, is topologically completely characterized by k, which equals the degree of m in $\mathcal{M}$ (Definition 2.4.7).

The question is then how to infer the number k in a data setting rather than a theoretical setting. The answer is based on the fact that in the theoretical model, k equals the number of connected components in the star-shaped topology $\mathcal{S}$, after removing the point m representing x. Accounting for the fineness of data and noise, we can hence infer k through a cluster algorithm on all points within a *punctuated neighborhood* $B_d(x,r)\backslash B_d(x,r')$ for some $r' < r$. In [13], the used cluster counting method corresponds to counting the number of connected components in the Rips graph $\mathcal{R}_{4r'/3}\left(B_d(x,r)\backslash B_d(x,r')\right)$, and they let $r = 5r'/3$, where an optimal choice of r' (if existing) depends on the scale of the data. In contrast to this, we impose that the (X,d) is derived as a Rips graph $G = \mathcal{R}_\epsilon(X)$ from some point cloud set X (e.g., as shown in Figure 4.1), where the scale of the data is captured through ϵ, and hence, through the unweighted shortest path distance metric on $d = d_G^{\mathrm{unw}}$. We then count the connected components in the subgraph in $\mathcal{R}_\epsilon(X)$ that is induced by the points in $B_d(x,r)\backslash B_d(x,r')$. This corresponds to the four components marked by the green points in Figure 4.1. We set $r = r'+1$, which in the unweighted case implies that $B_d(x,r)\backslash B_d(x,r') = \partial B_d(x,r')$, and observe that most often choosing $r' \in \{2,3\}$ leads to good empirical results. Naturally, other possible choices may be investigated as well.

Remark 4.3.1. *Neither our approach, nor the one in [13], is to be practically preferred over the other for inferring the degrees through LTDA. The intuition behind both methods is exactly the same. However, our approach is currently more heuristic. The choices for r and r′ have been theoretically justified in the sense that through these choices their full reconstruction method [13, Algorithm 1] is guaranteed to reconstruct the underlying ground truth metric graph $(\mathcal{M}, d_\mathcal{M})$ up to a homeomorphism, under certain assumptions [13, Theorem 1]. Furthermore, under these assumptions, they also provide a guarantee that the metric of the reconstruction approximates the metric $d_\mathcal{M}$ well [13, Theorem 2]. It is left to say that these assumptions are quite stringent, and quickly become impossible to satisfy whenever the metric distortion between (X,d) and $(\mathcal{M}, d_\mathcal{M})$ is too large relative to the shortest branch length of $\mathcal{M}$, e.g., as a result of the data set X containing too much noise. Nevertheless, even though geometric information is strictly stronger than topological information, geometric information can be captured well even when topological information is inferred incorrectly (see Section 6.2 and [6]). For this reason, we do not provide topological reconstruction guarantees as in [13].*

Remark 4.3.2. *There is no real contribution in our degree inference method over the presented in [13]. Our contributions lie in inferring cycles through comparing global to local topological information (Section 4.4), and our reconstruction method (Section 4.5).*

4.4 Comparing Local to Global Topological Information for Cycles

If $\mathcal{M}$ is a metric graph, then for any $x \in \mathcal{M}$, $B_\mathcal{M}(x,r)$ does not contain any cycle whenever $r > 0$ is small enough. Hence, cycles in (metric) graphs correspond to global and not local topological properties. However, we may reveal the presence of a cycle through x by comparing global topological information relative to local topological information near x. This is formalized through the following theorem.

Theorem 4.4.1. *Let $G = (V, E)$ be a graph. Then for each $\alpha \in V \cup E$, the number of cycles passing through α is bounded from below by*

$$\delta_1(\alpha) := \delta_0(\alpha) + \beta_0(G) - (\beta_0(G\backslash\alpha) + 1) \geq 0, \tag{4.1}$$

where $G\backslash\alpha$ denotes the graph G after removing α, and

$$\delta_0(\alpha) := \begin{cases} \delta(\alpha) & \text{if } \alpha \in V, \\ 2 & \text{if } \alpha \in E. \end{cases}$$

Moreover, for each $\alpha \in V \cup E$, a cycle passes through α iff $\delta_1(\alpha) > 0$.

Proof. Let $\alpha \in V \cup E$ and denote the number of cycles through α by $\beta_1(\alpha)$. Since the statement is trivially valid for $\beta_0(G) = 0$, we may assume G to contain at least one node, and hence, $\beta_0(G) \geq 1$. Furthermore, as the contributions of the components in G other than $[\alpha]$[1] to $\beta_0(G)$ and $\beta_0(G\backslash\alpha)$ in (4.1) cancel each other out, we may as well assume that $\beta_0(G) = 1$. Hence, it is left to prove that

$$\beta_1(\alpha) \geq \delta_1(\alpha) = \delta_0(\alpha) - \beta_0(G\backslash\alpha) \geq 0,$$

and that a cycle passes through α iff $\delta_0(\alpha) > \beta_0(G\backslash\alpha)$. This follows easily from the fact that each cycle through α passes through two neighbors of α which remain connected after discarding α, and conversely, if two neighbors of α remain connected after discarding α, then they induce a cycle through α in G. □

Remark 4.4.2. *As additional paths between two fixed neighbors of $\alpha \in V \cup E$ in $(G\backslash\alpha)$ do not change $\beta_0(G\backslash\alpha)$ more than one such path, $\delta_1(\alpha)$ only provides a lower bound on $\beta_1(G)$.*

Inferring δ_1 for a given metric space (X, d) in a data setting is very similar to inferring $\delta_0 = \delta$ (Section 4.3). Once δ has been inferred through the connected components constituted by the points in $B_d(x, r)\backslash B_d(x, r')$, we consecutively subtract the number of connected components constituted by the points in $\{y \in X : d(x, y) < \infty\}\backslash B_d(x, r')$. This corresponds to the two components marked by the blue and green points in Figure 4.1.

[1]The definition of connected components (Definition 2.4.3) can be naturally extended to edges through their endpoints.

4.4.1 Algorithm for (δ_0, δ_1)-Classification

Following the discussions above, given a given metric space (X, d) which is obtained through a rips Graph G for which $d = d_G^{\text{unw}}$, we may provide a mapping

$$X \to \mathbb{N} \times \mathbb{N} : x \mapsto (\delta_0(x), \delta_1(x)),$$

expressing the underlying local topology near x, as well as a lower bound on the number of cycles through x, furthermore indicating whether or not a cycle passes through x. We refer to this as (δ_0, δ_1)*-classification*.

For a graph G with n nodes and m edges, all pairwise unweighted shortest path distances can be obtained in $\mathcal{O}(nm)$ time [75]. As the number of connected components can be obtained in $\mathcal{O}(n + m)$ time during each of the $\mathcal{O}(n)$ iteration of the while loop, the computational complexity of (δ_0, δ_1)-classification is $\mathcal{O}(n(n + m))$. For computational efficiency, the proposed algorithm marks neighbors of a node with a particular local topology with the same local topology, which allows to reduce the computational complexity further to $\mathcal{O}(n^2)$ in practice [3]. Furthermore, during each iteration for disconnected graphs, the computational complexity for obtaining the unweighted distances may be further decreased, depending on the maximal number of nodes and edges in a component.

```
Data: Rips graph $G$ & (unweighted) distance parameter $r$
Result: $(\delta_0, \delta_1)$-classification of the nodes
1  queue = V($G$) initialize queue of the nodes to be classified
2  LG = matrix(length(V($G$)), 2) initialize matrix to store node topologies
3  DG = distances($G$, weights=NA) get all pairwise unweighted
                                   shortest path distances
4  while queue do
5      u = queue[1] specify current node u to $(\delta_0, \delta_1)$-classify
6      d0 = nocomponents(subgraph(G, which(DG[u,] == $r$))
           calculate the underlying degree $\delta_0(u)$ near u
7      d1 = d0 - nocomponents(subgraph(G, which($r <=$ DG[u,] $<$ Inf)))
           calculate lower bound $\delta_1(u)$ on underlying cycles near u
8      for v in union(u, neighbors(G. u)) do
9          map neighbors of u to the same class for efficiency
10         LG[v] = (d0, d1)
11         que.remove(v)
12     end
13 end
14 return LG
```

Algorithm 1: Pseudocode for (δ_0, δ_1)-classification. 'nocomponents' computes the number of connected components of a given graph. Note that in this algorithm, r plays the role of the smaller inner radius r' discussed above. We also start counting from 1, such as in R.

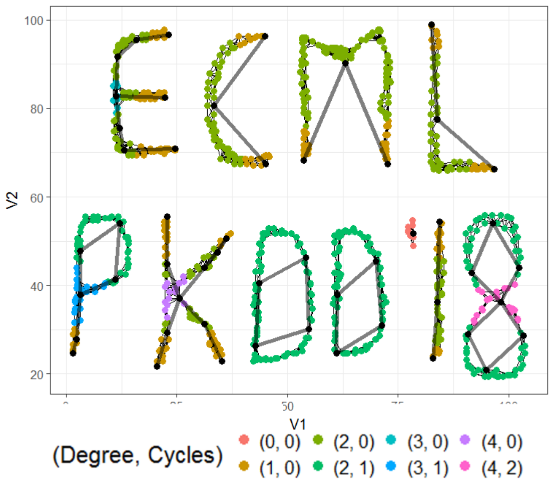


Figure 4.2: When the underlying graph-structured topology of a data set X is well-modeled by a Rips graph $\mathcal{R}_\epsilon(X)$, counting connected components in induced subgraphs suffices to learn topological structures locally, as well as the presence of cycles, through Algorithm 1 ($|D| = 873$, $\epsilon = 3.5$, $r = 3$, comp. time: 0.43s). By using these identified local topologies, we are able to reconstruct a graph homeomorphic to the underlying space through Algorithm 2 ($\tilde{r} = 4$, comp. time: 8.04s).

Tuning ϵ and r The distance parameters ϵ and r may usually be tuned by manual investigation. For all results in this chapter, it was sufficient to investigate the use of either $r = 2$ or $r = 3$. Tuning ϵ is more data dependent, and can e.g. be done through persistent homology (Section 2.5.2.2), by inferring a time during which the most prominent features (components and cycles) persist. One may also integrate over different parameter ranges, which are bounded by the maximal pairwise (original) distance for ϵ, and by the unweighted radius of the graph for r, which is defined similarly as in Definition 2.4.8. Consequently, one inspects how well the reconstructed graph (Section 4.5) approximates the original graph, checking for a balance between reducing the distance between them, the mean squared error, or metric distortion (e.g., one may redefine distances as their projected distances on the reconstruction), and the reduction of the graph size, as also discussed in [13].

4.5 Reconstructing the Graph from Local Topological Information

Figure 4.2 shows the result of running (δ_0, δ_1)-classification, i.e., Algorithm 1, on (a Rips graph constructed from) a point cloud data set X with an underlying graph-structured topology. The question is now how we can use this local information to reconstruct the entire underlying topology (the gray graph in Figure 4.2).

In [13] it has been shown that it essentially suffices to consider which points belong to an edge or not. In our setting, this corresponds to a binary '$(\delta_0 == 2)$'-classification. One can then cluster all points that are marked as an edge and those that are not separately. The obtained clusters then form the nodes of the reconstructed graph model that will be outputted. Nodes corresponding to clusters of non-edge points are then connected by an edge based on the criteria whether there exists a cluster of edge points between them [13, Algorithm 1]. A cluster of edge points connected to only one cluster of non-edge points marks a self-loop. Although through the same reasoning a cluster of edge points connected to no cluster of non-edge points marks an isolated cycle, this is not clearly mentioned in [13].

The problem with this approach is that different clusters may not be separated sufficiently, and branches may be too short relative to the amount of noise to even detect points on edges. This is illustrated in Figure 4.3. Given the '$(\delta_0 == 2)$'-classification that results from this example, [13, Algorithm 1] would provide a graph reconstruction consisting of only one node, which would not model the underlying bifurcating topology of the data.

Nevertheless, the exact (δ_0, δ_1)-classes provided for the data in Figure 4.3 completely characterize a bifurcating topology. Hence, the information retrieved by (δ_0, δ_1)-classification needs to be both *stored* and *used*. Applying complete-

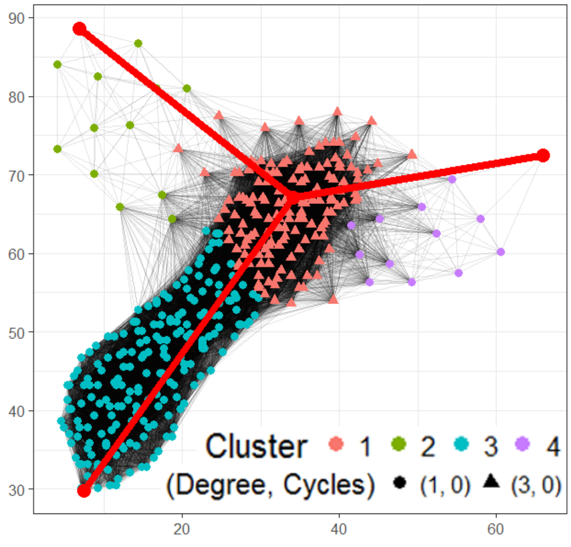


Figure 4.3: Classifying the local topologies ($\epsilon = 15$, $r = 3$, comp. time: $0.17s$), and using these to reconstruct the underlying graph topology (comp. time: $0.34s$) for a noisy sample of 395 points approaching a Y-structured topology with nonuniform density.

```
**Data:** Output LG of Algorithm 1 & unweighted distance parameter $\tilde{r}$
**Result:** A graph representing the underlying topology of $X = V(G)$
1  Cluster $\{x \in X : \delta_0(x) \geq 3\}$ by $(\delta_0, \delta_1)$-group in $G$
2  Let $N_1$ be the collection of obtained clusters
3  $\forall C \in N_1$, use $D$ to obtain a representative center $x_C \in X$
4  $\forall C \in N_1$, use $D$ to cluster $\{x \in D \backslash C : d_G^{\mathrm{unw}}(x_C, x) \leq \tilde{r}\}$ in $\delta_0(x_C)$
     components (e.g., through hierarchical clustering)
5  Let $N_2$ be the collection of obtained clusters
6  $\forall C \in N_2$, Use $d$ to obtain a representative center $x_C \in D$
7  If for $C, C' \in N_2$, $C \cap C' \neq \emptyset$, split $C \cup C'$ into two nonempty disjoint
     sets by ordering and thresholding the distances of the included points to
     $x_C$, according to $D$
8  Connect $C \in N_1$ and $C' \in N_2$ by an edge if $C'$ merged from $C$ in Step 5
     or 8
9  Cluster $D \backslash (\bigcup N_1 \cup \bigcup N_2)$ by $(\delta_0, \delta_1)$-group in $G$
10 Let $N_3$ be the collection of obtained clusters
11 Split each $C \in N_3$ consisting of (2,1)-classified points that is
     disconnected from $N_2$ in at least three consecutive connected
     components (this is an isolated cycle)
12 Connect $C \in N_2 \cup N_3$ and $C' \in N_3$ by an edge if they are connected in $G$
13 Connect $C, C' \in N_2$ by an edge if they are connected in $G$, unless this
     contradicts $\delta_0(x_C)$ or $\delta_0(x_{C'})$ in the current construction (this reduces
     sensitivity to the $\epsilon$ parameter for constructing the Rips graph)
14 **return** *A graph with (centers of)* $\bigcup_{i=1}^{3} N_i$ *as vertices and the obtained
     edges*
```

Algorithm 2: Pseudocode for reconstructing the graph topology based on (δ_0, δ_1)-classification

linkage hierarchical clustering [76] allows us to separate the points neighboring the cluster of (3,0)-classified points in the example in Figure 4.3 into three separated cluster. Inspired by this result, we use Algorithm 2 for reconstructing the underlying graph-structured topology from a (δ_0, δ_1)-classification. The pseudocode of Algorithm 2 assumes the graph G used in Algorithm 1 is given, as well as a pairwise distance matrix D on the point cloud data X that was used to construct G, e.g., the original (Euclidean) distances used to construct G. Note that the example in Figure 4.3 also shows that we can no longer directly use connected components in subgraphs of our Rips graph to efficiently separate disjoint branches. We solve this by introducing an additional unweighted radius parameter $\tilde{r}$, and force the non-multifurcation points within a distance $\tilde{r}$ of a multifurcation point to be clustered in a number of clusters that is consistent with the inferred degree. However, as we will discuss below, this additional parameter is only needed for connected components displaying non-isolated cycles.

The pseudocode of Algorithm 2 is written in a way that allows for many variants in its implementation. We use the original (Euclidean) metric used to construct our Rips graph for Algorithm 1, but one may as well choose the weighted distance matrix on G, which may lead to better results for computing centers of long and curvy patches, at the cost of computational efficiency. We define the center of a set $U \subseteq X$ as the data point $c_X := \arg\min_{x \in U}(\max_{y \in U} D(x,y))$, which leads to better and more centered point than the point closest to the mean in the case of nonuniform density. Representing the center in our current way works well for short patches of the underlying topology, but is less efficient for patches representing long and curving trajectories, as shown by the red graph in Figure 4.4. An alternative method is to use a breadth-first traversal to decompose long clusters representing edges into short and consecutive patches, resulting in the black graph in Figure 4.4. Note that both graphs are nevertheless homeomorphic. One may connect different centers by shortest paths as well, leading to a subgraph (backbone) model. Isolated circles are separated into four components by starting a breadth-first traversal at a random point, dividing points according to low, medium, or high distance from the root, and dividing the points at medium distance into two separate components. Finally, we replace the representative point of a (1,0) component—which represents a leaf in the underlying topology—such that it is furthest from its adjacent center.

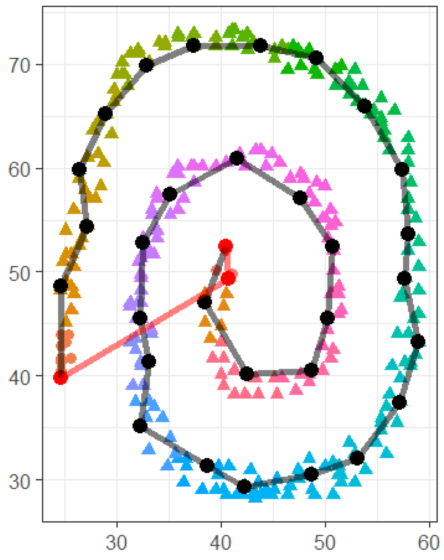


Figure 4.4: By a breadth-first traversal of the (2,0)-cluster, one may construct even better approximations of the underlying structure (black) than the original reconstructed graph (red).

Tuning $\tilde{r}$ The unweighted distance parameter $\tilde{r}$ may be either tuned manually (all results in this chapter were obtained by using either $\tilde{r} = r$ or $\tilde{r} = r + 1$, r being the distance parameter used to obtain the output of Algorithm 1), or tuned in an integration scheme as discussed in Section 5.2.1. However, a new distance parameter $\tilde{r}$ is not needed for components resembling isolated points, edges, cycles or multifurcating trees. This last observations follows from the fact that for a tree graph $T = (V, E)$, i.e., a connected graph without cycles, whenever $|E| \geq 1$ and there are no vertices of degree 2 (these are irrelevant for representing the topology of a geometric realization of the tree in terms of homeomorphisms), we have

$$\begin{cases} |E| = \frac{1}{2}\sum_{v\in V}\delta_0(v) = \frac{1}{2}|\{v \in V : \delta_0(v) = 1\}| + \frac{1}{2}\sum_{\substack{v\in V\\ \delta_0(v)\geq 3}}\delta_0(v), \\ |E| = |V| - 1 = |\{v \in V : \delta_0(v) = 1\}| + |\{v \in V : \delta_0(v) \geq 3\}| - 1. \end{cases}$$

This implies that the union of points having either (1,0) or (2,0) local topologies must be clustered into

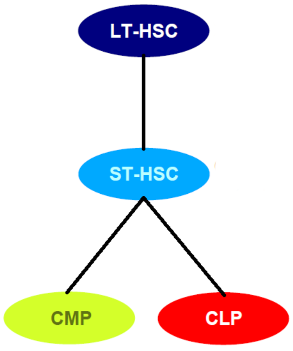
$$|E| = \sum_{\substack{v\in V\\ \delta_0(v)\geq 3}} \delta_0(v) - |\{v \in V : \delta_0(v) \geq 3\}| + 1 \tag{4.2}$$

components, where this number is computed with respect to the connected components with $\delta_0 \geq 3$. If the tree has at least one multifurcation point, all such obtained clusters of edges will be incident to at least one multifurcation point and represented by at least two nodes in the reconstructed graph topology. This allows for another variant of Algorithm 2 for tree-structured topologies: cluster the union of (1,0) and (2,0) classified points in the obtained number of clusters through (4.2), and connect each component with $\delta_0 \geq 3$ to all adjacent clusters of edges. This results in the reconstruction shown in Figure 4.3.

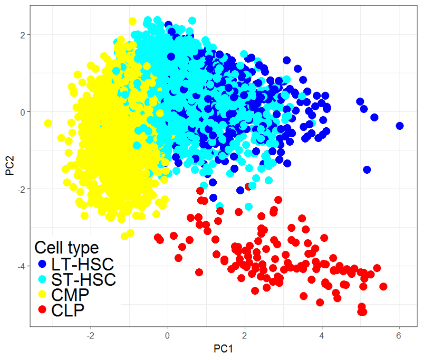
This shows the advantage of including δ_1-labels in our classification. Indeed, these do not contribute much to Algorithm 2, as isolated $(\delta_0 = 2)$-clusters imply $(\delta_0 = 2, \delta_1 = 1)$-clusters (in theory) and vice versa. However, if $\delta_1(x) = 0$ for each data point x in a particular component of our Rips graph, we directly conclude that there is no cycle in the underlying topology of that component (Theorem 4.4.1), and we do not need the additional distance parameter $\tilde{r}$ for this component.

4.5.1 An Example for Cell Trajectory Data

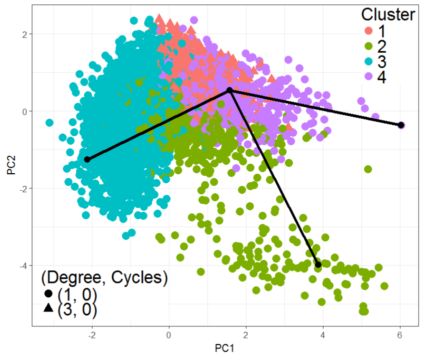
We considered a normalized expression data set X of 4647 manually analyzed bone marrow cells containing measurements of five surface markers (proteins): CD34, CD1632, CD117, CD127, and Sca1. These cells are known to differentiate from long-term hematopoietic stem cells (LT-HSC) into short-term hematopoietic stem cells (ST-HSC), which can in turn differentiate into either common myeloid progenitor cells (CMP) or common lymphoid progenitor cells (CLP) [77]. I.e., the



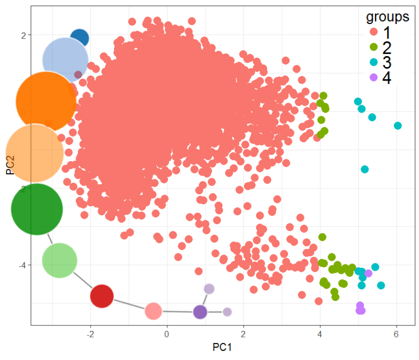
(a) The 4647 analyzed bone marrow cells consist of four cell types that are interconnected by means of cell differentiation.



(b) PCA plot of the expression data.



(c) LTDA of the expression data.



(d) Mapper graph and its induced assignments.

Figure 4.5: A normalized expression data set X visualized through a 2-dimensional PCA embedding, as well as its ground truth model and cell grouping, along with two different inferred models. Only our method captures the topology well, and induces a grouping that correlates well with the ground truth.

topology underlying this data set is that of a geometric realization in $\mathbb{R}^5$ of the graph depicted in Figure 4.5a. No data preprocessing was applied, and the Euclidean distance was used as the original metric. A PCA plot of the data is shown in Figure 4.5b. Comparing Figure 4.5a and 4.5b, we indeed note the presence of the Y-structured topology. However, it is clear that identifying this topology would be a crucial problem in absence of the cell labeling. Hence, our method may serve as a first step in the context of cell trajectory inference [77, 78], identifying the branching structure and different stages within a cell differentiation process. Our method infer (δ_0, δ_1)-classes in 15.55s ($\epsilon = r = 2$), and used these to reconstruct the underlying topology in 5.46s. Note that the present (1,0)- and (3,0)-classes imply an underlying tree-structured topology, and no additional distance parameter $\tilde{r}$ was needed for the reconstructing the graph. We inferred the exact same graph using both complete and McQuitty's linkage [79] as (hierarchical) clustering methods in Algorithm 2. However, the labeling induced by using the latter method, of which the result is shown in Figure 4.5c, correlated slightly better with the original cell types. The obtained branch-assignments correlate well with the original assignments, except for, most notably, non-CLP cells near the base of the ST-HSC$\rightarrow$CLP branch being assigned to the branch itself.

We compared our method to the original method [13] using two metrics. For the Euclidean distances, this took 1h17min, and using the weighted shortest path distances in the (same) graph $\mathcal{R}_2(X)$, this took 1h35min. Note that the main reason for the long computation times is that there is no (heuristic) speedup included in [13, Algorithm1] such as in Algorithm 1. Both methods were unable to capture the underlying topology, as both resulted in an isolated cycle in both cases. This is because more than 98% of the data points were marked as branch (non-edge) points through [13, Algorithm1], and the remaining edge points were unable to be separated, resulting in one selfloop. We also compared our method with Mapper (Section 2.6). Experimenting with different filter functions, only the projection onto the first principal component allowed us to correctly infer the underlying topology in 11.85s. However, this was a matter of luck, as the assignments induced by the resulting Mapper graph correlates badly with the original assignments, as shown in Figure 4.5d.

4.6 Discussion and Conclusion

Applying clustering techniques to study local topologies, and how these affect the global topology, introduces new possibilities for learning graph-structured topologies underlying point cloud data sets, as one may even detect cycles without the need of 1-dimensional homology (Theorem 4.4.1). We combined both LTDA and reconstruction methods in a simple and intuitive way, leading to a framework (Algorithm 2) for reconstructing the underlying graph based on local topological information. We furthermore showed how exact knowledge of the underlying degrees improves reconstruction methods.

These reconstruction methods are inspired by the fact that correct knowledge of local topological information guarantees the reconstruction of the correct global model [13]. This fact is translated into a data setting by means of clustering algorithms and (local) Rips graphs. Indeed, empirically we find other types of graphs (such as kNN graphs) to be less stable for inferring degrees, i.e., local topological information, through spherical neighborhoods, with minimum spanning trees being the most extreme examples (e.g., one will be able to deduce this from Figure 6.5 in Chapter 6). Since Rips graphs cannot effectively deal with nonuniform density (Section 2.4.1), one generally requires more manual input for tuning the

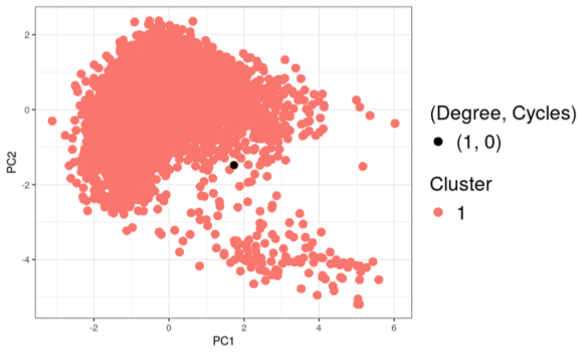
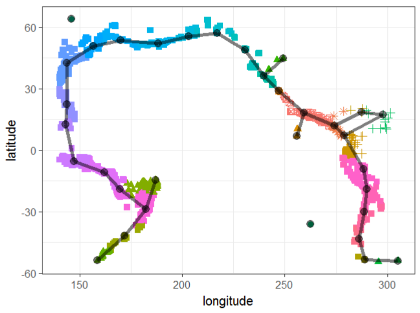


Figure 4.6: Local topological information and the model inferred thereof for the same cell trajectory data set X considered in Section 4.5.1, but where we increased the distance parameter of the constructed Rips graph from $\epsilon = 2$ to $\epsilon = 4$. The inferred local topological information is now consistent with that of a space/graph with an intrinsic dimension that is larger than 1 everywhere. Currently, our reconstruction method cannot effectively deal with such cases, and the whole data is represented by one single (black) point.

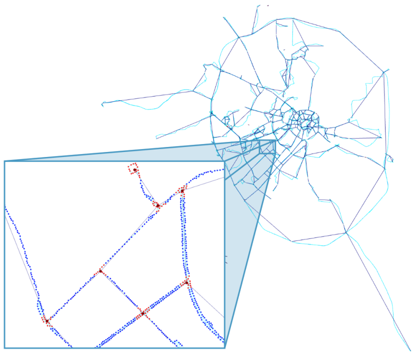
parameters of reconstruction method, especially when the data is accompanied by noise that makes it more difficult to separate branches. This can be illustrated through the cell trajectory data set X we considered in Section 4.5.1. As the data is more sparse along the LT-HSC$\rightarrow$ST-HSC and ST-HSC$\rightarrow$CLP branches (Figure 4.5b), we required a distance parameter $\epsilon = 2$—which is large relative to the scale of the data—for constructing a Rips graph that captures the connectedness of the model well. However, increasing ϵ further may result in distinct regions being directly connected without passing through the bifurcation location, i.e., the ST-HSC cells, first. Intuitively, the intrinsic dimension of the graph is now larger than 1 everywhere, instead of 1 almost everywhere,[2] and reconstruction methods cannot effectively deal with such cases, as shown in Figure 4.6.

To accommodate for the parameter sensitivity of reconstruction methods, they are ideally applied to more clean and/or processed data, that approximates the underlying model well. Examples of such data are shown in Figure 4.7. The advantage of these methods is however that they can directly model cycles, and mark and group important regions through the used clustering algorithms (unlike the method we will present in the following chapter). Furthermore, as we will discuss in more detail in Section 8.2, combining topological signatures obtained through persistent homology (see e.g. Chapter 6) with topological inference methods [80],

[2]Although this is intuitively clear, this can also be mathematically formalized. More specifically, for a *measure space* (X, Σ, μ), a property P is said to hold *almost everywhere* if there exists a set $N \in \Sigma$ with measure $\mu(N) = 0$, such that all $x \in X \backslash N$ have the property P. Letting X be a metric graph, N be the set of nodes of X (Definiton 2.4.7), and $P(x)$ state that the intrinsic dimension near x equals 1, one easily sees that P holds almost everywhere for sensible choices of Σ and μ.



(a) LTDA and underlying graph reconstruction through Algorithm 2 of earthquake locations restricted to a small rectangular are on the Earth. Image from [3].



(b) A graph reconstruction through [13, Algorithm 1] from GPS traces tagged 'Moscow' from Open-StreetMap (`http://www.openstreetmap.org/`). Image obtained with permission from [13].

Figure 4.7: Two 'clean' data sets for which graph reconstruction methods work well.

may further extend the applicability of graph reconstruction methods to more noisy data with a nonuniform density.

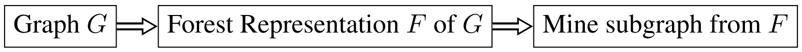
5
Inferring Topological Models through Forest Representations

This chapter is based on the following publications.

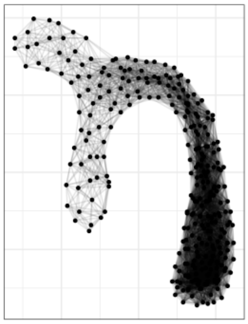
- *The Boundary Coefficient: a Vertex Measure for Visualizing and Finding Structure in Weighted Graphs.* In Proceedings of the 15th International Workshop on Mining and Learning with Graphs (MLG), 2019. [4]
- Robin Vandaele, Yvan Saeys, Tijl De Bie. *Mining Topological Structure in Graphs through Forest Representations*. Journal of Machine Learning Research, 21(215):1–68, 2020. [1]

5.1 Introduction

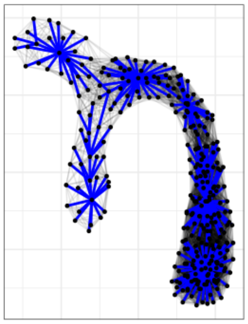
In the previous chapter, we discussed methods that reconstruct a non-subgraph model based on local topological information in metric data. In this chapter, we present a method for inferring subgraph models, i.e., *backbones*, that generalizes well to any graph in the sense of Definition 2.4.1, and is not restricted to (local) Rips graphs constructed from point cloud data. For this, we will introduce a vertex measure termed the *boundary coefficient*, also abbreviated as BC (Section 5.2), which—unlike the vertex degree—identifies well which nodes are located near the backbone in graphs that either satisfy the small-network model or not. This coefficient will be used to construct a forest representation, more specifically an f-pine, in Section 5.3. The final backbone will be inferred from the



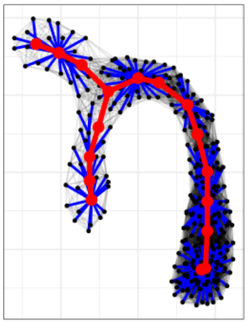
(a) High level overview of our introduced method for mining substructures in graphs.



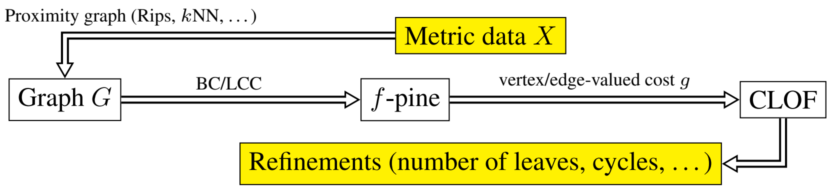
(b) The original graph G.



(c) A forest representation F of G.



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


Proximity graph (Rips, kNN, . . .)
Metric data X
Graph G
BC/LCC
f-pine
vertex/edge-valued cost g
CLOF
Refinements (number of leaves, cycles, . . .)

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

Figure 5.1: Overview of the method proposed in this paper.

f-pine through a graph optimization problem which we term **C**onstrained **L**eaves **O**ptimal sub**F**orest (CLOF, 5.4). The combined use of the BC and f-pines makes this method robust, while the use of forest representations makes solving CLOF computationally efficient. In Section 5.7, we will confirm the effectiveness of our method by inferring backbones in a variety of graph-structured data, such as social networks, earthquake locations scattered across the Earth, and high-dimensional cell trajectory data, the latter of which we will also discuss in more detail in Section 5.8.

The overview of the method we present in this chapter is shown in Figure 5.1.

5.2 The Boundary Coefficient

The first step of our method requires us to locate core nodes in our graph. Intuitively, these are the nodes that lie close to the backbone of our graph, i.e., its underlying simplified graph-structured topology (Figure 5.1). In Section 5.2.2 we present the *boundary coefficient* (BC), defined as the negative average *transmissiv-*

ity (Section 5.2.1) of a node. We discuss important properties of the BC, as well as its relationship to the ordinary *local cluster coefficient* (Section 5.2.3). In Section 5.2.4, we point out the differences between the BC and many existing measures that might be used to determine the core nodes of a graph, that lack important properties required for identifying such nodes in many practical weighted graphs. Finally, we present a way to efficiently compute the BC through (sparse) matrix multiplication in Section 5.2.5.

5.2.1 The Transmissivity of a Node

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

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

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

Definition 5.2.1. *Let $G = (V, E)$ be an undirected, positively weighted graph. Suppose that $u, v, w \in V, u \neq v \neq w$, belong to the same connected component of V. We define the (cosine of the)* angle $\widehat{uvw}$ *as*

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

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

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

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

Furthermore, it is important to note that the weights ω of a graph G do not have to satisfy the triangle inequality in G. Thus, we may have $\omega(\{u,v\})+\omega(\{v,w\}) < \omega(\{u,w\})$ for $\{u,v\}, \{v,w\}, \{u,w\} \in E$. The shortest path metric d will always naturally satisfy the triangle inequality, which is needed to generalize the (Euclidean) angle to graphs.

5.2.2 The Boundary Coefficient as the Average Transmissivity

The *boundary coefficient* (BC) of a node v is defined as its negative transmissivity averaged over the pairs of neighbors of v. As illustrated by Fig. 5.2 and Fig. 5.3, this is a measure for how close vertices are near the 'boundary' of the graph (hence the name), and by this, whether the nodes are close or far from the graph's core.

Definition 5.2.2. *Let $G = (V, E)$ be an undirected, positively weighted graph, without selfloops. For every $v \in V$ we define $\mathcal{N}(v) \subseteq V$ to be the set of neighbors of v in G. For every $v \in V$ with degree $\delta(v) = |\mathcal{N}(v)| > 0$, we define its* boundary coefficient *(BC) as*

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

The geometric interpretation of the BC applies to any graph, including those that satisfy the small-world network model. For this, observe that in this model, hubs will be transmissive for many other nodes in the graph. A concrete example of this for the Karate network will be provided in Section 5.7.

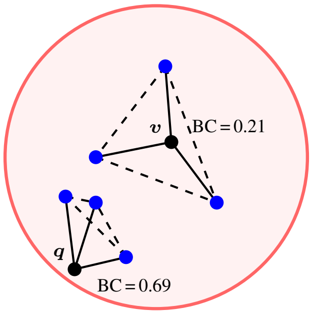
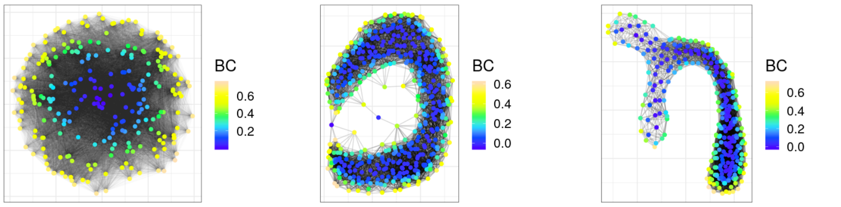


Figure 5.2: Geometric interpretation of the boundary coefficient: a point v lying further from the boundary has many more pairs of neighbors defining a large angle, than a point q lying close to the boundary. The dashed line represents the shortest path—not necessarily an edge—between two nodes. The boundary coefficients are computed using only the drawn connections and their Euclidean lengths.



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

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

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

Figure 5.3: The boundary coefficients for various Rips graphs $\mathcal{R}_{10}(X)$ built from 2D point cloud data sets X. Weights of the edges equal the Euclidean distance between their endpoints. The BC can handle fully weighted networks, curvature, as well and outliers, which are separated from the major core through boundary nodes.

5.2.3 Properties of the Boundary Coefficient

The BC is closely connected to the well-known *local cluster coefficient* (LCC) [70]. For every node $v \in V$ with degree $\delta(v) > 1$, it is defined as

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

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

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

$$-\mathcal{T}(u,v,w) = \cos \widehat{uvw} = \begin{cases} \frac{1}{2} & \text{if } \{u,w\} \in E \ , \\ -1 & \text{if } \{u,w\} \notin E \wedge u \neq w \ , \\ 1 & \text{if } u = w \ . \end{cases} \tag{5.1}$$

The intuition behind this is as follows. Suppose for $\{u, v\}, \{v, w\} \in E$, that (u, v, w) forms a closed triangle adjacent to v, i.e., $\{u, w\} \in E$. Since the graph is unweighted, each edge of this triangle gets assigned the same distance. Hence, the triple (u, v, w) is regarded as an equilateral triangle, which has all angles equal to 60°. This coincides that with the fact that $\mathcal{T}(u, v, w) = -\frac{1}{2} = -\cos 60°$. If $\{u, w\} \notin E$, we regard the triplet (u, v, w) as a straight line segment, defining a

$180°$ angle in v. Again, this coincides with the fact $\mathcal{T}(u,v,w) = 1 = -\cos 180°$. In this case, there may be other shortest paths from u to w in G, but $u \to v \to w$ is definitely one of them. If $u = w$, we regard the triple as two coinciding line segments defining a $0°$ angle in v. In this case $\mathcal{T}(u,v,w) = -1 = -\cos 0°$.

The explicit relationship between the BC and LCC is as follows.

Proposition 5.2.3. *Suppose $G = (V,E)$ is an unweighted graph, i.e., a graph in which every edge gets a weight equal to 1, without selfloops. Then for every $v \in V$ with $\delta(v) > 1$*

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

Proof of Proposition 5.2.3. Using the equalities given in (5.1), we have

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

Since

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

we find

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

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

Corollary 5.2.4. *Suppose $G = (V,E)$ is an unweighted graph without selfloops. Then for every $v \in V$, $\lim_{\delta(v)\to\infty} BC(v) = \frac{3}{2}LCC(v) - 1$.*

Proof. This is an immediate consequence of Proposition 5.2.3. □

Proposition 5.2.3 implies that the BC does not fulfill the *general versatility* requirement, i.e., it does not coincide with the ordinary LCC on unweighted graphs, as other generalizations of the LCC to weighted graphs do [81]. However, the BC does appear to be closely related to the LCC: it is nearly an affine transformation of the LCC (as given in Corollary 5.2.4). In Section 5.3.2, we show that such transformations result in the same forest representation through f-pines (Proposition 5.3.4).

The fact that the relationship between the BC and the LCC is not an exact affine transformation, is due to us allowing BC to be well-defined for nodes v with $\delta(v) = 1$, i.e., allowing $u = w$ in the summation over triples (u, v, w) adjacent to v. BC$(v) = \cos \widehat{wvw} = 1$ for these nodes, which coincides with our idea of nodes with a high boundary coefficient lying at the boundary of the graph. Any path that enters a node v with $\delta(v) = 1$ from a node w and wishes to continue, has no choice than to take a $180°$ turn back to w. Intuitively, the path has reached a 'dead end' in v, and hence, reached the boundary of the graph. Furthermore, if F is a spanning forest of G (Definition 5.3.1), then a *leaf* of G, i.e., a node $v \in V$ for which $\delta(v) = 1$, will always be a leaf of F as well. In Section 5.3.3, we will use the BC to obtain a particular spanning forest, in which leaves are exactly meant to represent boundary nodes of G, i.e., for which the coefficient is high. Hence, for our method, it makes sense that the BC is both well-defined at leaves, and obtains its maximal value there.

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

Though the BC does not coincide with the ordinary LCC on unweighted graphs, it does satisfy four other essential properties of generalizations of the LCC to weighted graphs, as discussed in [81]. One of these, i.e, its *applicability to fully weighted networks*, has been illustrated in Figure 5.3. We consider this property of the boundary coefficient, as well as its *weight-scale invariance*, *continuity*, and *robustness to noise* [81], far more important for our purpose of identifying core structure in a wide variety of weighted graphs, than coinciding with the LCC on unweighted graphs. We state and prove these properties formally below.

Proposition 5.2.5. *(Weight-scale invariance, [4]). Let $G = (V, E)$ be an undirected graph without selfloops, with weighting function $\omega : E \to \mathbb{R}^+$. Let $\omega_\lambda : E \to \mathbb{R} : \{u, v\} \mapsto \lambda\omega(\{u, v\})$ for a global scale factor $\lambda > 0$. Then for every $v \in V$, $BC_\lambda(v) = BC(v)$, where $BC_\lambda(v)$ equals the boundary coefficient of v for the new weighting function ω_λ.*

Proof. By multiplying each edge weight with a global scale factor $\lambda > 0$, the shortest path distance between any two nodes is also scaled by the same factor λ. Hence, the stated equality easily follows from Definitions 5.2.1 & 5.2.2. $\square$

Lemma 5.2.6. *Let $G = (V, E)$ be an undirected graph, with weighting function $\omega : E \to \mathbb{R}^+$. Suppose that $\mathbb{E}$ denotes an additive noise matrix, which defines a new weighting function $\omega_\epsilon : E \to \mathbb{R}^+ : \{u, v\} \mapsto \omega(\{u, v\}) + \mathbb{E}_{u,v}$. Then the following statements are valid:*

1. $\lim_{\|\mathbb{E}\|_\infty \to 0} \|d - d_\epsilon\|_\infty = 0$, *where d denote the shortest path metric on V according to ω, and d_ϵ according ω_ϵ, where we follow the convention that $d(u,v) - d_\varepsilon(u,v) = 0$ if u and v lie in different connected components of G;*

2. *for any $u, v, w \in V$ belonging to the same connected component of G, with $u \neq v \neq w$, $\lim_{\epsilon \to 0} \mathcal{T}_\epsilon(u,v,w) = \mathcal{T}(u,v,w)$, where $\mathcal{T}(u,v,w)$ denotes transmissivity of v for u and w according to ω, and $\mathcal{T}_\varepsilon(u,v,w)$ according to ω_ϵ.*

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

2. This easily follows from Proposition 5.2.6.1. and Definition 5.2.1. □

Proposition 5.2.7. *(Continuity). Let $G = (V, E)$ be an undirected graph without selfloops, with weighting function $\omega : E \to \mathbb{R}^+$. Suppose ω_ϵ is a new weighting function on E that differs in exactly one edge $e \in E$ by an additive constant $\epsilon \in \mathbb{R}$, i.e., $\omega_\epsilon(e) = \omega(e) + \epsilon \in \mathbb{R}^+$, and $\omega_\varepsilon|_{E \setminus \{e\}} \equiv \omega|_{E \setminus \{e\}}$. If BC_ϵ denotes the boundary coefficient function according to the new weighting function ω_ϵ, then $\lim_{\epsilon \to 0} BC_\epsilon(v) = BC(v)$ for all $v \in V$ with $\delta(v) > 0$.*

Proof. This easily follows from Lemma 5.2.6 and Definition 5.2.2. □

The problem in the ordinary formulation of the 'robustness to noise' property stated by [81], is that the error-value $\Delta(\mathcal{E})$ is not well-defined when a node has a boundary coefficient of 0. Hence, we consider a slight variant below.

Proposition 5.2.8. *(Robustness to noise, [4]). Let $G = (V, E)$ be an undirected graph, with weighting function $\omega : E \to \mathbb{R}^+$. Suppose that $\mathbb{E}$ denotes an additive noise matrix, which defines a new weighting function $\omega_\epsilon : E \to \mathbb{R}^+ : \{u, v\} \mapsto \omega(\{u,v\}) + \mathbb{E}_{u,v}$. If $f : \mathbb{R} \to \mathbb{R}$ is any continuous function such that $f \circ BC(v) \neq 0$ for any $v \in V$, then*

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

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

Proof. This follows from Lemma 5.2.6 and f being continuous. □

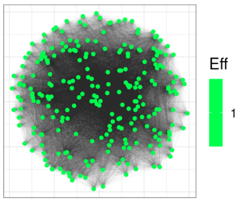
Remark 5.2.9. *The fact that we consider our variant to robustness of noise an equally important property for our method for topological data analysis of graph-structured data, is due to Proposition 5.3.4 stating that BC-pines are invariant under affine transformations of the BC with a positive scaling factor. The BC may always be mapped to an interval excluding 0 using such continuous transformation.*

Note that the theoretical rate of convergence of $\Delta(\mathbb{E})$ in Theorem 5.2.8 depends on $|E|$ (see the proof of Lemma 5.2.6.1.), a consequence of allowing arbitrary long paths (in terms of number of edges) between the endpoints of a triple adjacent to a node, to compute the BC.

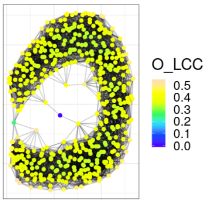
5.2.4 Comparing the Boundary Coefficient to other Measures

Certain other vertex measures that might be used to identify core nodes, including the ordinary LCC, already existed before we introduced the BC [4]. It turns out that the LCC is particularly useful to our method for inferring backbones in a variety of unweighted graphs (Section 5.7).

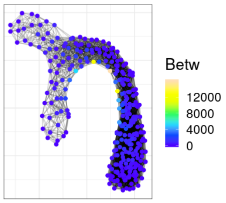
However, we found its generalizations to weighted graphs—an extensive summary of these is given by [81])—as well as other existing measures trying to quantify the coreness of a node, such as graph centrality measures [82–84], to be insuf-



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



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



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

Figure 5.4: Various possible existing 'core' measures for the Rips graphs in Figure 5.3. None of them capture the true core nodes of the graph well.

ficient for many of our practical examples (Figure 5.4). These were not designed for the purpose of identifying or visualizing a global core structure within a wide variety of graph-structured data sets. As such, they lack important properties of the boundary coefficient, such as scalability, applicability to fully weighted networks (compare Figure 5.4a to 5.3a), robustness to outliers (compare Figure 5.4b to 5.3b), and the ability to deal with nonlinear substructures (compare Figure 5.4c to 5.3c).

In summary, the crucial differences between the BC, the LCC and many of its generalizations [81], and standard global centrality measures such as eccentricity [83] or betweenness [84], as well as the main reasons why the BC outperforms these measures for a wide variety of applications, are that for a node $v \in V$:

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

5.2.5 Computation of the Boundary Coefficient

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

Definition 5.2.10. *Let $G = (V, E)$ be an undirected, positively weighted graph, and D the matrix of pairwise shortest path distances between the nodes of G. Let $|E(P)|$ denote the unweighted length of a path P. For $k \in \mathbb{N}$, we define the* hop-k-approximation *of D as the matrix $\mathcal{H}_k(D) = (\mathcal{H}_k(D)_{u,v})_{u,v\in V}$, where*

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

It is easy to see that increasing k leads to a better approximation of D (hence the name 'hop-k-approximation'). This is formalized through the following proposition.

Proposition 5.2.11. *Let $G = (V, E)$ be a connected, undirected, positively weighted graph, and D the matrix of pairwise shortest path distances between the nodes of G. Then the following statements are valid:*

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

where diam$_{unw}$(G) denotes the unweighted diameter of G.

Proof. 1. If u can be reached from v in 0 steps, then $u = v$, which implies that $D_{u,v} = 0$.

2. For any nodes $u, v \in V$, v can be reached from u within $\text{diam}_{\text{unw}}(G)$ steps.

3. This is clear from the definition of $\mathcal{H}_k(D)$. □

For a disconnected graph G, $\mathcal{H}_k(D)$ will never be equal to D for any $k \in \mathbb{N}$. $D_{u,v}$ is either undefined or defined to be $+\infty$ if u and v lie in different connected components of G. However, $\mathcal{H}_k(D)_{u,v}$ is always well-defined, and will be equal to 0 if there is no path between u and v. This will show to be convenient for computing the BC (Theorem 5.2.13).

Notation 5.2.12. *For any $z \in \mathbb{Z}$, we define the mapping*

$$\cdot^{\odot^z} : \bigcup_{n,m\in\mathbb{N}} \mathbb{R}^{n\times m} \to \bigcup_{n,m\in\mathbb{N}} \mathbb{R}^{n\times m} : A \mapsto A^{\odot^z} ,$$

with

$$A^{\odot^z}_{u,v} := \begin{cases} A^z_{u,v} & \text{if } z \geq 0 \vee A_{u,v} \neq 0 , \\ 0 & \text{otherwise} . \end{cases}$$

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

Theorem 5.2.13. *Let $G = (V = \{v_1, \ldots, v_n\}, E)$ be an undirected, positively weighted graph, without selfloops. Let D denote the matrix of pairwise shortest* *path distances between the nodes of G. If $\delta(v) > 0$ for all $v \in V$, then*

$$\begin{pmatrix} BC(v_1) \\ \vdots \\ BC(v_n) \end{pmatrix} = \begin{pmatrix} \frac{1}{\delta(v_1)^2} \\ \vdots \\ \frac{1}{\delta(v_n)^2} \end{pmatrix} \odot \Bigg[\Big(\sum_{u \in V} \mathcal{H}_1(D)_u \Big) \odot \Big(\sum_{u \in V} \mathcal{H}_1^{\odot^{-1}}(D)_u \Big) - \frac{1}{2} diag\left(\mathcal{H}_1^{\odot^{-1}}(D) \mathcal{H}_2^{\odot^{2}}(D) \mathcal{H}_1^{\odot^{-1}}(D) \right) \Bigg], \tag{5.2}$$

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

Proof. For $v \in V$, with $\delta(v) > 0$, we have

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

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

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

which concludes the proof. □

Note that both the left hand side and right hand side in (5.2) are undefined for nodes $v \in V$ with $\delta(v) = 0$. We regard such nodes simultaneously as boundary nodes, as well as core nodes within their own component.

```
Input: weighted graph $G$
Output: vector of boundary coefficients BC
1 hop1 = spam(hop_k_approx($G$, k=1)) #compute hop-1-approximation
2 hop2 = spam(hop_k_approx($G$, k=2)) #compute hop-2-approximation
3 BC = (apply.spam(hop1, 1, sum) * apply.spam((1 / hop1), 1, sum)
            - diag((1 / hop1) %*% hop2^2 %*% (1 / hop1)) / 2)
        / (degree($G$)^2) #compute boundary coefficients
4 return(BC)
```

Algorithm 3: Computing the boundary coefficients through sparse matrices. 'spam' converts matrices to sparse matrix format. 'apply.spam(A, 1, f)' applies the function f to each row of the sparse matrix A. The operations '*' and '^2' denote element-wise multiplication, whereas '%*%' denotes matrix multiplication. '1 / A' returns the element-wise inverse of a sparse matrix A. 'diag(A)' returns the diagonal of matrix A.

It follows that the BC may be computed using pairwise Dijkstra's algorithm with early termination, and (sparse) matrix multiplications. Pseudocode for an algorithm computing the BC is provided in Algorithm 3.

Theorem 5.2.14. *Let G be an undirected, positively weighted graph, with n nodes and m edges. The boundary coefficients of all nodes in V can be determined in $\mathcal{O}(n(m + n^{1.374}))$ time using Algorithm 3.*

Proof. Obtaining the hop-k-approximations can be done $\mathcal{O}(n(m+n\log n))$ time through Dijkstra's algorithm. The matrix operations can be done in $\mathcal{O}(n^{2.374})$ time, which is the computational cost of matrix multiplication [85]. □

Remark 5.2.15. *The computational cost stated in Theorem 5.2.14 is worst-case, since it does not account for the potential sparsity of $\mathcal{H}_k(D)$, $k \in \{1, 2\}$.*

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

5.3 Forest Representations of Graphs through f-Pines

In this section, we introduce the intermediate step that represents a given graph G by means of a (spanning) forest of G, as shown in Figure 5.1. This step will use the concept of the *f-pine* of a graph [4], which we present in Section 5.3.1. We will discuss a variety of its properties in Section 5.3.2. Finally, in Section 5.3.3 we illustrate how the boundary coefficient can be used to deduce a forest representation, from which we may efficiently infer backbones.

5.3.1 The f-Pine of a Graph

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

Definition 5.3.1. *Let $G = (V, E)$ be a graph. A spanning forest F of G is a subgraph of G that contains all vertices of G, such that each connected component of G is also a connected component of F, and F contains no cycles.*

Definition 5.3.2. *Let $G = (V, E)$ be a graph, and $f : V \to \mathbb{R}$. A spanning forest F of G is called an f-*pine[1] *in G, if*

$$F \in \arg\min \left\{ \sum_{v \in V} \delta_{F'}(v) f(v) : F' \textit{is a spanning forest of } G \right\}, \tag{5.3}$$

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

The intuition behind the naming is that an f-pine corresponds to trees having many 'needles' that 'stick out and point' towards (locally) high values of f (Figure 5.1c). Note that the number of edges in F is fixed through the number of nodes and components of G.

5.3.2 Properties and Computation of the f-Pine

Looking at Definition 5.3.2, an f-pine of a graph G is a spanning forest of G that prefers high-degree nodes where f attains a low value. More specifically, an f-pine attaches every node u to a node v where f reaches a local minimum.

Proposition 5.3.3. *Let $G = (V, E)$ be a graph, $f : V \to \mathbb{R}$, and F an f-pine in G. For every $u \in V$ with $\delta_G(u) > 0$, there exists $v \in \arg\min\{f(w) : w \in \mathcal{N}_G(u)\}$ such that $\{u, v\} \in E(F)$.*

Proof. Assume $u \in V$ with $\delta_G(u) > 0$. If $\{u, v\} \notin E(T)$ for every $v \in \arg\min\{f(w) : w \in \mathcal{N}_G(u)\}$, then choose such v. Let $P = (u = x_0, x_1, \dots, x_k = v)$ be the unique path from u to v in T. Since $\{u, x_1\} \in E(T)$, $f(x_1) > f(v)$, and we can replace $\{u, x_1\}$ by $\{u, v\}$ to obtain a tree attaining a lower cost as expressed by (5.3). $\square$

The intuition behind the proposition above is that the building blocks of an f-pine are several large star graphs that result from pulling every node towards a node where f attains a local minimum. Furthermore, Definition 5.3.2 implies that

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

the centers of these star graphs will be connected through nodes where f attains a low value on average as well.

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

Proposition 5.3.4. *Let $G = (V, E)$ be a graph, $f : V \to \mathbb{R}$, and F an f-pine in G. If $g = af + b$ for some $a \in \mathbb{R}^+, b \in \mathbb{R}$, F is also a g-pine in G.*

Proof. We have

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

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

- the sum of degrees over all vertices of a graph is twice its number of edges;
- the number of edges in any forest graph containing n vertices, is n minus its number of connected components;
- by Definition 5.3.1, the number of connected components of a graph equals the number of connected components of a spanning forest of that graph.

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

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

Proposition 5.3.5. *Let $G = (V, E)$ be a graph, and $f : V \to \mathbb{R}$. Finding an f-pine in G is equivalent to finding a minimum spanning tree for each connected component in G, where each edge $\{u, v\}$ is assigned to have weight $f(u) + f(v)$.*

Proof. It holds that

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

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

Theorem 5.3.6. *Let G be a graph with n nodes and m edges, and $f : V \to \mathbb{R}$. Then an f-pine of G can be determined in $\mathcal{O}(\alpha(m,n)m)$ time using Algorithm 4. Here α is the classical functional inverse of the Ackermann function, which for all practical purposes may be considered a constant no greater than 4 [49].*

Proof. The stated complexity is that of computing the regular minimum spanning forest [86]. Reweighing the edges can be done in $\mathcal{O}(m)$ time. □

```
**Input:** graph $G$, vertex-valued cost function $f$ on $G$
**Output:** an $f$-pine of $G$
1 new_weights = $f[E(G)[,1]]$ + $f[E(G)[,2]]$
2 pine = mst($G$, weights=new_weights)
3 return(pine)
```

Algorithm 4: Computing the f-pine. 'mst' is a function that computes the minimum spanning of a graph. The used weights follow from the result in Proposition 5.3.5.

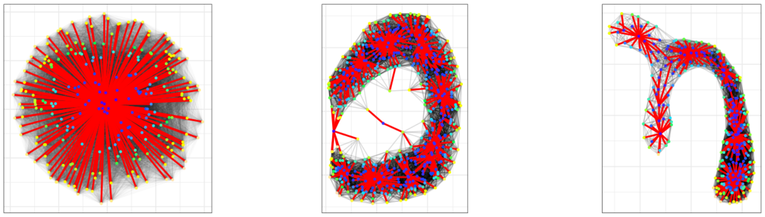
We are now prepared to show how the BC (Section 5.2) and f-pines work together to provide effective forest representations for topological data analysis of graphs.

5.3.3 The BC-pine of a Graph

Proposition 5.3.3 states that an f-pine connects nodes to its local minima. Hence, if f is a 'core' measure identifying nodes close to the underlying core structure of the graph, then an f-pine is the result of interconnecting star graphs through the core of the given graph. This is the exact purpose for which we designed the boundary coefficient, taking on low values near the core structure of a graph, and high values near the boundary nodes of the graph. Three example BC-pines are illustrated in Figure 5.5.

An important property of the BC-pine is displayed on Figure 5.5b. As the BC is able to separate outliers from the main core structure through boundary nodes where the BC attains locally higher values by design, the BC-pine avoids passing through outliers to reach one (true) core region from another. This would increase the cost of the pine according to (5.3), as it would include too many boundary nodes on either side of the outlier node.

[4] showed that by iteratively 'pruning' the BC-pine, i.e., discarding its leaves, we retract our pine towards the topological core underlying the graph. However, if the original graph has a very complex or even no ground-truth underlying topology, revealing a low complexity topology by means of this 'top-down' method may be difficult, discarding (too) many nodes. Furthermore, even in graphs with a simple underlying topology, the presence of outliers may require us to prune



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

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

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

Figure 5.5: The BC-pines (with edges in red) for the three example graphs shown in Figure 5.3. Note that by Proposition 5.3.3, a BC-pine will result in a star graph that connects all nodes to a global minimum in a complete graph, such as for the graph on the left.

many times to remove these connections (Figure 5.5b), resulting in our backbone retracting from the true underlying leaves as well. Hence, we are clearly in need of an approach that allows us to identify interesting substructures in forest graphs, spreading out across the entire graph, while the resulting complexity remains easy to tune and control. This leads us to the following section, introducing a 'bottom-up' method for identifying our backbone.

5.4 Constrained Leaves Optimal subForest (CLOF)

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

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

Definition 5.4.1. *(CLOF). Let $G = (V, E)$ be a graph, and suppose f is a real-valued function, associating a positive cost to either each vertex or each edge of G (i.e., the domain of f, denoted $\mathrm{Dom}(f)$, is either equal to V or to E). For a subgraph $H = (V(H), E(H))$ of G, we define its cost $f(H) := \sum_{\alpha \in \mathrm{Dom}(f) \cap V(H) \cap E(H)} f(\alpha)$. The* **C**onstrained **L**eaves **O**ptimal sub**F**orest *problem (CLOF) is stated as follows.*

$$\text{Given } k \in \mathbb{N}_{\geq 2}, \text{ find a subforest } F \text{ in } G \text{ with at most } k \text{ leaves, maximizing } f(F)\,. \tag{5.4}$$

Proposition 5.4.2. *CLOF is NP-hard.*

Proof. For an edge-valued cost function and $k = 2$ leaves, the stated problem is equivalent to the NP-hard longest path problem [87]. $\square$

Though CLOF is NP-hard in general, we are actually not interested in its solution for arbitrary graphs. Such solutions may just not be topological meaningful. An example of this would be the longest path in the graph shown in Figure 3.3, which would 'wiggle' through the entire graph without reflecting its 'straight' linear topology. This is one of the main reasons we choose for a forest representation as an intermediate step. In this way, we can design our forest such that these solutions are indeed meaningful as well as robust. Furthermore, as we will show in this section, CLOF is efficiently solvable for forest graphs in practice.

In Section 5.4.1 we will derive an efficient solution to CLOF for tree graphs, which will lead us to an efficient solution for forest graphs in Section 5.4.2.

5.4.1 Solving CLOF in Tree Graphs

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

Theorem 5.4.3. *Let $T = (V, E)$ be a tree graph and f a real-valued function associating a positive cost to either each vertex or each edge of T. Then (5.4) is equivalent to a monotone submodular set function maximization problem subject to a cardinality constraint [88].*

Proof of Theorem 5.4.3. First observe that for any set of leaves $L = \{l_1, \dots, l_k\}$ of T, there is a unique subtree T_L of T that contains exactly the same set as leaves. Hence, if $V \supseteq \mathcal{L}$ is the set of all leaves of T, we may define $\tilde{f} : 2^{\mathcal{L}} \to \mathbb{R}^+ : L \mapsto f(T_L)$, where $f(T_L)$ is the cost of the subtree T_L as defined in Definition 5.4.1. Suppose now that $L \subseteq L' \subsetneq \mathcal{L}$, and take any $l \in \mathcal{L} \backslash \{L'\}$. Observe that T_L is a subtree of $T_{L'}$, for which the unique path from l to T_L includes the unique path from l to T'_L. Hence, $\tilde{f}(L \cup \{l\}) - \tilde{f}(L) \geq \tilde{f}(L' \cup \{l\}) - \tilde{f}(L')$, i.e., $\tilde{f}$ is a submodular set function on $\mathcal{L}$. Furthermore $\tilde{f}$ is clearly monotone, as $L \subseteq L' \subseteq \mathcal{L} \implies \tilde{f}(L) \leq \tilde{f}(L')$. Finally, by definition of $\tilde{f}$, (5.4) is equivalent to maximizing $\tilde{f}$ subject to the cardinality constrained given by k. $\square$

The general problem of maximizing a monotone submodular function subject to a cardinality constraint is NP-hard, but admits a $1 - 1/e$ approximation algorithm [88]. However, the interestingness of Theorem 5.4.3 lies in the fact that (5.4) is equivalent to a nontrivial monotone submodular set function maximization problem, for which we are able to actually provide an exact solution in polynomial time.

Theorem 5.4.4. *(A greedy solution for CLOF). Let $T = (V, E)$ be a tree graph and f a real-valued function associating a positive cost to either each vertex or* *each edge of T. Given $k \in \mathbb{N}_{\geq 2}$, the following algorithm finds a subtree T' in T that maximizes $f(T')$ over all subtrees T' in T with at most k leaves.*

1. *Let T' be the longest path (according to f) between two leaves in T.*

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

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

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

   The endpoint of P different from l_i is a multifurcation point m of T^k_{opt}. If $m \in V(T^{k-1}_{gr})$, then since $f(P) > f(P_k)$, the algorithm would have chosen to add P instead of P_k to obtain T^k_{gr}, a contradiction, so that $m \notin V(T^{k-1}_{gr})$. Let Q be the unique path from m to T^{k-1}_{gr}. If P is edge-disjoint from Q, then $P + Q$ would have been chosen instead of P_k by the greedy algorithm, so that P and Q partially overlap. Now let l_j be any leaf in T^k_{opt} different from l_i. The path R from m to l_j in T^k_{opt} is now both edge- and vertex-disjoint from T^{k-1}_{gr} (see Figure 5.6a for a sketch of this case). Furthermore $f(R + Q) > f(R) > f(P_k)$, and the greedy algorithm would have chosen to add the path $R + Q$ instead of P_k, a contradiction.

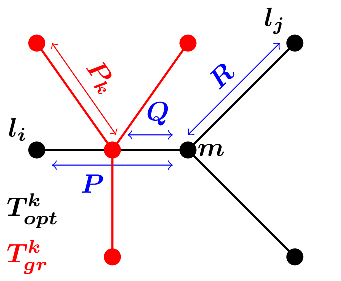
2. *For every leaf l_i of T^k_{opt}, the path P that connects l_i to the tree induced by the leaves $\{l_1, \ldots, l_{i-1}, l_{i+1}, \ldots, l_k\}$ contains edges from T^{k-1}_{gr}.*

   Consider an arbitrary leaf l_i of T^k_{opt}, and let v_i be the point closest to l_i on the first edge e_i on the path from l_i to the tree induced by the set of leaves $\{l_1, \ldots, l_{i-1}, l_{i+1}, \ldots, l_k\}$, that is also contained in $E(T^{k-1}_{gr})$. Note that possibly $l_i = v_i$. Now let l'_i be any leaf of T_{gr}^{k-1} that is reachable from v_i after removing e_i from in T_{gr}^{k-1}. If $l_i \neq l_j$ are both leaves of T_{opt}^k, then $l'_i \neq l'_j$. To see this, observe that for $l_i \neq l_j$, by definition of v_i, we have $v_i \neq v_j$, and that the path from v_i to v_j in T_{opt}^k must go through both e_i and e_j. As this path is unique in T, it is also fully contained in T_{gr}^{k-1}. Hence, the path $v_i \to v_j \to l'_j$ is the unique path from v_i to l'_j in T_{gr}^{k-1}, and passes through e_i. Hence, l'_j is not reachable from v_i after removing e_i from T_{gr}^{k-1}. As such, we obtain an injection $l_i \mapsto l'_i$ of the k leaves in T_{opt}^k to the $k-1$ leaves in T_{gr}^{k-1}, a contradiction. Note that this case is simply not possible, independent of the used cost function. We provided a sketch for the closest possible case in Figure 5.6b.

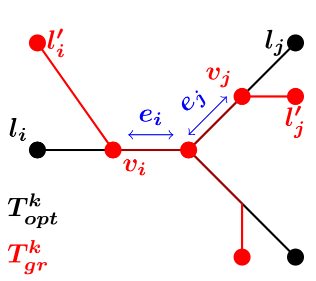
Since both cases lead to a contradiction, we conclude that T_{opt}^k cannot achieve a cost strictly higher than T_{gr}^k. This implies that T_{gr}^k is an exact solution to (5.4).

For f a vertex-valued function, the proof goes analogous to the proof of Theorem 5.4.4. However, the increase in cost of adding a new path P to the current tree is now the sum of the cost over all vertices in P, minus the cost of the connecting node. The only resulting change we need to apply in the proof of Theorem 5.4.4, is that instead of writing '$f(R+Q) > f(R) > f(P_k)$' in the first considered case, we now write '$f(R+Q) > f(P_k)$', as $f(R)$ may not be well-defined according to this convention. $\square$

Remark 5.4.5. *Due to a greedy algorithm resulting in the optimal subtree according to (5.4) for tree graphs, we only need to conduct the algorithm once to get all solutions up to the given value $k \in \mathbb{N}_{\geq 2}$. By storing the included vertices or edges for each iteration, we can quickly obtain the corresponding subgraph and analyze the results for different number of leaves (see also Algorithm 5 for the pseudocode*



(a) Sketch for the first case in the proof of Theorem 5.4.4. The cost of $Q+R$ must be bounded from above by the cost of P_k due to the definition of the greedy algorithm.



(b) Sketch for the second case in the proof of Theorem 5.4.4. There is a systematically defined injection from the leaves of T_{opt}^k to the leaves of T_{gr}^k.

Figure 5.6: Sketches for the different cases in the proof of Theorem 5.4.4.

of the corresponding algorithm). Storing the cost up to a certain number of leaves turns out to be useful in practice as well. It may serve as a tool for tuning the number of leaves, as we will discuss in Section 5.5.

Theorem 5.4.6. *Let $T = (V, E)$ be a tree graph with n nodes (and $n-1$ edges), and f a real-valued function associating a positive cost to either each vertex or each edge of T. Then for given $k \in \mathbb{N}_{\geq 2}$, an exact solution to (5.4) can be computed in $\mathcal{O}(n \min(k, l) l)$ time using Algorithm 5, where $l = |\{v \in V : \delta(v) = 1\}|$, i.e., l denotes the number of leaves in T.*

Proof. The advantage of working with tree (or forest) graphs, is that the path between a pair of nodes is always unique. The pairwise distance matrix D_f between all leaves and all nodes in T, according to f, can be obtained in $\mathcal{O}(n \cdot l)$ time by using a bread-first-search for every leaf. For each of the no more than $\min(k, l)$ iterations of the algorithm described in Theorem 5.4.4, we can add the new path in $\mathcal{O}(n \cdot l)$ time. This is clear for the first iteration: search for the maximal entry of D_f and add the corresponding path to the current (empty) structure. After the first path has been added, each consecutive leaf to be added can also be determined in $\mathcal{O}(n \cdot l)$ time, after which the path can be added in $\mathcal{O}(n)$ time. □

The following result shows to be very useful for practical applications, as we will discuss in Section 5.5. First, we need another definition.

Definition 5.4.7. *Let $G = (V, E)$ be a tree graph, and suppose f is a real-valued function, associating a cost to either each vertex or each edge of G. We say that*

```
  Input: tree $T$, positive cost function $f$, upper bound $k \geq 2$ on leaves
         (standard $\infty$)
  Output: A list L such that the subgraph of $T$ induced by the nodes in
          L[1:j] equals the solution to CLOF for $j \leq k$.
1 L = list(NULL, longest_path($T$, cost=$f$))
          #the solution for $k \in \{0, 1\}$ is the empty graph by convention
          #the solution for $k = 2$ is the longest path according to $f$
2 current_leaves = 2 #number of leaves in the current solution
3 while current_leaves $<$ k & L != $T$ do
4     v = which.max(distances($T$, leaves($T$), L), cost=$f$)[1]
              #determine furthest leaf from current solution according to $f$
5     current_leaves += 1
6     L[[current_leaves]] = path(from=L, to=v) #update the current solution
7 end
8 return(L)
```
Algorithm 5: **C**onstrained **L**eaves **O**ptimal sub**F**orest (CLOF) in trees. The function 'distances(G, U, V)' returns the distances between the nodes in U and V of a graph G.

f is constant on leaves *of G, if either f is constant on $\{\{u,v\} \in E : \delta(u) = 1 \vee \delta(v) = 1\}$ if f is edge-valued, or f is constant on $\{v \in V : \delta(v) = 1\}$ if f is vertex-valued.*

Theorem 5.4.8. *Let $T = (V, E)$ be a tree graph with $|E| > 1$, and suppose f is a real-valued function, associating a positive cost to either each vertex or each edge of T. If f is constant on leaves of T, then a solution to (5.4) for the subgraph T' of T that results from discarding all leaves of T, i.e., by* pruning *T, can be converted to a solution to (5.4) for T in linear time.*

5.4.2 Solving CLOF in Forest Graphs

The main problem for solving (5.4) for forest graphs is that the greedy approach described in Theorem 5.4.4, will not work for forest graphs. E.g., consider the union of a linear graph L and a bifurcating tree T (Figure 5.7). Suppose f is an edge-valued cost function such that $f(L) = 10$, and $f(B) = 4$ for each of the three branches B connecting the bifurcation point to a leaf in T. The longest path (according to f) in the union of these graphs is L. However the maximal subforest with at most 3 leaves is T, which does not contain L.

Nevertheless, we are able to straightforwardly apply the algorithm solving (5.4) for tree graphs, to each separate connected component of the forest. From this, a solution for forest graphs is easily derived as follows.

Theorem 5.4.9. *Let $F = (V, E)$ be a forest graph with n nodes, and suppose f is a real-valued function, associating a positive cost to either each vertex or each edge of F. Given $k \in \mathbb{N}_{\geq 2}$, an exact solution to (5.4) can be computed in $\mathcal{O}\left(n + \beta_0(F)(\min(k, l_c)l_c n_c + l_c^{\beta_0(F)})\right)$ time using Algorithm 6, where $\beta_0(F)$ is the number of connected components in F, and l_c and n_c, respectively, are the maximal number of leaves and nodes included in a connected component of F.*

Proof. The components of F can be determined in $\mathcal{O}(n)$ time. The complexity term $\mathcal{O}(\beta_0(F)\min(k, l_c)l_c n_c)$ comes from applying Algorithm 5 to each connected component of F. After this, we can iterate over all possible combinations $\left(k_1, \ldots, k_{\beta_0(F)}\right)$ of leaves for each connected component, to obtain the overall

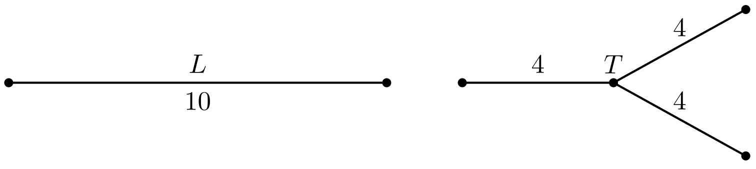


Figure 5.7: A greedy approach will always start from the path L, which cannot be extended to a forest containing three leaves. However, the optimal subforest with three leaves is T.

best corresponding sum of total costs, in $\mathcal{O}\left(\beta_0(F)l_c^{\beta_0(F)}\right)$ time. Note that the total cost from 2 up to $\min(k, l_c)$ for each component may also be stored during Algorithm 5, and does not need to be recomputed. □

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

```
Input: forest $F$, positive cost function $f$, number of leaves $k$.
Output: A subgraph of $F$ corresponding to the solution of CLOF.
1  treeSols = lapply(components($F$), function(T) Algorithm5(T, $f$, $k$))
           #apply Algorithm 5 to each connected component of F
           #return a list of all solutions (each solution is a list itself)
2  currentCost = -Inf #cost of the current best solution in F
3  for $0 \leq l_1, \ldots, l_{\beta_0(F)} \leq k$ do
4      if(sum($l_1, \ldots, l_{\beta_0(F)}$) $> k$) continue #skip if number of leaves is $> k$
5      thisCost = sum(cost(treeSols[[1]][1:$l_1$]), $\ldots$,
         cost(treeSols[[$\beta_0(G)$]][1:$l_{\beta_0(F)}$]))
               #evaluate the cost of the current potential solution for f
               #the cost of each subtree can be stored during the execution
               #of Algorithm 5 for fast evaluation
6      if thisCost $>$ currentCost then
7          currentCost = thisCost
8          bestSol = subgraph($F$, c(treeSols[[1]][1:$l_1$], $\ldots$,
             treeSols[[$\beta_0(F)$]][1:$l_{\beta_0(F)}$]))
               #update the current optimal solution
               #the solution is determined as the subgraph in F
               #induced by all partial solutions
9      end
10 end
11 return(bestSol)
```
Algorithm 6: **C**onstrained **L**eaves **O**ptimal sub**F**orest (CLOF) in forests.

5.5 f-Pines for Topological Data Analysis of Graph-Structured Data

We are now fully prepared to present our new method for topological data analysis of graph-structured data, the schematic overview of which is given in Figure 5.1e.

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

2. Based on the core measure f, we build an f-pine F in G (Definition 5.3.2) using the minimum spanning tree algorithm (Proposition 5.3.5). In case of weighted graphs where each weight represents a notion of distance between vertices, i.e., higher weights denote more distant nodes, we use the boundary coefficient (BC) as core measure (Definition 5.2.2). Its superior effectiveness over existing core measure for locating core structure near the underlying backbone of a graph, has been discussed in Sections 5.2.4, and qualitatively and quantitatively demonstrated in [4]. For unweighted graphs, we use the ordinary local cluster coefficient (LCC), due to its close relation to the boundary coefficient (Proposition 5.2.3, Corollary 5.2.4 & Proposition 5.3.4), and the extensive amount of research that has already been performed on both its theoretical and computational aspects [70, 89].

3. We solve CLOF for a well-chosen cost function g on either the vertices or edges in F (Section 5.5.1). We solve (5.4) for either a given number of leaves $k \in \mathbb{N}_{\geq 2}$ of interest, or a (possibly infinite) upper bound on the expected number of leaves.

4. Further analysis may be required to result in the final backbone topology. E.g., in the case of an unknown number of leaves where an upper bound was provided, visual inspection or an elbow locating method may be used to infer the number of leaves, which we discuss in Section 5.5.3 (see also Remark 5.4.5). A further step may be required to identify cycles that are missing a representation in the forest-structured backbone, which we discuss in Section 5.5.4.

5.5.1 Vertex Betweenness as Cost Function for CLOF

To effectively mine topological substructures through CLOF in forest representations, we need a function g defined on either the vertices or edges of the representations to optimize in terms of (5.4). We will mainly use the *vertex betweenness*, which equals how many shortest paths go through a particular node. Some other interesting options will be discussed in Section 5.5.2, in which we also discuss why we prefer the vertex betweenness over these.

The intuition for using this measure is as follows. By Proposition 5.3.3, many (boundary) nodes are not located on the backbone, but attached to it by means of a local minimum. To reach one (boundary) node from another, we must first travel to the backbone, then from one location on the backbone to another location, and thereafter leave the backbone to connect to the other node. Hence, nodes on the

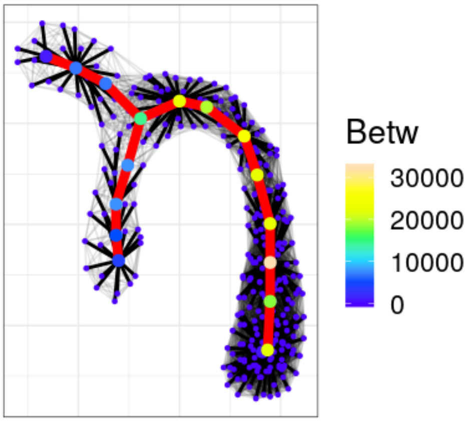


Figure 5.8: Optimal subgraph with 3 leaves in a pruned BC-pine according to the vertex betweenness (edges in red). Nodes with high betweenness represent important nodes for accessing many others, due to the uniqueness of paths between nodes in a forest.

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

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

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

- Not only is the vertex betweenness constant on leaves (Definition 5.4.7), making it a lot more efficient to solve CLOF by prepruning the forest representation (Theorems 5.4.8 & 5.4.6), it also equals 0 on leaves. This implies that a solution to (5.4) in the pruned forest representation equals a solution to (5.4) in the original representation.

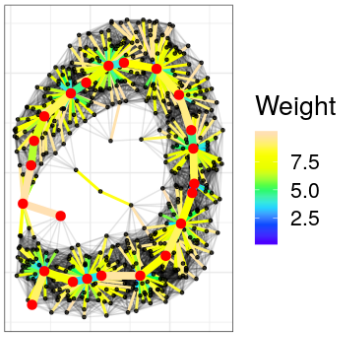
- Contrary to other interesting vertex/edge-valued functions (Appendix 5.5.2), the vertex betweenness also accounts for the global topological structure of the forest. Hence, given an effective forest representation, the relation between the vertex betweenness and the backbone substructure goes in both directions. This means that nodes with a high vertex betweenness correspond to nodes inducing the backbone substructure in the forest representation, and vice versa (Figure 5.8).

We emphasize that the same measure, in this case the betweenness centrality, might only be come topologically meaningful for identifying backbone structures in graphs through a forest representation. This can be seen by comparing Figures 5.4c &. 5.8.

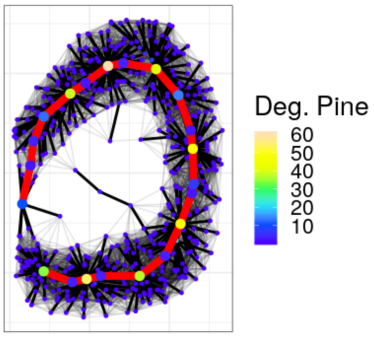
5.5.2 Other Cost Functions for CLOF

Above, we discussed the usefulness of betweenness centrality as vertex-valued cost function g used for identifying a subforest by means of (5.4) for the purpose of topological data analysis of graph-structured dat. Here, we discuss some other interesting choices.

The original edge weights As our backbone is meant to span the entire underlying topology of our given graph seen as a (shortest path) metric space, we may consider a longest or multiple longest paths in our f-pine to make up the backbone. E.g., the longest path shown in Figure 5.9a identifies the correct underlying model, apart from the location of its leaves, chosen to be the furthest points in the local noise around the true leaves.



(a) Optimal subgraph with 2 leaves in a BC-pine according to the original edge weights (vertices in red). A single edge weight is however not representative for whether the corresponding edge is important for inclusion in the backbone or not, as it is not based on the resulting f-pine. Using our current implementation, the algorithm described in Theorem 5.4.6 takes 56s to execute for the resulting BC-pine with 477 nodes, with no upper bound on k.



(b) Optimal subgraph with 2 leaves in a pruned BC-pine according to the vertex degree (edges in red). High degree nodes represent important nodes that should be included in the backbone, according to Theorem 5.4.8. The algorithm described in Theorem 5.4.6 now takes 0.1s to execute with no upper bound on k, a significant improvement compared to using the original edge weights as cost. This illustrates the power of Theorem 5.4.8 when working with pines.

Figure 5.9: Examples of solutions to (5.4) for cost functions other than vertex betweenness.

In terms of performance, this method is affected by the presence of outliers. E.g., the linear backbone in the pine shown in Figure 5.9a is of similar length as the path in the pine that takes a turn to pass pass through the centered outlier. Though we got lucky in this case, we note that such 'interbranching regions of outliers' are often present in many practical examples, such as in cell trajectory data, as discussed by [9].

In terms of scalability, note that our f-pine generally has many leaves due to Theorem 5.3.3. If we take a look at the case where our original graph is connected, i.e., where the f-pine F is a tree graph, then this implies that the number of leaves l in F may be of order n, where n is the number of nodes in F. If one fully grows the pine by through Algorithm 5, and consequently estimates an appropriate number of leaves k as discussed in Section 5.5.3, Theorem 5.4.6 implies that this computation may be close to cubic in n. Its memory usage will be close to squared in n, due to storing the pairwise distance matrix between leaves and all other nodes. Hence, apart from often not having a meaningful interpretation (Figure 5.9a), allowing the inclusion of any leaf of the pine makes our current implementation for solving (5.4) difficult to scale to larger data sets.

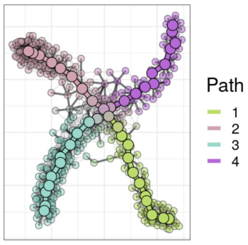
The degree of a vertex By Proposition 5.3.3, (locally) high degree nodes represent local minima of f in an f-pine F. Given f is a core measure where low values indicate core nodes, these are exactly the nodes where we want our backbone to pass through. Hence, we may use the vertex-valued degree function $g \equiv \delta_F$ for optimizing (5.4). The result for $g \equiv \delta_F$ is less affected by outliers due to their low density in the original graph.

This cost function g is constant on leaves by definition. Hence, we may apply Theorem 5.4.8 to first prune F, often leading to a significant reduction of the graph size due to Proposition 5.3.3. In terms of Theorem 5.4.6, this implies that both terms l and n decrease significantly, leading to a much better computation time, as well as storage cost (which is $\mathcal{O}(l \cdot n)$).

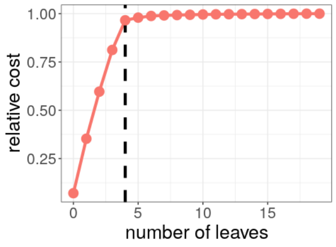
Figure 5.9b shows the resulting solution of (5.4) in a pruned BC-pine. High degree nodes correspond to core nodes in the backbone, but not necessarily conversely. Though extending the two leaves of the linear backbone by connecting each one of them to an arbitrarily chosen neighboring leaf results in the optimal solution of (5.4) in the original pine (Theorem 5.4.8), we do not conduct this step as this will again introduce randomness to the choice of leaves.

5.5.3 Estimating the Number of Leaves

Solving CLOF requires one parameter as input, namely the number k of leaves to be included in the backbone. For the purpose of topological data analysis of graphs, this parameter is ideally inferred from the data itself. As discussed in



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



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

Figure 5.10: Extracting the optimal result to (5.4) for a tuned number of leaves $k = 4$ in a BC-pine using vertex betweenness as cost. The pine was obtained for a Rips graph ($\epsilon = 8$) on a 2D point cloud data set with an underlying X-shaped topology.

Remark 5.4.5, the fact that CLOF can be solved through a greedy algorithm admits a convenient way to perform this estimate. We will consider the case where our original graph G is connected, i.e., the resulting pine is a tree graph T. In case of disconnected graphs, our current approach is to estimate the number of leaves for each component separately.

As the cost of the subtree increases with each iteration of the algorithm described in Theorem 5.4.4, we can track the increase in cost according to the added number of leaves. Initially, the increase in cost is high when we add true branches to our current subtree. When all true branches are added, we start connecting our subtree to surrounding noise or outliers, and the increase in cost drops. This is illustrated in Figure 5.10, which also shows that an 'elbow' inference method may be used to tune the number of leaves. This may be done either visually, or by a using an automated procedure, such as minimizing the second-order finite differences of the function shown in Figure 5.10b. Note that we can easily extract any solution with fewer leaves from the current solution (Remark 5.4.5)

5.5.4 Identifying Missing Cycles

As by Definition 5.3.1 a spanning forest may never include a cycle, we are unable to use any of its subgraphs for representing the underlying topology of a graph if the true underlying model contains cycles. However, as we will illustrate in this section, identifying a simplified underlying forest-structured topology can be highly beneficial for identifying cycles missing in the backbone representation of the underlying topology. We emphasize that though the approach discussed in this section is still experimental, it leads to effective results in practice.

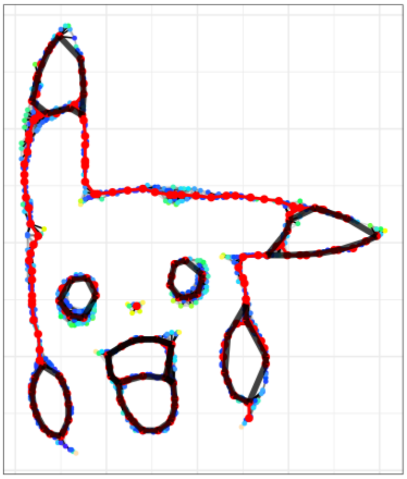
Consider the forest-structured backbone B we constructed throughout a point cloud data set X of 807 observations representing Pikachu in the Euclidean plane, by means of our method for TDA of graph-structured data sets (Figure 5.11a). We used a Rips graph G with $\epsilon = 3.5$ as a graph modeling the underlying topology of Pikachu.

Figure 5.11b shows four different persistence diagrams resulting from D (Section 2.5.2). One for the metric space $(X, d_{\text{euclidean}})$, one for (X, d_G) where d_G denotes the shortest path metric on the Rips graph G, one that results from approximating the diagram for (X, d_G) through the method described by [50], and finally one for $(V(B), d_G)$.

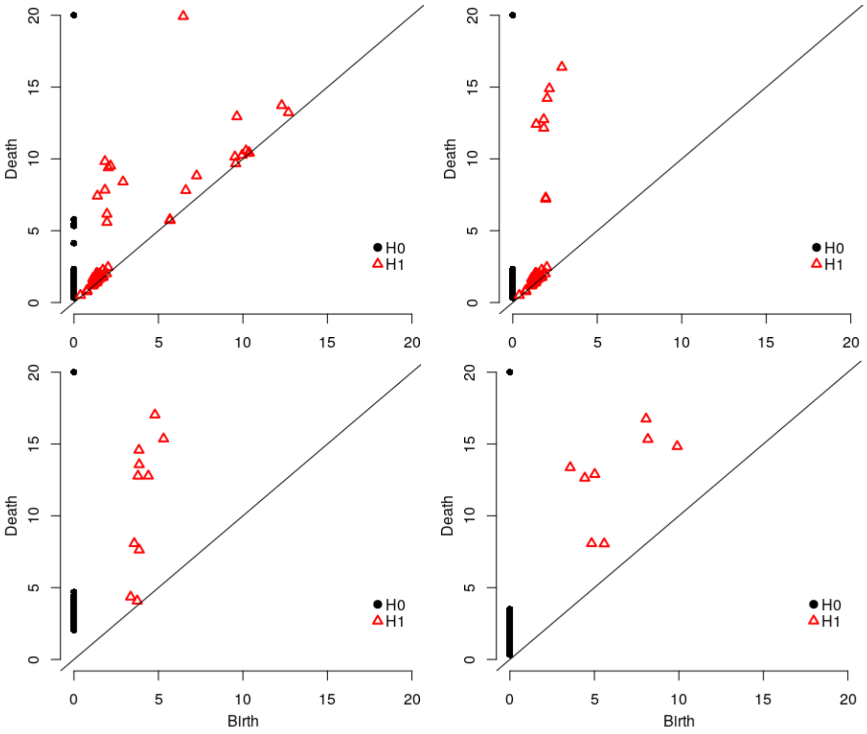
The diagram for the original point cloud data (Figure 5.11b, Top Left) displays many long persisting cycles, as well as some cycles corresponding to topological noise (holes with a low persistence). The diagram for (X, d_G) (Figure 5.11b, Top Right) does not include the original large cycles with a large birth value anymore. These cycles are never born due to infinite distances between nodes in different connected components of G. However, the topological noise with a low birth value remains. A small amount of topological noise also remains in the approximated diagram for (X, d_G) (Figure 5.11b, Bottom Left). However, the diagram for $(V(B), d_G)$ (Figure 5.11b, Bottom Right) also disposes of the topological noise, and what remains is exactly one point (H1) for each one of the eight 'gaps' that are still present in B. One for each of Pikachu's two cheeks, ears, and eyes, one for its lip, and one for its tongue. These gaps are characterized by two nodes of B lying close to each other in the original graph G, i.e., according to d_G, but not in B, i.e., according to d_B.

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

Furthermore, depending on the used implementation, the computation of persistent homology also allows one to locate a cycle that represents the hole corresponding to each one of the points (H1) in the diagram [90]. This allows one to locate the cycles that are missing a representation within the backbone topology. These cycles are shown in Figure 5.11a. Note that however, the cycle is computed to be underlying the metric space $(V(B), d_G)$, and might not correspond to an actual subgraph of G.



(a) A forest-structured backbone graph B (edges and nodes in red) for a point cloud data set X resembling Pikachu. Persistent homology (Figure 5.11b) allows one to identify cycles that should be represented in the backbone B (edges in black).



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

Figure 5.11: Compared to the standard approaches to persistent homology, our forest structured backbone inferred through solving CLOF in the BC-pine allows for a more effective and efficient approach to identify cycles in Pikachu.

5.6 Backbone Inference as a Facility Location Problem in Networks

In this section, we discuss how we can regard our method for backbone inference as a *facility location problem* in networks, and discuss how it differs from the existing approaches in this context.

The general setting of *Facility Location Problems in Networks* [91] is:

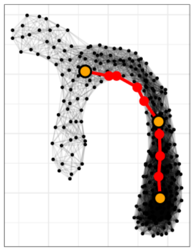
"given a graph G, a collection $\mathcal{F}$ of subgraphs of G,
and a cost function $f : \mathcal{F} \to \mathbb{R}$,
optimize $f(F)$ *subject to* $F \in \mathcal{F}$."

Note that the term 'facility' in our context refers to 'backbone'. Both the inference of f-pines (Section 5.3) and solving CLOF (Section 5.4), can immediately be seen as location problems in networks according to this formulation. A more commonly known example of subgraph inferred through a facility location problem is the minimum spanning tree (MST). Here, $\mathcal{F}$ is the set of spanning trees of G (forests if G is disconnected), and f maps a tree onto the sum of the weight of its included edges, which is the cost to be minimized. *Steiner trees* generalize this concept. They minimize the same cost function as minimum spanning trees, but are only required to cover a given set of nodes, called *terminals*. Finding a minimum spanning tree can be done in linear time [86], whereas finding a Steiner tree is an NP-hard problem [92]. In Section 5.7, we show that neither facility is effective for inferring backbones modeling the underlying topology of a graph well.

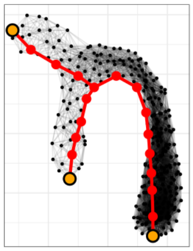
Existing facility location problems come with a variety of issues that prevent them to effectively identify and locate backbones in graphs. These issues, summarized below, are the main reasons why we introduced the forest representation as an intermediate step for mining topological substructures, as we overcome all of these by designing such representation of our graph.

Computational complexity Many facility location problems in networks are NP-hard for general graphs [91]. Certain formulations even lead to NP-hard problems when the original graph is a tree graph [95]. In contrast to this, we presented effective and efficient algorithms that provide an exact solution to our introduced facility locations problems, which are identifying an f-pine and solving CLOF in forest representations (Section 5.4).

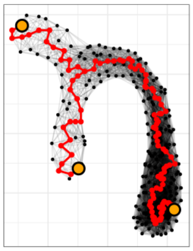
Sensitive to outliers Outliers are harmful when either the constraint $\mathcal{F}$ [96] or the cost f [97] specifies that all nodes in the original graph should lie close to the facility. Furthermore, facilities may 'pass through' outliers to reach one region



(a) An approximated Steiner tree [93] in red, which we constructed through three terminal nodes/medoids selected by a partitioning around medoids (PAM) algorithm (orange). The selection of these medoids is regarded as a facility location problem in metric spaces [94], and is highly biased towards dense regions.



(b) A subgraph (red) obtained by iteratively choosing farthest points, and connecting them by the shortest path between them and the current tree structure. These paths always take 'shortcuts' when available, shifting them away from the true core in the presence of curvature. Furthermore, outliers are especially harmful when connecting to farthest points (Section 5.7).



(c) The output (red) of our method presented in Figure 5.1e, where we replaced the BC-pine as a forest representation by the ordinary minimum spanning tree (MST). The selected leaves during CLOF are shown in orange. Subgraphs minimizing the maximum edge weight, such as the MST, are biased towards including low-weight edges, leading to 'wiggled' results.

Figure 5.12: Subgraphs mined from an original graph G (black), (c) with and (a-b) without intermediate forest representation. Comparing these results with Figure 5.1d, (a-b) illustrate the usefulness of an intermediate forest representation for mining topological substructures in graphs, whereas (c) illustrates the importance of designing an effective representation for this purpose, both the subject of this paper. Note that all methods may be regarded as a combination of selecting important nodes and constructing a subgraph through these. We will discuss these methods in detail in Section 5.7.

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

Sensitive to density To overcome the sensitivity to outliers, the constraint or cost may be postulated in terms of the average/mean distance of the facility to all other nodes [98]. However, such approach tends to fail revealing important structure in case of a non-uniform density across the underlying topology (Figure 5.12a). In contrast to this, our method effectively infers backbones that extend across the entire original graph, while remaining near the core of the graph (Figure 5.1d).

Not or too topologically constrained Facility location problems are mainly considered in topics such as routing, logistics, and dispatching [99]. In these scenarios, rather than representing the topological model underlying the graph, the objective of the facility is to reach all nodes of the original graph as close as possible, while maintaining a low cost of the facility. These facility location problems are insufficient for inferring topological models underlying graphs. There may not be any constraints on the topological complexity of the facility (e.g., in terms of the number of leaves or multifurcations), allowing for an arbitrary complex backbone that fails to provide insight into the underlying structure. E.g., the only topological restriction on the facility may simply be that the facility is a tree [98]. In other cases, the facility is topologically too constrained. In particular, facility location problems may also search for a specific path [100], which is not suited to capture the underlying topology in many practical examples.

Steiner trees allow some control over the topological complexity of the final facility through the number of terminals that are specified [101]. However, the presence of outliers or non-uniform density may be harmful when selecting these terminals in an unsupervised manner (Figure 5.12a). Furthermore, the topological complexity, such as the number of leaves, of the resulting (approximated) Steiner tree is often not consistent with the number of terminals (Figure 5.12a and Section 5.7).

In contrast to these methods, our objective is to reveal the underlying topology of a graph—the cost of which does not matter to us—in a robust and effective way. For this reason, we are able to provide a method for tuning the topological complexity of our backbone in a data- and scale-independent way (Section 5.5.3).

Topological bias Many facilities may just not be meaningful representations for the underlying topology. E.g., a trivial example is the longest path through a graph (if existing), which may 'wiggle' through the entire graph without reflecting the true underlying topology, even if this is linear. Furthermore, other facilities may only be meaningful in the absence of outliers (as also discussed above), in the absence of curvature (Figure 5.12b), or may be biased to include mostly low-weight edges due to the minimization of a sum or maximum of the edge weights of the facility. Extreme examples of this are the MST and its subgraphs, which 'wiggle' through the entire graph (Figure 5.12c).

5.7 Experiments: Inferring Backbones in a Variety of Graphs

In this section, we show that our method is applicable to a wide variety of graphs arising from different fields of science. For our applications, we will focus on topological data analysis, visualization, and graph simplifications. Note that graph simplifications recently showed to increase the performance of existing graph embedding methods [102]. Our method will show to be applicable to a wide variety of data sets, from social networks, to high-dimensional point cloud data.

We will present the ten different data sets on which we will conduct our experiments in Section 5.7.1. In Section 5.7.2, we will discuss the baseline methods we will use to verify the effectiveness of our newly introduced method on these data sets. In Section 5.7.3, we qualitatively discuss our obtained results. Section 5.7.4 considers the introduction and results of our quantitative metrics used to measure the performance of our method. In Section 5.8, we will also conduct a separate large scale experiment on 333 cell trajectory data sets, using a domain specific baseline and set of quantitative measures.

5.7.1 Summary of the used Data Sets

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

Swiss Roll (SR) The Swiss Roll is a commonly used manifold for analyzing the performance of nonlinear dimensionality reductions [103]. We generated a synthetic data set of 1000 points lying on such manifold. A Rips graph with 47 013 edges ($\epsilon = 0.75$) was constructed from this data for analysis through our method.

Karate Network (K) Zachary's karate club is a well-known social network of a karate club, studied by Wayne W. Zachary for a period of three years from 1970 to 1972 [12]. The network consists of 34 members of a karate club. Each one of the 78 edges between pairs of members denotes an interaction outside the club. During the study a conflict arose between the—under pseudonyms known—administrator "John A" and instructor "Mr. Hi", which led to the split of the club into two. Half of the members formed a new club around Mr. Hi, whereas members from the other part either found a new instructor or gave up karate. This splits the network into two ground-truth communities. Furthermore, each edge is weighted with the the number of common activities the club members took part of. We mapped each edge weight to its inverse, as higher weights corresponded to more distant nodes when we introduced the BC [4].

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

Game of Thrones Network (GoT) We consider an unweighted network G defined on 208 characters of the Game of Thrones saga[2]. This graph is obtained from `https://shiring.github.io/networks/2017/05/15/got_final`. Each one of the 326 edges between two nodes denotes either a (undirected) 'mother', 'father', or 'spouse' relationship.

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

Earthquake Locations (EQ) We obtained a data set containing information on 80 549 earthquakes ranging between the years 1950 and 2017. This data is freely accessible from USGS Earthquake Search. The topology underlying such a data set was already analyzed by [13] and [3] (see also Figure 4.7a). However, contrary to their followed procedure, we do not restrict ourselves to a particular small rectangular domain with a low amount of noise, do not apply any noise filtering in advance, and are able to obtain a topological simplification through a kNN graph. A random sample of 5000 earthquake locations was taken, from which we constructed an undirected 10NN graph with 31 129 edges using the Great circle (geographic) distances between location coordinates for analysis. These distances where also used as weights of resulting edges.

Cell Trajectories We will first demonstrate the effectiveness of our method for cell trajectory inference or visualization by means of a synthetic cell trajectory data set of 556 cells in a 3475-dimensional gene expression space (**SC**). This data represents a snapshot of these cells at a specific point during a cell differentiation process. Different stages in the differentiation process correspond to differentially expressed genes. Hence, the ground-truth underlying topology of the point cloud gene expression data set is the (embedding of the) differentiation network. A kNN graph ($k = 10$; 4646 edges) will be used to analyze the underlying topology of this

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

data. We will conduct a similar experiment on a kNN graph ($k = 5$; 1543 edges) constructed from a real cell trajectory data of 355 cells embedded into a 3397-dimensional gene expression space (**RC1**), as well as on a kNN graph ($k = 10$; 974 edges) constructed from a second real cell trajectory data set of 154 cells in a 1770-dimensional gene expression space (**RC2**). We will use the latter data set to show how a dimensionality reduction can significantly improve our obtained results. We will use Euclidean distances to construct these graphs, and for the corresponding edge weights.

5.7.2 Summary of the Baseline Methods

We will evaluate the performance of our method by comparing our results to those obtained through three different baselines, chosen to address the following questions (Figure 5.12).

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

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

Facility Location + Steiner Tree (FacilitySteiner) We will use this baseline will to investigate the applicability of Steiner trees to our problem. First, we use an intermediate step based on facility location *in metric spaces*. We start by selecting *k medoids* through a partitioning around medoids (PAM) algorithm [94]. PAM is a clustering algorithm reminiscent to the ordinary k-means algorithm [104], but chooses data points as centers (medoids) and can be used with arbitrary distances. Once these medoids have been selected, we pass them to an algorithm for approximating a Steiner tree through these nodes, as described by [93]. The result of this method is illustrated in Figure 5.12a.

Farthest Point Sample (FarthestPoint) Our second baseline method is inspired by the algorithm for solving CLOF in tree graphs (Theorem 5.4.4), which includes a farthest point sampling according to a user-defined cost function. Instead, we apply a similar procedure to the original graph. We start with either the center of the graph (if we only wish to select one node), or the longest shortest path between two nodes of the graph (measured according to the original distances). Consecutively, we connect the next farthest point to the current tree by means of the shortest path between them, until a prespecified number of points has been selected. The result of this method is illustrated in Figure 5.12b.

CLOF in the Regular Minimum Spanning Forest (CLOFinMSF) Our third baseline method is also inspired by our algorithm for solving CLOF in tree graphs. However, this time we solve CLOF in the regular minimum spanning forest (MSF) constructed from the original graph. The purpose of this baseline is to illustrate the effectiveness of representing graphs through BC/LCC-pines for mining topological substructures through CLOF. The result of this method is illustrated in Figure 5.12c.

5.7.3 Qualitative Analysis of the Results

Swiss Roll Figure 5.13 shows our considered point cloud data X lying on a 'Swiss Roll'. Figures 5.14a-5.14d show the Rips graph constructed G from this data, as well as the backbones obtained through our various procedures. The 2D embedding of X was obtained through an Isomap embedding of X based on G [103].

Various observations can be made from Figure 5.14. First, FacilitySteiner performs well in terms of centering the backbone and its smoothness (Figure 5.14a). However, it is unable to fully extend to the true 'outside' of the backbone, as the

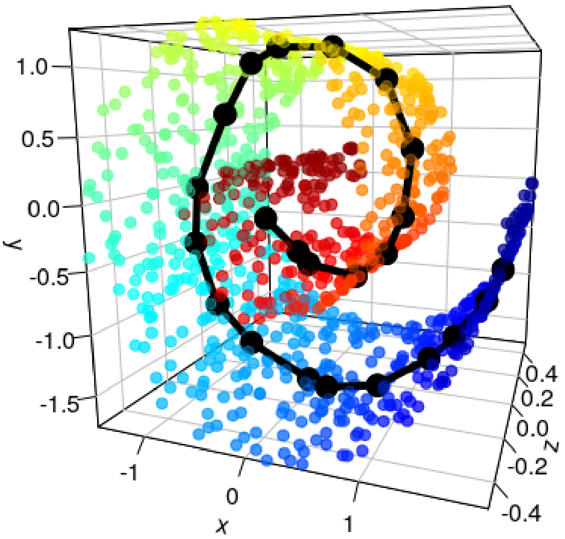
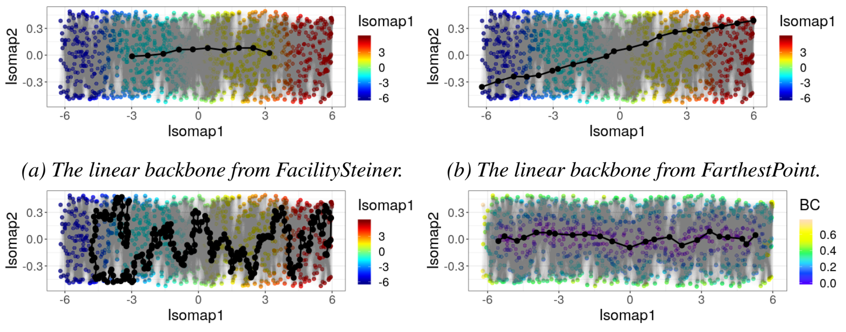


Figure 5.13: 3D point cloud data X lying on a 'Swiss Roll'. Points are colored according to the first coordinate of the 2D Isomap embedding of X based on G. The backbone obtained through our method curls all the way around the core of the manifold.



(a) The linear backbone from FacilitySteiner. *(b) The linear backbone from FarthestPoint.*

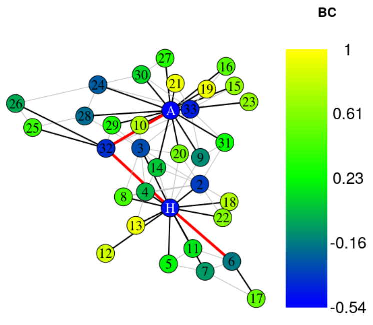
(c) The linear backbone from CLOFinMSF. *(d) A linear backbone from the BC-pine.*

Figure 5.14: Various backbones (black) through a 'Swiss Roll'-shaped point cloud data set. Only through the BC-pine, we are able to infer a smooth and centered backbone that extends to the true leaves of the linear-structured model that underlies the data.

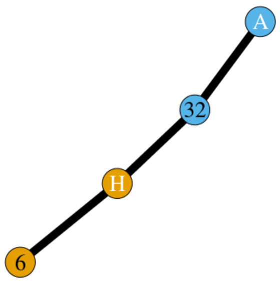
selection of medoids is not analogous to the selection of leaves through CLOF. FarthestPoint performs better in terms of fully extending to the true underlying leaves (Figure 5.14b). However, the resulting backbone crosses the topology diagonally instead of through its center, as it searches for the *maximal* shortest path between two nodes. CLOFinMSF performs well in approximating a vast majority of the nodes in G, as it 'wiggles' through the entire graph. The result of CLOFinMSF is however far from smooth, and it also does not extend to the true underlying leaf on the left (Figure 5.14c). In contrast, our newly introduced method of mining the backbone through solving CLOF in a BC-pine performs well in terms of centering the backbone, while also fully extending it to the true underlying leaves of G (Figure 5.14d).

Karate Network Figure 5.15a shows the original Karate Network, the BC-pine, as well as a linear backbone mined from this BC-pine. Figure 5.15b displays the backbone where nodes are colored according to their ground-truth community. Given the ground-truth separation of two communities, a linear backbone fits our network well. Furthermore, this separation of the two communities is preserved by our backbone (Figure 5.15b). We further remark that both John A (**A**), as well as Mr. Hi (**H**), achieve a very low BC (Figure 5.15a), which coincides with these nodes being highly transmissive nodes in the ground-truth model.

Harry Potter Network Figure 5.16 shows the original Harry Potter Network G, an LCC-pine in G, a forest-structured backbone B mined from this LCC-pine, and a representative cycle obtained through persistent homology of $(V(B), d_G)$ (Figure 5.17). It turns out that only our newly proposed method and FacilitySteiner were able to actually capture Harry Potter—the main protagonist—within



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



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

Figure 5.15: Our method identifies a linear backbone in the Karate network, consistent with the ground-truth separation of the network into two different communities.

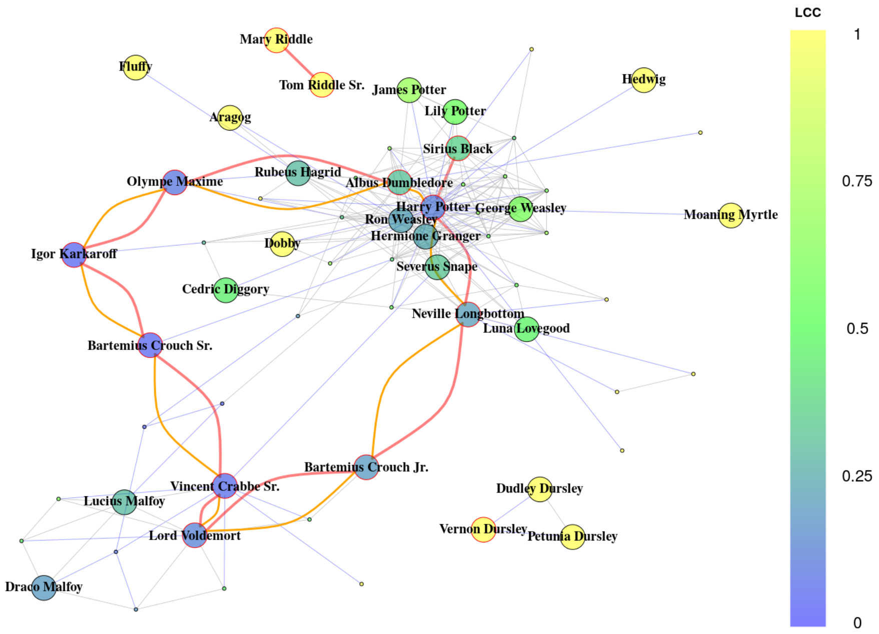


Figure 5.16: A backbone (red edges + red vertex borders) mined from the LCC-pine (blue edges) constructed from the Harry Potter network (grey edges). Persistent homology of the metric space induced by the original metric in G on the nodes of the backbone can be used to find a representative cycle missing in the backbone (orange edges). Layout provided through a force-directed layout algorithm.

the major component of the forest-structured backbone.

The other smaller components correspond to special cases. We let the backbone component of the single isolated edge of “Riddles” simply be itself (note that both vertices have betweenness 0 in any spanning forest of the graph). The component in the LCC-pine corresponding to the triangle of “Dursleys”, is pruned to a single node (Theorem 5.4.8), which is chosen as the representation of this component. Note that we also did this to obtain a representation of Pikachu’s nose in Figure 5.11a. However, unlike for the component representing this nose, all nodes in the triangle of “Dursleys” have an LCC of 1. Hence, there is no meaningful interpretation to the LCC-pine having chosen Vernon Dursley as the inner node of the corresponding linear component consisting of these three nodes in the pine.

Connoisseurs of the saga may be surprised by Harry Potter and Sirius Black being quite distant from Albus Dumbledore, according to the linear backbone component representing the major component of G, first needing to pass through Lord Voldemort. This is because of the lack of ability to include cycles through a (sub)forest representation of our original graph. However, as can be seen from Figure 5.16, this linear component (red edges) goes all the way around through the corresponding component. Its leaves are actually very close to each other according to d_G. As discussed in Section 5.5.4, persistent homology allows us to discover that a cycle is missing from our backbone representation (Figure 5.17). Furthermore, Figure 5.16 also displays the representative cycle corresponding to the single identified hole in the underlying topology (orange edges), placing Harry much closer to Dumbledore in the underlying topology of the graph.

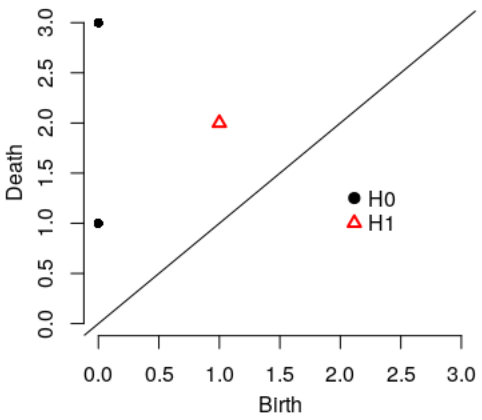


Figure 5.17: The persistence diagram of (B, d_G) in the Harry Potter Network reveals the presence of one cycle (H1) missing in the forest-structured backbone representation. Note that the multiplicity of the top left point (H0) equals three, i.e., the number of connected components in B according to d_G.

Game of Thrones Network Figure 5.18 shows the original Game of Thrones Network G, an LCC-pine in G, a forest-structured backbone B mined from this LCC-pine, and the union of the representative cycles for the two most persistent holes obtained through persistent homology of $(V(B), d_G)$. Nodes are colored according to their ground-truth community, i.e., the family they belong to.

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

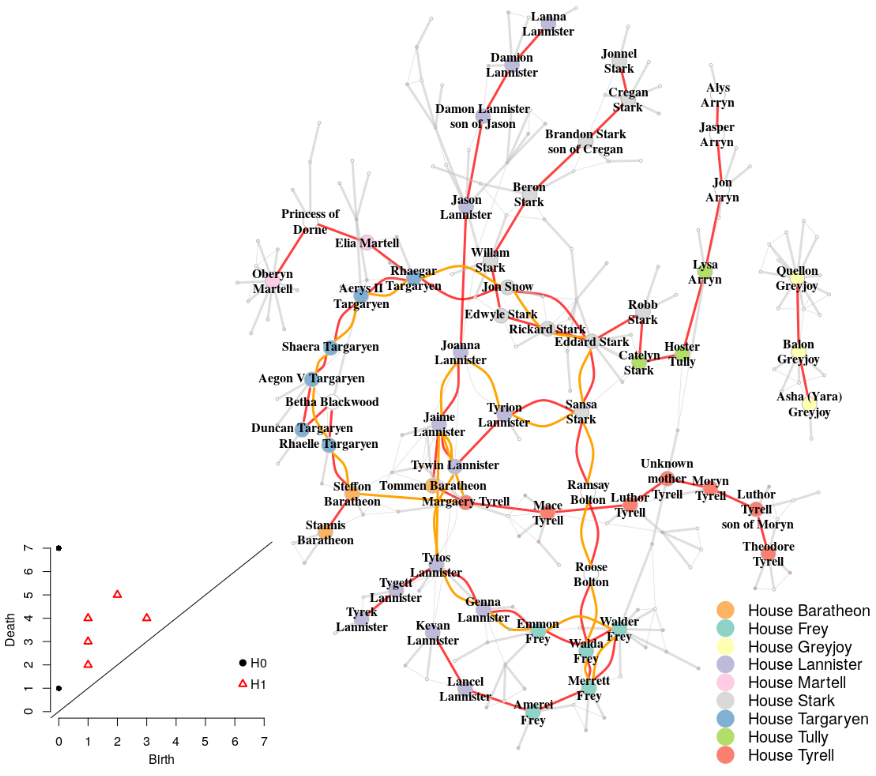


Figure 5.18: A forest-structured backbone with 10 leaves (red edges) mined from the LCC-pine constructed from the Game of Thrones network (grey edges). Nodes are colored according to their ground-truth community, i.e., the House they belong to. White nodes correspond to members that are unassigned to any house. Persistent homology of the metric space induced by the original metric in G on the nodes of the backbone is used to identify cycles missing in the backbone (orange edges). Layout provided through the Fruchterman-Reingold layout algorithm.

the number of leaves for the smaller component to be represented in the backbone, or manually specify the number of leaves for each component.

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

There are again some obvious 'gaps' in our backbone B. E.g., Sansa Stark immediately connects to a branch of Lannisters through Tyrion Lannister, but to reach the other cluster of Lannisters, she must first travel through the Boltons, and then through House Frey, making it apparent that these groups of Lannisters form seperated communities. Another obvious gap is present in House Baratheon, as we first need to travel through House Lannister, House Stark, and House Targaryen before reconnecting this community. Though some smaller gaps seem to be present as well (and may also be identified using persistent homology), the ones discussed above correspond to the two most persistent holes one obtains through persistent homology of the metric space $(V(B), d_G)$ (Section 5.5.4). The union of the two corresponding representative cycles are shown in Figure 5.18 as well. Note that these cycles overlap through a path between Jaime Lannister and Sansa Stark.

KDD & NeurIPS Co-authorship Network We first applied our method to construct a tree-structured backbone with 5 leaves in the KDD co-authorship network. The resulting tree graph is shown in Figure 5.19.

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

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

For comparison, we compared this to the same measures for the tree representations corresponding to our baselines. This equals the regular MST for CLOFin-MSF, which is the solution to (5.4) for the maximal possible number of points (leaves) that can be selected through the algorithm in the MST (Theorem 5.4.4).

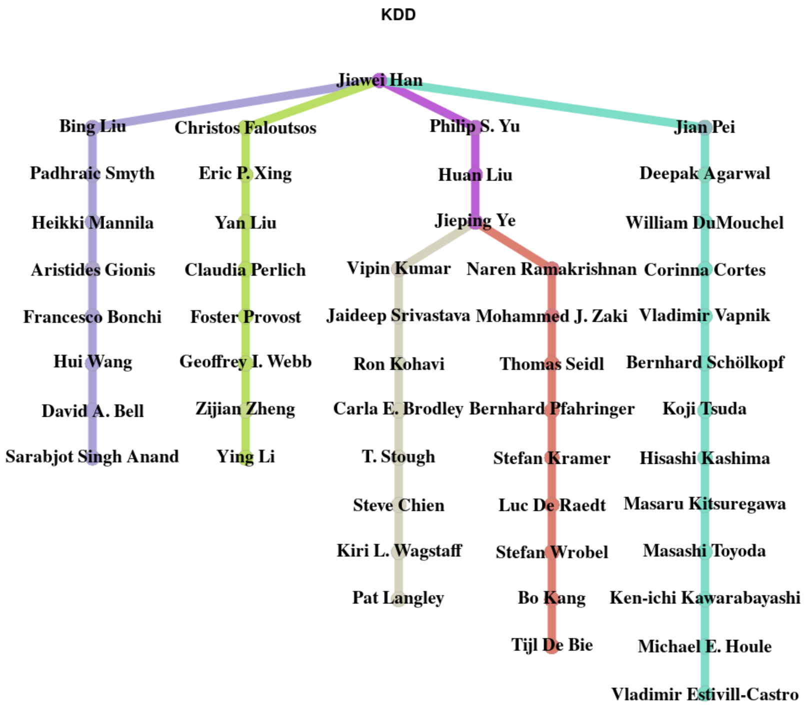


Figure 5.19: A tree-structured backbone for the KDD co-authorship network with 5 leaves. Nodes and edges are colored according to their closeness to one of the 6 resulting branches. Layout provided through a tree layout algorithm.

Since FacilitySteiner and FarthestPoint do not make use of an intermediate forest representation for inferring a backbone, we analogously consider the trees that result from selecting the maximal number of nodes during the algorithm. For FacilitySteiner, this equals the Steiner tree induced by all nodes, and hence, also the regular MST. For FarthestPoint, we consider the tree that results from continuing the sampling until all nodes all included. We will refer to this tree as the *FPS tree*.

The obtained metrics according to the number of pruning iterations are shown in Figure 5.20. By iteratively pruning leaves, we note that the BC-pine retracts much faster to a core structure than the regular MST, discarding the majority of the nodes after the first iteration, a consequence of Proposition 5.3.3. The FPS tree quickly discards of many nodes as well. However, the BC-pine retracts towards a core structure marking authors with a high number of citations, and who have been present very early in the considered conference. This is exactly what we expect from the core structure of the original graph.

Note that through a similar analysis, we showed the effectiveness of the BC over the LCC even in non-metric weighted graphs [4].

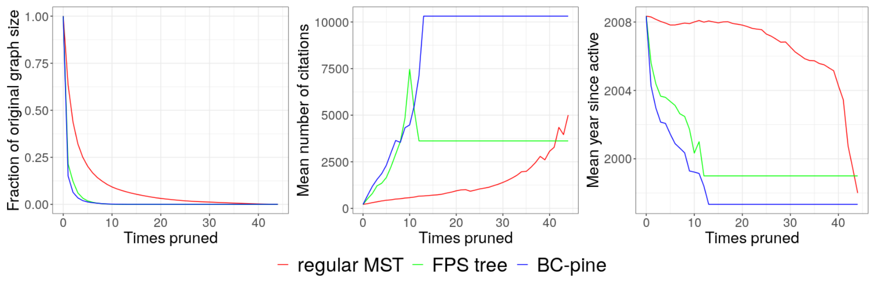


Figure 5.20: Various measures after iteratively pruning the regular MST (red), the FPS tree (green), and the BC-pine (blue). (Left) Fraction of original graph size. (Middle) Average number of (KDD) citations. (Right) Average year of first published (KDD) paper.

We applied the exact same method to construct a tree-structured backbone with

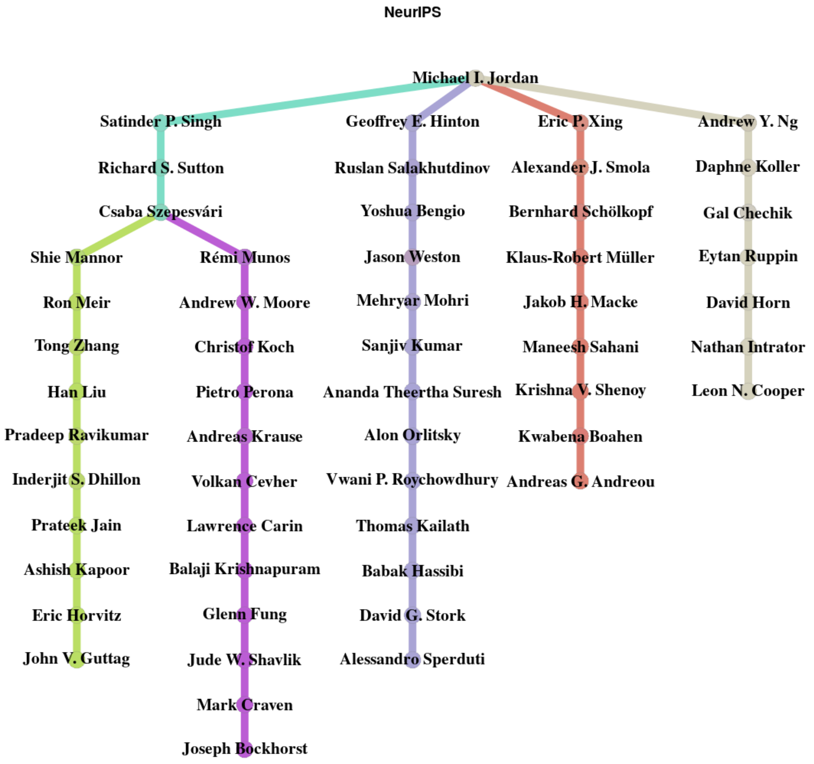


Figure 5.21: A tree-structured backbone for the NeurIPS co-authorship network with 5 leaves. Nodes and edges are colored according to their closeness to one of the 6 branches. Layout provided through a tree layout algorithm.

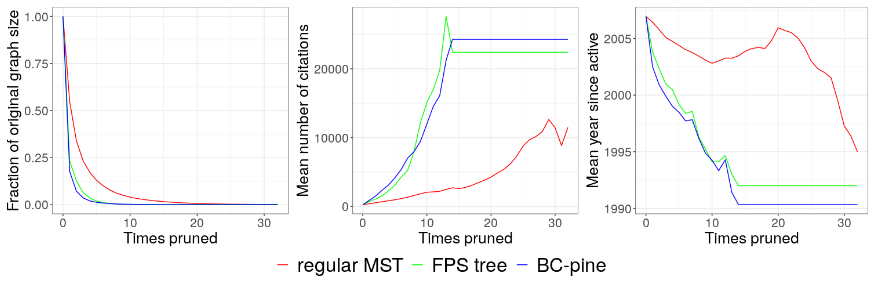


Figure 5.22: Various measures after iteratively pruning the regular MST (red), the FPS tree (green), and the BC-pine (blue), for the NeurIPS network. (Left) Fraction of original graph size. (Middle) Average number of (NeurIPS) citations. (Right) Average year of first published (NeurIPS) paper.

5 leaves in the NeurIPS co-authorshop network. The resulting tree graph is shown in Figure 5.21.

As we did for the KDD network, we verified that the cores of our tree representations, i.e., the BC-pines, indeed correspond to meaningful core structures (Figure 5.22). We observe that our method provides consistent results, as again, the BC-pine retract towards a core structure marking authors with a high number of citations, and who have been present very early in the considered conference.

Earthquake Locations Figure 5.23 shows the result of constructing a forest-structured backbone for our graph representing the earthquake locations through the three baseline methods, and our newly introduced method. Each method was specified to select 35 nodes.

FacilitySteiner is again unable to extend across the entire underlying topology, leaving large patches unrepresented. The resulting backbone also only contains 23 leaves. The backbone resulting from FarthestPoint *connects* to many outliers, prone to be selected through this method. Furthermore, the backbone also *passes through* outliers, as shortest paths in the original graph will take any ‘shortcut’ available. The backbone also only contains 26 leaves. This is due to leaves added at one iteration being connected to other leaves in further iterations. In contrast, both backbones obtained through CLOF exactly contain 35 leaves as specified. Both extend well across the entire underlying topology, while avoiding outliers. Note that the regular MST avoids connecting through outliers through multiple edges as the corresponding weights are usually high, whereas the BC-pine avoids connecting through these as they are separated from the true core by means of boundary nodes. The main difference between the backbones obtained through CLOF is their size: through the BC-pine, we approximate the entire underlying topology of the graph with less than three times the nodes than through the regular

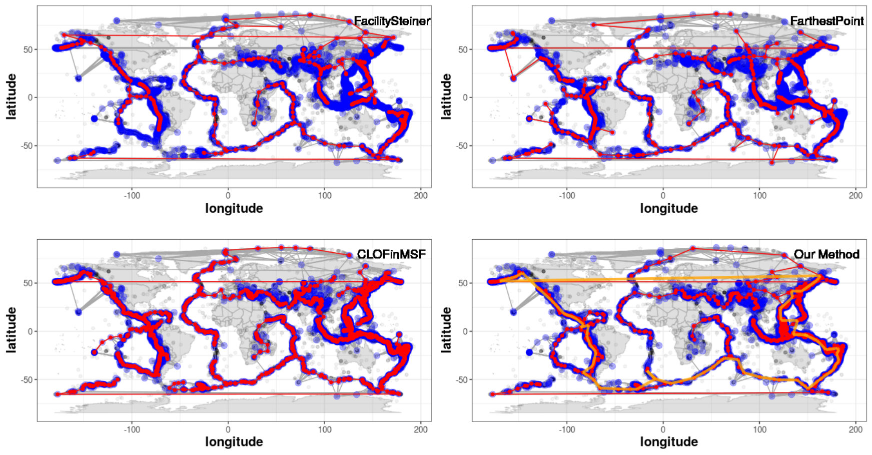


Figure 5.23: Backbones (red) derived from earthquakes scattered across the Earth using various methods. (Bottom Right) A representative cycle for the most persisting hole in (B, d_G) identified through topological persistence is overlayed in orange. Only through the BC-pine, we are able to infer a backbone that passes smoothly through the entire graph, while avoiding outliers. We quantitatively verify this in Section 5.7.4

MST (Table 5.2). This is again a consequence of CLOFinMSF resulting in a very 'wiggled' backbone.

Figure 5.24 shows the persistence diagram of $(V(B), d_G)$—B being the backbone derived through our newly introduced method—which indicates that there are

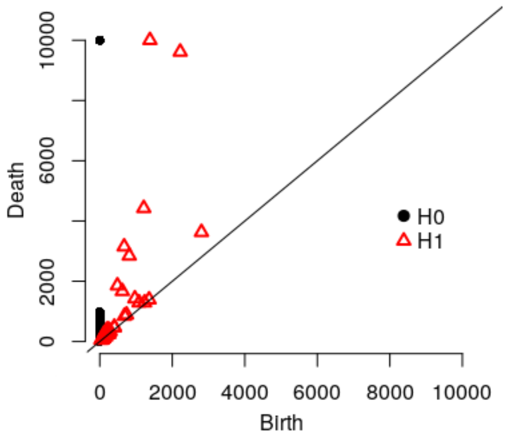


Figure 5.24: The persistence diagram of (B, d_G) reveals the presence of many small, medium, as well as larger cycles missing from the forest-structured backbone B derived through our newly introduced method.

many small, medium, as well as larger cycles missing from the forest-structured backbone. Figure 5.23 (Bottom Right) also displays a cycle representing the most persisting hole, spanning the entire Earth.

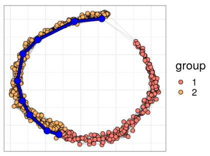
Cell Trajectories Figures 5.25 & Figures 5.26 visualize our obtained results for the first two cell trajectory networks discussed in Section 5.7.1, as well as the known ground-truth underlying topologies (Figures 5.25e & 5.26b). The point cloud data, as well as the obtained kNN graphs and the resulting backbones, are visualized on multidimensional scaling plots using Pearson correlations as distances between cells.

It should be noted that even though the constructed proximity graph may incorrectly connect different parts of one or multiple branches, our method is well able to both correctly identify and locate the present linear topology. E.g., Figure 5.25d shows that even though the kNN graph forms a cyclic structure through the entire data, our method is able to correctly identify and locate the ground-truth linear underlying topology. In contrast, each one of our baseline methods incorrectly connects the two underlying leaves (Figures 5.25a-5.25c).

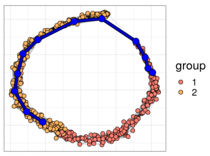
Even without CLOF to infer topologies from our pine, Figure 5.25g & 5.26d show that the BC-pine visualizes the ground-truth topologies present in the data well through the Fruchterman-Reingold layout algorithm [105]. For comparison, we also illustrated the regular minimum spanning tree (MST) of the graphs (Figure 5.25f & 5.26c), from which it is a lot more difficult to visually deduce the ground-truth underlying topologies. Furthermore, it can not be deduced from the visualization in Figure 5.25f that two groups of cells are actually connected through the incorrect region (Figure 5.25c).

As our method is specifically developed for topological data analysis of graphs, it is important that the ground-truth underlying topology of our point cloud data corresponds to the underlying topology of our graph used to represent this data. This is generally a very difficult task, especially for high-dimensional data sets that commonly suffer from the curse of dimensionality. E.g., the Euclidean distance measure and the concept of closest neighbors become much less meaningful in high-dimensional spaces [106]. This may lead to a low quality representation of the data's underlying topology through a kNN graph. Though our method was able to infer the topology of the underlying cell trajectory network in our previous examples, we do note that the stated observation already applies to the constructed proximity graphs. Their quality for representing the underlying topology is affected by interconnections between different parts of branches (Figure 5.25d) or 'hubs' that connect to many other points (Figure 5.26a). The latter is a typical problem occurring in kNN graphs constructed from high-dimensional data [107].

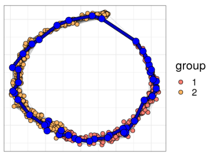
We continue with our third cell trajectory data set, which is an extreme example of how the curse of dimensionality may affect our results. Figure 5.27a visualizes



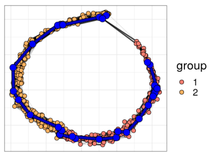
(a) A linear backbone (blue) mined through FacilitySteiner.



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

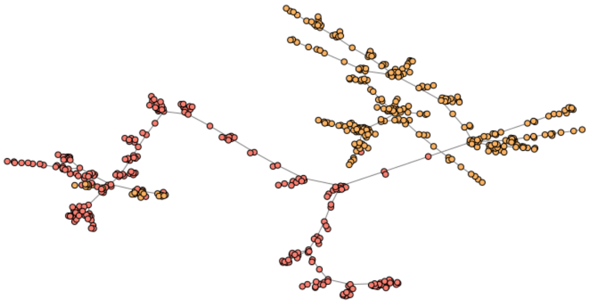


(c) A linear backbone (blue) mined through CLOFinMSF.

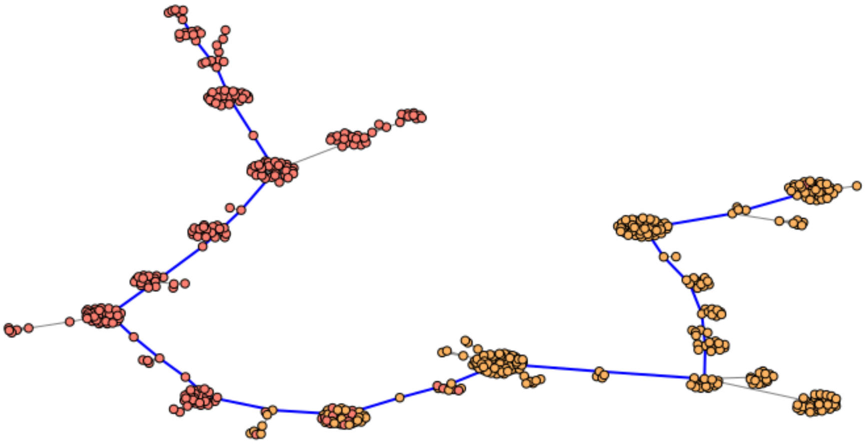


(d) A linear backbone (blue) mined from the BC-pine.

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

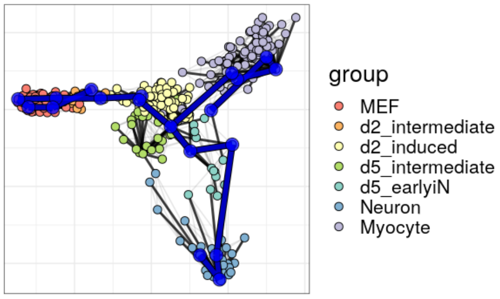


(f) Visualizing the MST of G using the Fruchterman-Reingold layout algorithm.

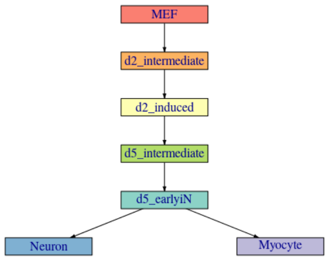


(g) Visualizing the BC-pine of G using the Fruchterman-Reingold layout algorithm, with the edges of the found linear backbone in blue.

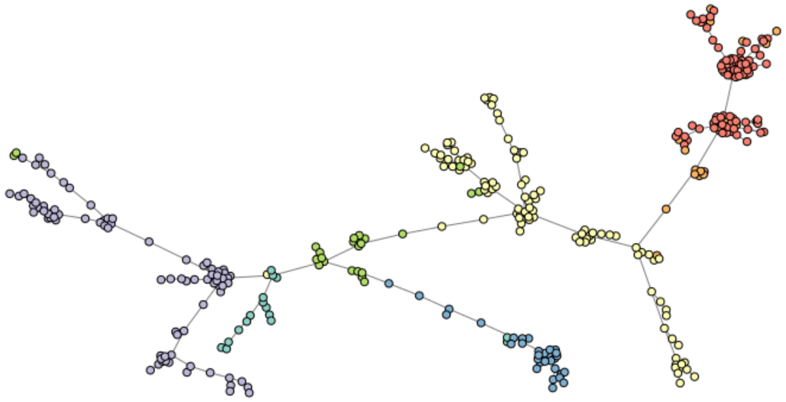
Figure 5.25: The BC-pine allows one to both visualize and infer the underlying topology from a synthetic high-dimensional gene expression data set through a 10NN graph G.



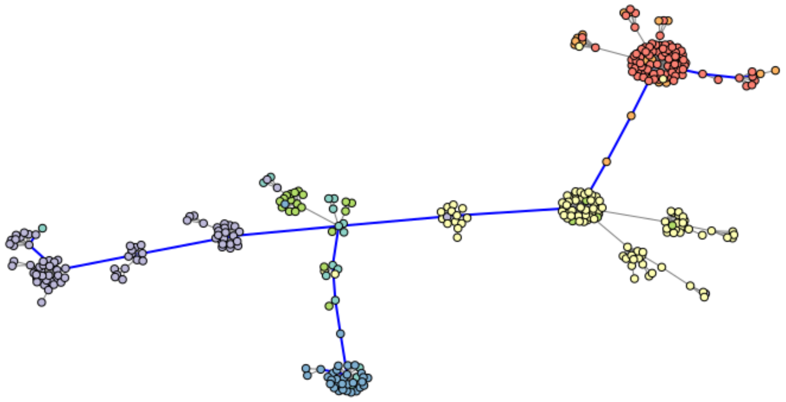
(a) A bifurcating backbone (blue) is mined from the BC-pine (black edges) constructed from a 5NN graph G (grey edges) of real gene expression data X.



(b) The ground-truth underlying topology of X. Layout provided through a tree layout algorithm.

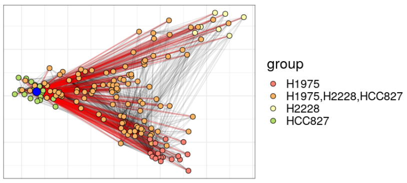


(c) Visualizing the MST of G using the Fruchterman-Reingold layout algorithm.

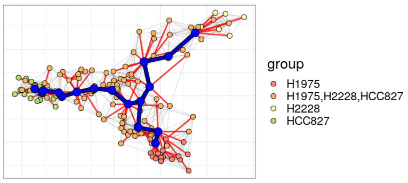


(d) Visualizing the BC-pine of G using the Fruchterman-Reingold layout algorithm, with the edges of the found bifurcating backbone in blue.

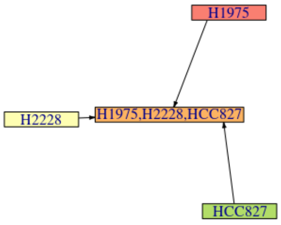
Figure 5.26: The BC-pine allows one to both visualize and infer the underlying topology from a real high-dimensional gene expression data set through a 5NN graph G.



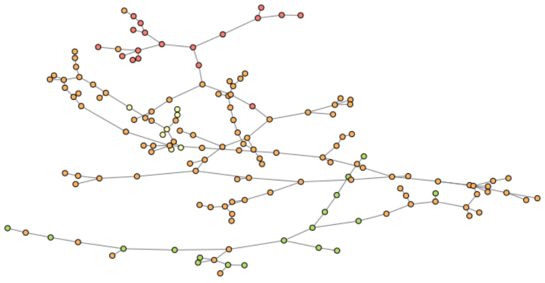
(a) A single node backbone (blue) is mined from the BC-pine (red edges) constructed from a 10NN graph G (grey edges) of real gene expression data X.



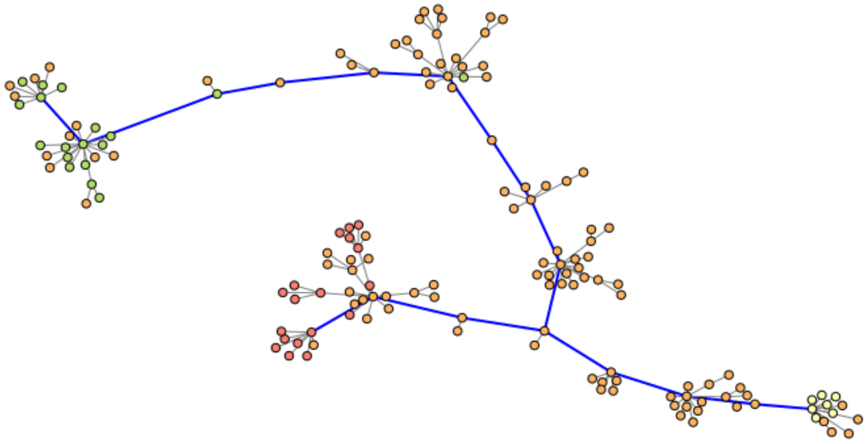
(b) A bifurcating backbone (blue) is mined from the BC-pine (red edges) constructed from a 10NN graph $\tilde{G}$ (grey edges) of a 3-dimensional embedding of X into diffusion map coordinates.



(c) The ground-truth underlying topology of X.



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



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

Figure 5.27: The BC-pine allows one to both visualize and infer the underlying topology from a real high-dimensional gene expression data set through a 10NN graph $\tilde{G}$.

the point cloud data set X using diffusion map coordinates, as well as the BC-pine of a 10NN graph G constructed from X using Euclidean distances in the original high-dimensional space. The inferred backbone B, also shown in Figure 5.27a, is a single node. This is because a data point v is the closest neighbor of 147 out of the 153 other data points. In the 10NN graph constructed from the high-dimensional data, every one of these 153 data points is connected to v, and the obtained BC-pine is a star graph as in Figure 5.5a. Making use of Theorem 5.4.8, pruning discards all of these 153 nodes, and what remains is the single node v.

Clearly, Figure 5.27a shows that the underlying topology of G is not a truthful representation of the underlying ground-truth topology of X. However, as commonly used in cell trajectory inference tools [9], a dimensionality reduction may serve as a first step for reducing the amount of noise and improving the quality of our representation.

Figure 5.27b visualizes the point cloud data set X using the same (first two) diffusion map coordinates, but this time the BC-pine of a 10NN graph $\tilde{G}$ constructed from the first three diffusion coordinates of this embedding. The underlying topology of $\tilde{G}$ is now a much better reflection of the underlying ground-truth topology of X (Figure 5.27c), and is successfully mined through solving CLOF in the BC-pine. Again, comparing Figures 5.27d & 5.27e, we note that the BC-pine serves as a tool for graph visualization as well.

5.7.4 Quantitative Analysis of the Results

As to evaluate the performance of our method, ideally one would have access to a ground-truth underlying graph-structured topology. However, not only is it difficult to assess whether there exists a homeomorphic mapping from one (geometric realization of a) graph to another [108], more often than not we do not have access to a ground-truth model (Section 3.4).

Nevertheless, in Section 5.7.4.1 we will introduce general metrics allowing us to support and interpret our obtained results, which we will summarize in Section 5.7.4.2.

Furthermore, we will conduct a separate large scale cell trajectory inference experiment in Section 5.8. The knowledge of ground-truth topologies as well as the recent development of quantitative metrics for comparing different cell trajectory inference tools [9], allows us to objectively measure the performance of our method for this purpose.

5.7.4.1 Introducing General Quantitative Metrics

We will consider a variety of metrics that allow us to meaningfully interpret how well our inferred forest-structured backbone B models the underlying topology of a graph G. All these metrics will have an intuitive meaning, and are mostly inspired by easily interpretable concepts from simple linear regression.

Model size The relative size of B compared to G should be small. The reason for this is that we want to 'explain' the entire graph G through only a few landmark nodes. Hence, one may measure $n\%$, the percentage of nodes from G still included in the resulting backbone. Similarly, one may considering $m\%$, the percentage of nodes from G still included in the resulting backbone. However, in case of forest-structured backbones, there is an explicit relation between the original size number of nodes in G, $n\%$, and $m\%$, so that this latter quantity is redundant.

Goodness of fit Although we prefer a simple model, we still want to fit our data sufficiently well (compare Figure 3.3 to Figure 5.28a). Therefore, we may consider the metric

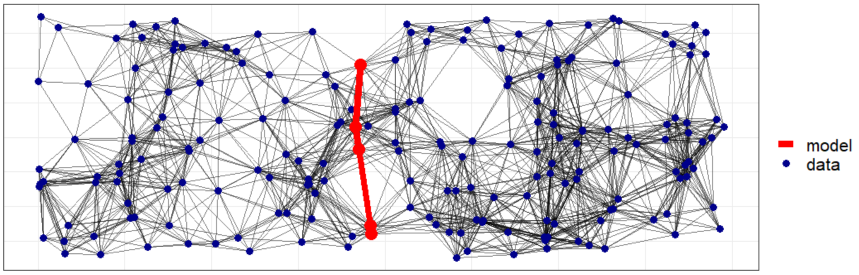
$$R(B) := 1 - \frac{\sum_{v\in V(G)} d_G(v, B)}{\sum_{v\in V(G)} d_G(v, C_G)},$$

where $d_G(v, B)$ denotes the distance of v to its closest node in B, and $d_G(v, C_G)$ denotes the distance of v to its closest node in the *center* C_G of G, defined as

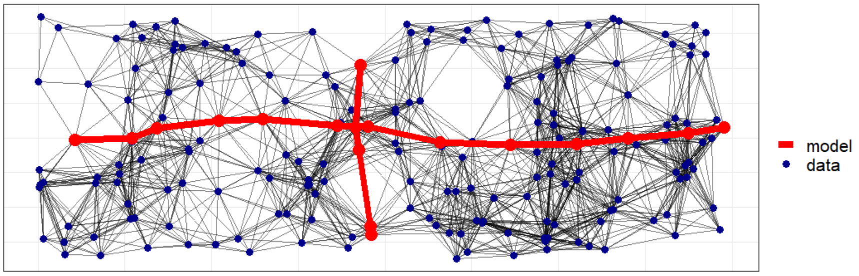
$$C_G := \left\{ u \in V(G) : (\forall v \in [u]_G) \left(\max_{w\in[u]_G} d_G(u, w) \leq \max_{w\in[u]_G} d_G(v, w) \right) \right\},$$

i.e., the set of nodes u with minimum *eccentricity* in their respective connected component $[u]_G$ of G. Inspired by the *coefficient of determination* in linear regression, we consider this metric to be a measure for how much of the 'variance' in our graph G is explained by the model B. However, we choose to not consider squared distance as in the usual definition of this coefficient, to lessen the effect of outliers. As we do not assume G to reside in any vector space, there is no definition of an 'average' node. Hence, our notion of 'mean' is fulfilled by the 'center' in graphs.

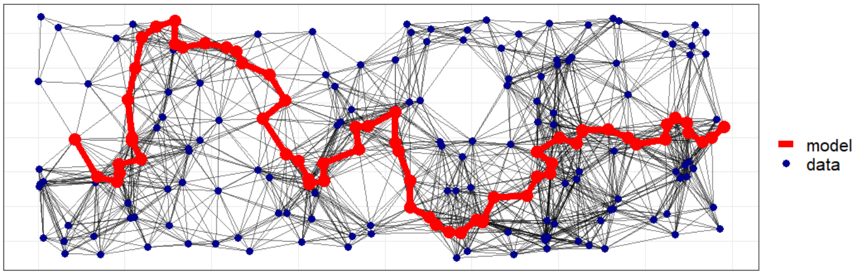
Topological complexity The linear backbone B in Figure 3.3 fits the graph G well. In this case, it captures the ground-truth topology of G, which was constructed from a noisy point cloud X sampled from a line segment. Apart from a single point, it is the 'simplest' graph-structured topological model that may occur. It has a straightforward geometric realization without any singularities, being the line segment $[0, 1]$. Clearly—and as our good friend Occam would agree—topological less complex models like these are preferred over topologically more complex models. E.g., Figure 5.28b shows another possible backbone in G, which approximates G nearly equally well (and smooth, see also the next metric), but clearly the extra branches are redundant. Various terms quantifying the topological



(a) A graph G (nodes in blue) and a subgraph B_2 (red) of G which models G in the 'wrong direction'.



(b) A graph G (nodes in blue) and a subgraph B_3 (red) of G which incorrectly represents the underlying topology of G.



(c) A graph G (nodes in blue) and a subgraph B_1 (red) of G that 'wiggles' through the entire graph G.

Figure 5.28: Various examples of 'bad' backbones.

complexity of the backbone, such as its number of leaves, cycles, and/or multifurcations, can be considered. We will consider the number of leaves, which can be directly controlled through CLOF.

Smoothness We want our backbone B to pass 'smoothly' through our graph, instead of 'wiggling' through many nodes. The intuition behind this is similar to preventing overfitting in linear regression (compare Figure 3.3 to Figure 5.28c). To this end, we compute the *projection G_B of G on B*, by connecting each node of G

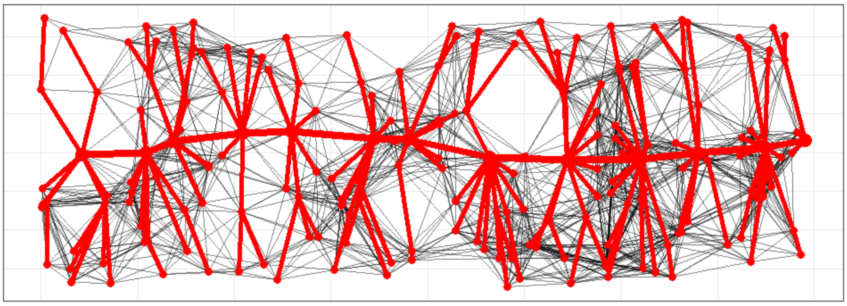


Figure 5.29: The projection G_B (red) of the graph G on the backbone B, displayed in Figure 3.3.

through its shortest path to B. Figure 5.29 illustrates as an example the projection G_B of the graph G on its backbone B shown in Figure 5.29. We then compute the ratio $\sigma := \frac{d_G(u,v)}{d_{G_B}(u,v)}$, where u and v are the most distant nodes in G. Hence, $\sigma(B)$ denotes how well the distance between the two furthest points in G is preserved through B. Note that trivially $\sigma(B) = 1$ if B was obtained through FarthestPoint. Furthermore, σ is 'penalized', i.e., further from 1, when centering our backbone and preventing it from passing through outliers. However, this is exactly what we want. E.g., Figure 5.30 shows a backbone B that is the (not necessarily longest) shortest path between two points. We see that instead of being centered in the main core of the graph, it passes through outliers to reach one destination from another.

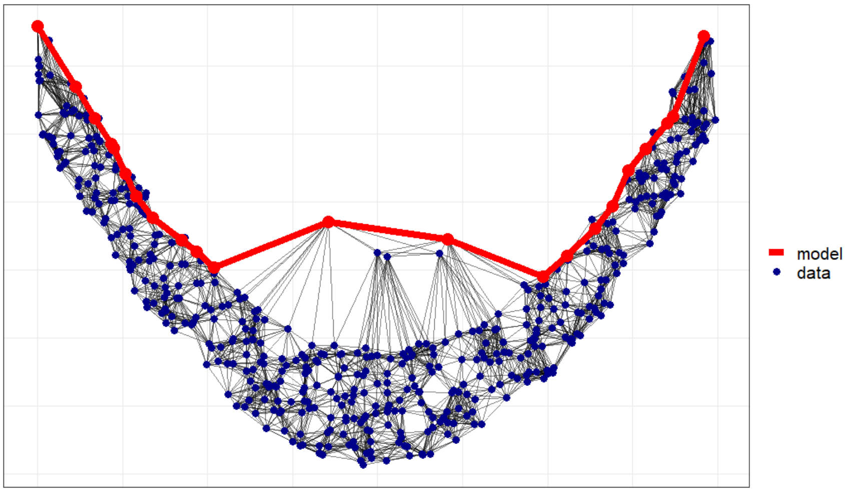


Figure 5.30: A graph H (nodes in blue) and a subgraph B_4 (red) of H which passes through outliers instead of being centered in H.

Commute time preservation A good backbone B should capture the geometrical properties, and hence, preserve the metric of the original graph G well. Unfortunately, this is one the most difficult qualities of a forest-structured backbone B to assess. Missing cycles may result in failing to preserve the original metric, whereas in the case of curvature and outliers, we actually wish to not preserve the original metric as close as possible. To this end, we consider the correlation cor_{act} between the *average commute times* of the original graph G and the projection of G on G_B [109]. These commute times are based on a Markov-chain model of random walk through the graph, which have shown to effectively deal with outliers in graphs [110]. Note that however, this metric does not account for the fact that we cannot include cycles in a forest-structured backbone.

5.7.4.2 Quantitative Summary of our Results

Table 5.1 summarizes the settings in terms of leaves we used to obtain the backbones for the graphs discussed in Section 5.7.3 through our method. Each of our baselines also requires a number of nodes to be selected per component, for which we used the k_{comp}-row of Table 5.1.

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

Table 5.1: Input and output summary to obtain a forest-structured backbone through BCB. $\mathbf{k}_{\max}$*: upper bound on the total number of leaves for solving CLOF to increase efficiency (blank if not specified).* $\mathbf{k}_{spec}$*: the specified number of leaves to be selected once the subforest had been fully grown or up until the maximum number of leaves (blank if estimated through minimizing second-order finite differences).* $\mathbf{k}_{comp}$*: the number of resulting leaves for each connected component (0 meaning a backbone component consisting of one node).*

Table 5.2 summarizes the network sizes for each graph G ($n = |V(G)|$, $m = |E(G)|$), runtimes, and our quantitative results for the metrics introduced in Section 5.7.4.1. All these results were obtained using non-optimized R code on a machine equipped with an Intel® Core™ i7 processor at 2.6GHz and 8GB of RAM. Note that the specified runtimes in Table 5.2 for our method are those when given the (possibly infinite) upper bounds $k_{\max}$ from Table 5.1. The runtimes for the baselines are those when specified to exactly select k_{comp} nodes. We observe that our method scales well to graphs with thousands of nodes.

Our method also scales well according to the topological complexity, whether given or bounded by a number of leaves. In contrast, FacilitySteiner is significantly slower when selecting more medoids in higher order graphs, taking 8.6min for the earthquakes data. Since the placement of medoids through FacilitySteiner

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

Table 5.2: Quantitative summary of our experimental results. Our method for mining backbones in graphs through forest representations is marked in blue.

is density based, it also struggles to identify important regions that make up the topology, even when specified to select the correct number of nodes to do so (Figure 5.12a). Clear examples of this are the bifurcating real cell trajectory data sets, for which FacilitySteiner infers a linear trajectory. Furthermore, unlike FarthestPoint and the backbones constructed through CLOF, there is no straightforward way to extract the result for a lower number of selections through the output of FacilitySteiner. One needs to rerun the entire algorithm for this, making it difficult to tune the resulting topological complexity through FacilitySteiner.

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

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

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

The main networks contradicting the observations above, are the unweighted networks (HP & GoT). Here, our method actually results in the best approximation of the original graph (R equals 0.56 and 0.81, respectively), while remaining a smooth approximation (σ equals 1 twice) that well preserves the average commute times (cor_{act} equals 0.96 and 0.91, respectively). Furthermore, our method appears to provide the least smooth approximation for the synthetic cell trajectory data set at first sight ($\sigma = 0.53$). However, through Figures 5.25a-5.25d, we have shown that our method was the only method capable of identifying the true underlying topology. The longest shortest path in the original graph—used for computing σ—incorrectly connects the two underlying leaves (Figure 5.25b), which explains this result.

5.8 Backbone Inference for Cell Trajectory Inference

The results we obtained through our newly introduced method, in this section abbreviated to **BCB** (**B**oundary **C**oefficients to **B**ackbone), on the cell trajectory data sets in Section 5.7.3, lead to a new *cell trajectory inference* method.

1. start from a (high-dimensional) expression data set X;
2. use a dimensionality reduction method to reduce the (high-dimensional) noise in X;
3. construct a kNN graph G from the lower dimensional representation of X;
4. construct the BC-pine in G;
5. identify the underlying topology from the pine by means of (5.4).

Contrary to many of the existing cell trajectory inference tools [9], we do not require the data to be represented in a vector space, can infer more complex topologies than linear, bifurcating, or connected ones, and do not preprocess the data through a clustering method.

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

We will compare our new cell trajectory inference method to *Slingshot* [63], which is the top ranked method for cell trajectory inference in terms of accuracy in [9], who developed a wrapper to provide a common input and output model that allows one to compare different cell trajectory inference methods. Slingshot requires a clustering of the cells into groups to start with, builds a minimum spanning tree on these clusters, and refines the such obtained tree by means of principal curves [111]. Note that the clustering method required for being able to compare Slingshot to other cell trajectory inference methods has been provided by [9].

Each of our 333 considered data sets described in Section 5.7.1 was embedded in a 20-dimensional space using diffusion maps with Pearson correlations and standard settings in R. Consecutively, we estimated the intrinsic dimension d, $3 \leq d \leq 20$, of our data using an inference method based on the eigenmultipliers of the embedding. We implemented the same inference method as the wrapper for *Slingshot* developed by [9]. The code for this wrapper can be found on `https://github.com/dynverse`. We evaluated our method over multiple values $k \in \{5, 10, 15, 20, 25\}$ for building kNN graphs from our diffusion coordinates, using the Euclidean distance between points. The resulting proximity graph was used to construct a BC-pine, from which we mined the underlying topology using CLOF. A number of leaves l, $2 \leq l \leq 30$, was estimated for each connected component in our BC-pine, by minimizing the second-order finite differences of the function mapping the number of leaves to the corresponding vertex betweenness cost, as discussed in Section 5.5.3.

We used the four metrics suggested in [9] to quantify the quality of our inferred trajectories, which we summarize below. For full details on the exact computation of these metrics, we refer to [9].

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

All these metrics lie within $[0, 1]$. Higher values correspond to better performances. We also evaluated the computational cost of our approach in terms of runtime (in seconds) and storage (in GB). Figure 5.31 visualizes the performance for each considered metric, as well as for various choices of k(NN graphs), through cumulative distribution plots.

We first note that the overall performance of our cell trajectory inference method is stable when it comes to the choice of k. In terms of the obtained results, our method turns out to be comparable to Slingshot. We especially note an increase in performance when it comes to the correlation between geodesic distances. Furthermore, our method scales at least as well as Slingshot in terms of runtime, which

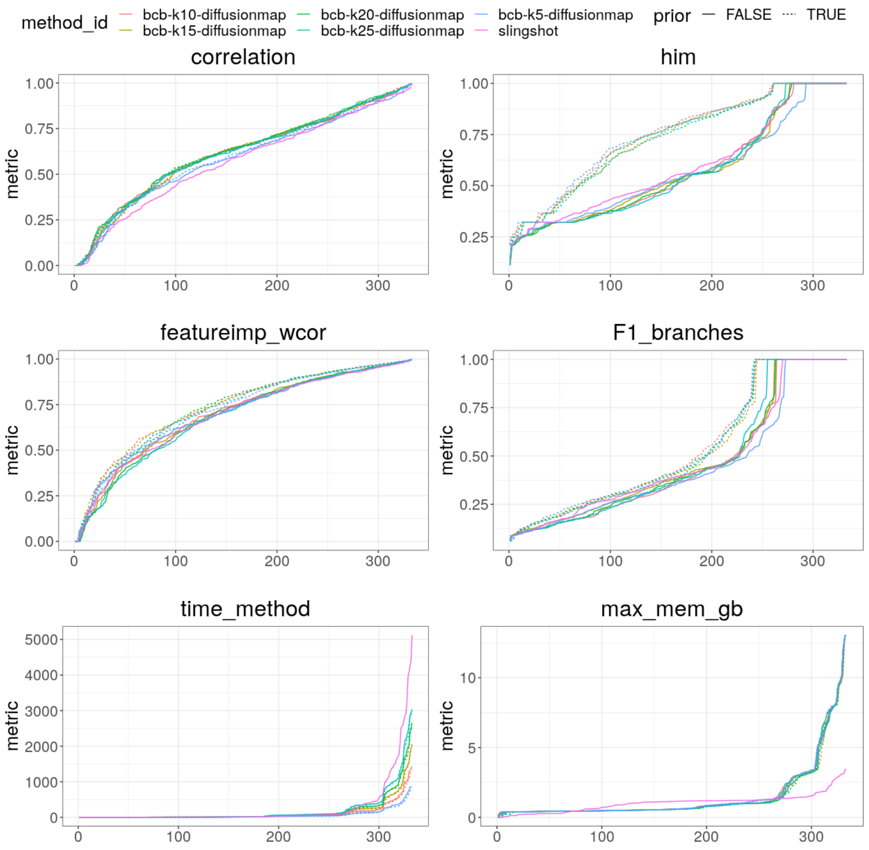


Figure 5.31: Various metrics for evaluating BCB as a cell trajectory inference tool, with and without the number of leaves as prior, sorted according to performance for each method. Our method shows to be comparable to the state-of-the-art for cell trajectory inference for all considered metrics. However, unlike Slingshot, our method allows to pass the number of leaves to the CLOF-algorithm, increasing the overall performance.

is mostly affected by the choice of k, i.e., the density of $\mathcal{H}_2(D)$ (Theorem 5.2.13). In terms of storage, our method does scale worse. However, it turns out this is due to computing and storing the entire Pearson correlation matrix for the embedding. Given the dimensionality reduction, our method scales better in terms of memory than Slingshot, through implementing 5.2.13 by means of sparse matrices (Algorithm 3), and cleverly making use of Theorem 5.4.8.

Slingshot does seem to perform better when it comes to the HIM-metric. Investigating this further, the HIM and F1 scores are mostly affected by the prediction of how many leaves are present in the underlying topology. Figure 5.32 shows

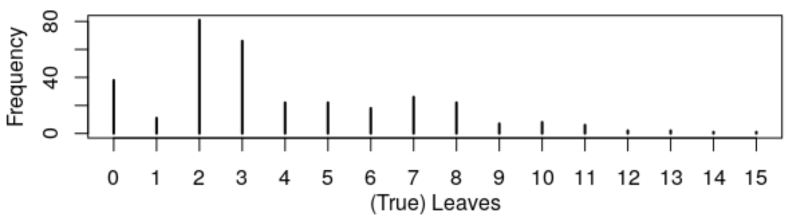


Figure 5.32: Distribution of the number of leaves across our 333 cell trajectory data sets.

the distribution of the number of leaves across our 333 cell trajectory data sets described in Section 5.7.1. Note that a trajectory with less than two leaves must include a cycle. The distribution of the (true) number of leaves across our cell trajectory data sets in Figure 5.32 shows that the cumulative plots in Figure 5.31 may be greatly affected by the performance on linear and bifurcating topologies, making up the majority of the trajectories. Especially on bifurcating topologies, Slingshot seems to provide a better estimate on the number of leaves (Figure 5.33).

We also evaluated the performance of our method when we know the true number of leaves in our network (Figure 5.31). In case of less than two leaves, e.g., which may be the case if the ground-truth topology is a cycle, we still estimated the number of leaves as above. Note that Slingshot does not allow one to input the number of leaves as prior.

We observe that the performance of our method now increases in terms of all three of the HIM metric, F1 score, and correlation between important features, most significantly for the latter two. The lack of change in the correlation between geodesic distances may be explained by this metric not being affected that much by shorter branches. These branches are often excluded from our original inferred trajectory, as our estimate based on minimizing second-order-finite differences was generally too low (Figures 5.32 & 5.33).

The results summarized in this section show how our newly introduced optimization problem constraining the number of leaves (5.4) leads to a highly effective cell trajectory inference tool. If prior knowledge on the number of leaves in the ground-truth model is available, BCB outperforms Slingshot—the until now most accurately ranked state-of-the-art method for cell trajectory inference—in terms of the metrics introduced by [9]. Without this prior knowledge, BCB is comparable to Slingshot by using our currently heuristic (elbow) estimator (Section 5.5.3). Slingshot is however unable to take the number of leaves as input. Hence, unlike Slingshot, BCB allows one to incorporate effective and independent machine learning models for estimating the number of leaves. This may eventually lead to a new best performing CTI method, even when no prior knowledge is available. Yet, in the following chapter, we will show that (and why) accurately predicting these leaves may be difficult based solely on topological information.

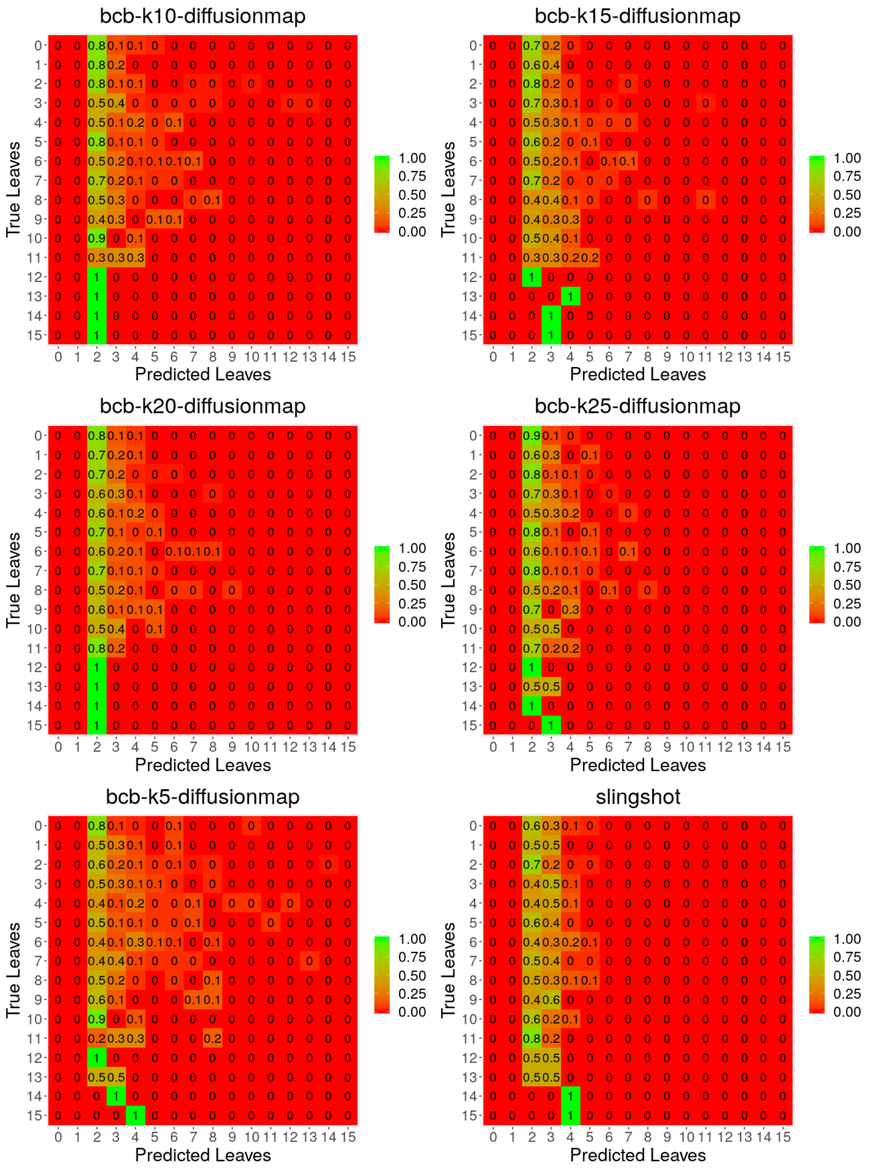


Figure 5.33: Normalized confusion matrices showing the true vs. the predicted number of leaves. Note that all methods output a trajectory with at least two leaves. In the case of 11 gene expression data sets, the number of neighbors $k = 5$ was too low to represent the connectedness of the true model through a neighborhood graph, resulting in fragmented backbones with many leaves (ranging from 17 to 104). These predictions were discarded from the corresponding confusion matrix for visualization purposes.

5.9 Discussion and Conclusion

Investigating and visualizing simplified graph-structured topologies in data is a core problem in many fields of science. We provided an effective method for backbone inference, by introducing a simple but crucial intermediate step that showed to be highly beneficial throughout this entire paper. That is, designing a forest representation from which we may efficiently, robustly, and meaningfully mine topological substructures.

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

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

There is no single best method when it comes to extracting the backbone from a network. There will be cases where our approach will not be the best one as well. Examples are when the connectedness of our graph does coincide with the connectedness of its model, or in case of metric data, when the used proximity graph is not a truthful representation of the underlying model (Figure 5.27a). Nevertheless, our results convincingly show that we provided a very promising method across a broad spectrum of realistic applications.

6 Topological Signatures through Graph Approximations

This chapter is based on the following paper.

- Robin Vandaele, Bastian Rieck, Yvan Saeys, and Tijl De Bie. *Stable Topological Signatures for Quantifying Patterns through Graph Approximations of Metric Trees*. Submitted to Pattern Recognition Letters, 2020. (Under revision)

6.1 Introduction

In the previous chapter we presented a method for inferring topological subgraph models, i.e., backbones in graphs. We showed how this led to a new CTI method, competitive with the state-of-the-art. However, we observed that CTI methods struggle to correctly infer the number of leaves, or that even high-ranked methods do not perform well on many data sets.

In this chapter, we introduce topological signatures that allow us to investigate these issues further. These signatures are obtained through 0-dimensional persistence of arbitrary graph approximations. The term ‘approximations’ is to be loosely interpreted, in the sense that we are given some graph that is meant to capture topological information of the data at hand. This can be a Rips graph, kNN graph, minimum spanning tree, or any type of neighborhood graph constructed from the data. Furthermore, this may also be the result of a (graph) model inference method such as the Mapper algorithm or one of the CTI methods we discussed in the previous chapter.

In Section 6.2, we illustrate why it may be practically more useful to guarantee that topological signatures are preserved well, rather than the (inferred) topologies. In Section 6.3, we discuss topological persistence through sublevel filtrations of graph approximations, and why an arbitrarily good preservation of both distances and functional values cannot guarantee an arbitrarily good approximation of the ground truth topological signatures. Note that this idea has been previously illustrated for Rips graphs [112], and we generalize this to any given graph. We also present and prove a new stability result for metric trees. Next, in Section 6.4 we use this to study and compare topological signatures for cell trajectory data sets, from which we will be able to conclude the presence of the issues mentioned above. Hence, in this chapter we present the use of our signatures in an exploratory data analysis setting, rather than for solely topological inference. Furthermore, we will discuss how we may regard these signatures as a method for quality control specifically in the field of CTI.

6.2 On Preserving Topology vs. Geometry

Recall our example in Section 2.3 that stated a coffee mug and a donut are topologically equivalent. Nevertheless, they are clearly geometrically distinct. The concept of an isometry between spaces is strictly stronger than the concept of a homeomorphism between spaces. One may therefore think that one should at least guarantee that we are able to correctly provide all topological information, before we can provide additional geometric information. However, this is generally untrue. An example to illustrate this is shown in Figure 6.1.

Figure 6.1 shows two different point cloud data sets. One of them is sampled from a ground truth 'H-structured' model, whereas the other is sampled from an 'X-structured' model. The middle branch of the H-structured model is so short, that without displaying the ground truth models, it becomes nearly impossible to visually distinguish the underlying models of the point clouds. Naturally, the same holds algorithmically. E.g., in [13], the authors require that the metric distortion is bounded by a function of the shortest branch length of the underlying topology to guarantee its correct reconstruction (Chapter 4).

Although the answer to both questions "are the underlying models homeomorphic" and "are the underlying models isometric" is either yes or no, the latter question admits some form of quantization that expresses to which extent the answer is true, in terms of the Gromov-Hausdorff distance. This means that even though our inferred model may be topologically completely incorrect, it can be geometrically arbitrarily close to our correct model [6].

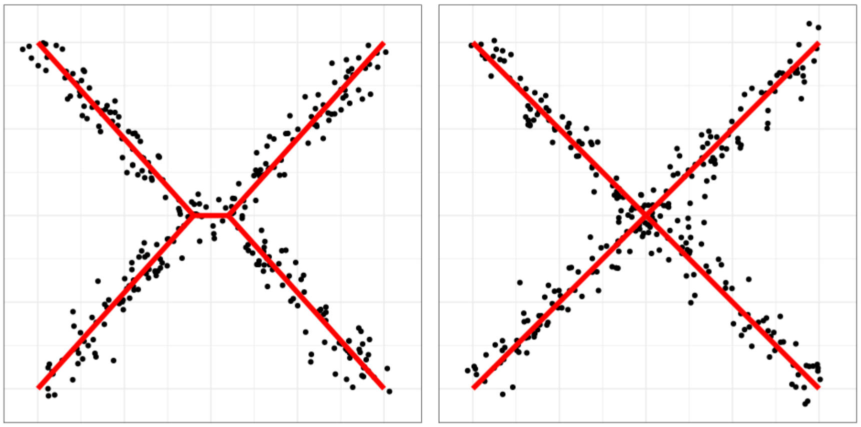


Figure 6.1: Point cloud data sets sample from (Left) an H-structured and (Right) an X-structured topology. The ground truth models are shown in red. As the middle branch of the H-structured topology is short relative to the amount of noise in the data, its underlying topology becomes difficult to distinguish from an X-structured topology. The purpose of this section is to theoretically and practically quantify that these patterns are similar.

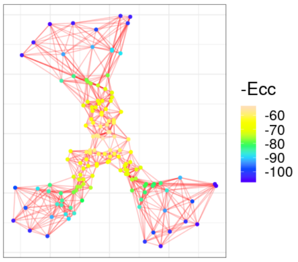
In this chapter we will not present a formal model inference method as in Chapters 4 & 5. We will study topological signatures of metric trees, obtained through graph approximations, that are exactly preserved well whenever the geometry of the model is preserved well through the approximation in terms of the Gromov-Hausdorff distance. Even if topological inference from these signatures may remain difficult, these allow us to provide a powerful quantization whether two (underlying) models, such as those in Figure 6.1, are indeed similar.

6.3 Topological Persistence through Graph Approximations

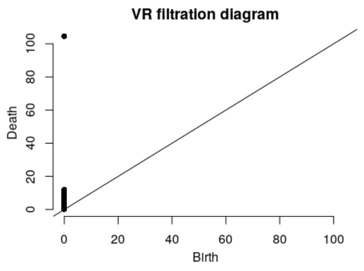
Fig. 6.2b shows that 'regular' (0-dimensional) persistent homology (Section 2.5.2.2) of the point cloud data set X shown in Fig. 6.2a misses out on capturing any topological information other than the underlying model being connected. We can however equip X with a function f that expresses how far a point is from the data center. To this end, we first constructed a 10NN graph G from X, and then computed its negative *eccentricity* function $f = -\mathcal{E}_G$, where

$$\mathcal{E}_G := \max_{x \in X} d_G(\cdot, x).$$

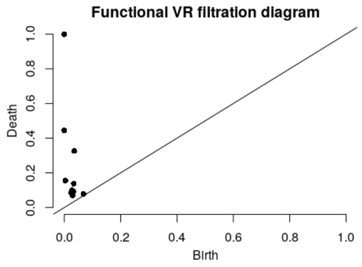
After rescaling both f and the shortest path distance metric d_G on G to [0, 1], the Rips based signature presented by [113] for the metric space (X, d_G) equipped



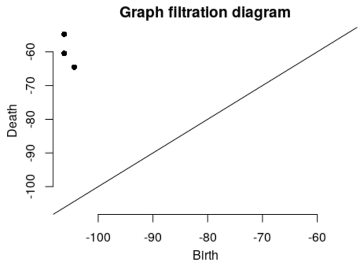
(a) A 10NN graph G of X, and its negative eccentricity function $-\mathcal{E}_G$.



(b) The Rips based signature for X (using the Euclidean distance metric).



(c) The Rips based signature for $\left(X, \frac{d_G}{\max d_G}, \mathscr{C}_G\right)$ [113].



(d) The persistence diagram obtained through G and $\mathcal{E}_G$ (Th. 6.3.3).

Figure 6.2: (0-dimensional) Rips based signatures for a point cloud data set X, and a custom defined filtration on a 10NN graph G constructed from X. The lower and upper limits of the diagram axes are defined through the first and last 'time' a simplex is added to the complex, respectively.

with resulting *normalized centrality* function

$$\mathscr{C}_G := \frac{\mathcal{E}_G^{\max} - \mathcal{E}_G(\cdot)}{\mathcal{E}_G^{\max} - \mathcal{E}_G^{\min}},$$

which more formally equals the (0-dimensional) persistence diagram computed from the filtration

$$\left(\mathcal{VR}_t^0(\bar{\mathcal{F}}_t(\mathscr{C}_G))\right)_{t\in\mathbb{R}},$$

(both the Rips graph as well as the subset of data points on which is constructed are indexed by the same time parameter t) now captures some additional structural information. The three 'leaves' present in the topology underlying X correspond to the three most elevated points in the diagram (Fig. 6.2c). However, the components representing these leaves merge quickly before reaching the center of bifurcation, due to the addition of higher weight edges at later times t. In contrast to this, (0-dimensional) persistent homology of the sublevel filtration

$$\left(\bar{\mathcal{F}}_t(\mathscr{C}_G) := G[\{v \in V(G) : \mathscr{C}_G(v) \leq t\}]\right)_{t\in\mathbb{R}}$$

easily identifies the presence of three leaves. Here, $G[U]$ denotes the subgraph of G induced by the set of nodes $U \subseteq V(G)$.

The purpose of this section is to provide a more formal theoretical foundation for last this type of persistence through such graph approximations. In Section 6.3.1, we will illustrate the concept of stability through graph approximations, and discuss the main obstacles for introducing an immediate stability result. In Section 6.3.2, we prove a new stability result for metric trees.

6.3.1 Stability through Graph Approximations

In this section, we introduce our first theorem, leading to our novel stability result in Section 6.3.2. We also discuss the necessity to split this result into two main parts (Th. 6.3.1 & 6.3.3).

The following theorem states that for any correspondence C between the points in a metric space (X, d_X) and nodes in a graph G, and functions $f : X \rightarrow \mathbb{R}$, $g : V(G) \rightarrow \mathbb{R}$, one may bound the bottleneck distance between the diagrams for f and g by a value $m = \max\{a, b\}$, measuring how well f and g preserve the connectivity in their respective sublevel filtrations under C.

Theorem 6.3.1. *Let (X, d_X) be a connected metric space, G a graph, $f : X \rightarrow \mathbb{R}$ a tame function, and $g : V(G) \rightarrow \mathbb{R}$. Let $a, b > 0$, and suppose $C \subseteq X \times V(G)$ is a correspondence with the following properties:*

- *if $x \sim y$ in $\{z \in X : f(z) \leq t\}$ and $(x, u), (y, v) \in C$, then $u \sim v$ in $G[w \in V(G) : g(w) \leq t + a]$,*
- *if $u \sim v$ in $G[w \in V(G) : g(w) \leq t]$ and $(x, u), (y, v) \in C$, then $u \sim v$ in $\{z \in X : f(z) \leq t + b\}$,*

where $\cdot \sim \cdot$ denotes that two points are connected in their respective (not necessarily topological) space, and $G[U]$ denotes the subgraph of G induced by the nodes $U \subseteq V(G)$. Then

$$d_b\left(\mathrm{Dgm}_0\left(\bar{\mathcal{F}}(f)\right), \mathrm{Dgm}_0\left(\bar{\mathcal{F}}(g)\right)\right) \leq \max\{a, b\}.$$

Proof. Our proof will use the concept of merge tree (Section 2.7). Note that the merge tree T_f is immediately defined. However, the nodes in G generally do not compose a connected topological space (e.g., for the metric d_G). We therefore take $|G|$ to be any geometric realization of G, defined through a set of functions $\psi : V(G) \rightarrow \mathbb{R}^d$, and a smooth curve $\varphi_e : [a_e, b_e] \rightarrow \mathbb{R}^d$ for each edge $e \in E(G)$, and extend g to $|G|$ through linear interpolation, i.e.,

$$|g| := |G| \rightarrow \mathbb{R} : y = \varphi_e(t) \mapsto \frac{b_e - t}{b_e - a_e} g\left(\psi^{-1}(a_e)\right) + \frac{t - a_e}{b_e - a_e} g\left(\psi^{-1}(b_e)\right).$$

For the rest of the proof, we will identify nodes $v \in V(G)$ with their images $\psi(v) \in |G|$. It holds that $\mathrm{Dgm}_0\left(\bar{\mathcal{F}}(g)\right) = \mathrm{Dgm}_0\left(\bar{\mathcal{F}}(|g|)\right)$ [36, 114]. Now let T_f and $T_{|g|}$ be the merge trees of f and $|g|$, respectively. Note that their elements (points) are equivalent classes. Furthermore, w.l.o.g. we may assume that the topology on a merge tree T_h is induced by the metric

$$\begin{aligned} &d_{T_h}([(x,t)]_{T_h}, [(x',t')]_{T_h}) \\ &\quad = 2\min\left\{\tilde{t} \geq \max\{t,t'\} : \left[(x,\tilde{t})\right]_{T_h} = \left[(x',\tilde{t})\right]_{T_h}\right\} - t - t'. \end{aligned}$$

Let $\mu := \max\{a, b\}$, and consider the mapping

$$\alpha^\mu : T_f \to T_{|g|} : [(x,t)]_{T_f} \mapsto [(y, t+\mu)]_{T_{|g|}} ,$$

where y is any node of G such that $(x, y) \in C$. Suppose $[(x,t)]_{T_f} = [(x',t')]_{T_f}$ for some $x, x' \in X$ and $t, t' \in \mathbb{R}$. As in Section 2.7, it follows that $t = t'$, with (by definition) $t \geq \max\{f(x), f(x')\}$, and (x,t) and (x',t) are connected in $\overline{f}^{-1}(t)$. Then necessarily x and x' are connected in $\bar{\mathcal{F}}_t(f)$. To see this, observe that if (x,t) and (x',t) are connected trough the path $P \times \{t\}$ in $\overline{f}^{-1}(t)$, with $P \subseteq X$, then P connects x and x' in $\bar{\mathcal{F}}_t(f)$. If now $y, y' \in V(G)$ are such that $(x,y), (x',y') \in C$, then by assumption y and y' are connected in $G[w \in V(G) : g(w) \leq t + \mu]$, and hence, in $\bar{\mathcal{F}}_{t+\mu}(|g|)$. This shows that $[(y, t+\mu)]_{T_{|g|}} = [(y', t+\mu)]_{T_{|g|}}$, so that α^μ is well-defined. Now take $[(x,t)]_{T_f}, [(x',t')]_{T_f} \in T_f$, and suppose that $(x,y), (x',y') \in C$. By our assumption, it holds that

$$\left[(x,\tilde{t})\right]_{T_f} = \left[(x',\tilde{t})\right]_{T_f} \implies \left[(y,\tilde{t}+\mu)\right]_{T_{|g|}} = \left[(y',\tilde{t}+\mu)\right]_{T_{|g|}} .$$

Hence, we find that

$$\begin{aligned} &d_{T_{|g|}}\left(\alpha^\mu([(x,t)]_{T_f}), \alpha^\mu([(x',t')]_{T_f})\right) \\ &\quad = d_{T_{|g|}}\left([(y,t+\mu)]_{T_{|g|}}, [(y',t'+\mu)]_{T_{|g|}}\right) \\ &\quad = 2\min\left\{\tilde{t} \geq \max\{t,t'\} + \mu : \left[(y,\tilde{t})\right]_{T_{|g|}} = \left[(y',\tilde{t})\right]_{T_{|g|}}\right\} - t - t' - 2\mu \\ &\quad \leq 2\min\left\{\tilde{t} \geq \max\{t,t'\} : \left[(x,\tilde{t})\right]_{T_f} = \left[(x',\tilde{t})\right]_{T_f}\right\} + 2\mu - t - t' - 2\mu \\ &\quad = d_{T_f}\left([(x,t)]_{T_f}, [(x',t')]_{T_f}\right) , \end{aligned}$$

so that α^μ is continuous.

Now consider the mapping

$$\beta^\mu : T_{|g|} \to T_f : [(\varphi_e(s), t)]_{T_{|g|}} \mapsto [(x, t+\mu)]_{T_f} ,$$

where x is any point of X such that $(x, \arg\min_{v \in e} g(v)) \in C$. Suppose that $[(\varphi_e(s), t)]_{T_{|g|}} = [(\varphi_{e'}(s'), t')]_{T_{|g|}}$ for $e, e' \in E(G)$, $s \in [a_e, b_e]$, $s' \in [a_{e'}, b_{e'}]$, and $t, t' \in \mathbb{R}$. Again, we have $t = t'$, and $\varphi_e(s)$ and $\varphi_{e'}(s')$ are connected in $\bar{\mathcal{F}}_t(|g|)$. By definition of g, $\arg\min_{u\in e} g(u)$ and $\arg\min_{v\in e'} g(v)$ are then connected in $G[w \in V(G) : g(w) \leq t]$. As before, this shows that β^μ is well-defined. Now take $[(\varphi_e(s), t)]_{T_{|g|}}, [(\varphi_{e'}(s'), t')]_{T_{|g|}} \in T_{|g|}$, and suppose that $(x, u := \arg\min_{u\in e} g(u)), (x', v := \arg\min_{v\in e'} g(v)) \in C$. Observe that $\varphi_e(s)$ and u, resp. $\varphi_{e'}(s')$ and v, are necessarily connected in $\bar{\mathcal{F}}_{\max\{t,t'\}}(|g|)$. Hence, we find that

$$\begin{aligned}
&d_{T_f}\left(\beta^\mu([(\varphi_e(s), t)]_{T_{|g|}}), \beta^\mu([(\varphi_{e'}(s'), t')]_{T_{|g|}})\right)\\
&\quad= d_{T_f}\left([(x, t+\mu)]_{T_f}, [(x', t'+\mu)]_{T_f}\right)\\
&\quad= 2\min\left\{\tilde{t} \geq \max\{t,t'\} + \mu : \left[(x,\tilde{t})\right]_{T_f} = \left[(x',\tilde{t})\right]_{T_f}\right\} - t - t' - 2\mu\\
&\quad\leq 2\min\left\{\tilde{t} \geq \max\{t,t'\} : \left[(u,\tilde{t})\right]_{T_{|g|}} = \left[(v,\tilde{t})\right]_{T_{|g|}}\right\} + 2\mu - t - t' - 2\mu\\
&\quad= 2\min\left\{\tilde{t} \geq \max\{t,t'\} : \left[(\varphi_e(s),\tilde{t})\right]_{T_{|g|}} = \left[(\varphi_{e'}(s'),\tilde{t})\right]_{T_{|g|}}\right\} - t - t'\\
&\quad= d_{T_{|g|}}\left([(\varphi_e(s), t)]_{T_{|g|}}, [(\varphi_{e'}(s'), t')]_{T_{|g|}}\right)\ ,
\end{aligned}$$

so that β^μ is also continuous.

Now take any $[(x,t)]_{T_f} \in T_f$. Clearly, it holds that

$$\widehat{|g|}(\alpha^\mu([(x,t)]_{T_f})) = t + \mu = \hat{f}([(x,t)]_{T_f}) + \mu\,.$$

Furthermore, for $(x,y) \in C$, we have

$$\begin{aligned}
\beta^\mu\left(\alpha^\mu\left([(x,t)]_{T_f}\right)\right) &= \beta^\mu\left([(y, t+\mu)]_{T_{|g|}}\right) = [(x, t+2\mu)]_{T_f}\\
&= \iota^{2\mu}_{T_f}\left([(x,t)]_{T_f}\right)\ .
\end{aligned}$$

Conversely, take $[(\varphi_e(s), t)]_{T_{|g|}} \in T_{|g|}$ with $(x, \arg\min_{v\in e} g(v)) \in C$. We have

$$\begin{aligned}
\hat{f}(\beta^\mu([(\varphi_e(s), t)]_{T_{|g|}})) &= t + \mu\\
&= \widehat{|g|}([(\varphi_e(s), t)]_{T_{|g|}}) + \mu\,,
\end{aligned}$$

and,

$$\begin{aligned}
\alpha^\mu\left(\beta^\mu\left([(\varphi_e(s), t)]_{T_{|g|}}\right)\right) &= \alpha^\mu\left([(x, t+\mu)]_{T_f}\right)\\
&= \left[\left(\arg\min_{v\in e} g(v), t + 2\mu\right)\right]_{T_{|g|}}\ .
\end{aligned}$$

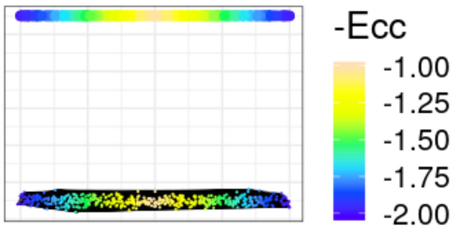
Now since $\varphi_e(s)$ and $\arg\min_{v\in e} g(v)$ are necessarily connected in $\bar{\mathcal{F}}_t(|g|)$, and hence, in $\bar{\mathcal{F}}_{t+2\mu}(|g|)$, it holds that

$$\left[\left(\arg\min_{v\in e} g(v), t+2\mu\right)\right]_{T_{|g|}} = [(\varphi_e(s), t+2\mu)]_{T_{|g|}} = \iota^{2\mu}_{T_{|g|}}\left([(\varphi_e(s), t)]_{T_{|g|}}\right)\ .$$

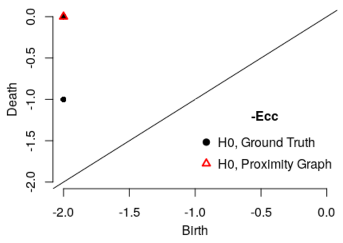
It follows that α^μ and β^μ define continuous μ-compatible maps. The result now follows from Theorem 2.7.6. □

Th. 6.3.1 cannot yet be interpreted as a stability result. We must still express how the distance between the diagrams depends on the closeness of (X, d_X) and G. However, even if (X, d_X) and G are arbitrarily close in the sense of an ϵ-correspondence C, and $f : X \to \mathbb{R}$ and $g : V(G) \to \mathbb{R}$ are arbitrarily well-preserved under this correspondence, there is generally no guarantee that the diagrams are close as well. This is illustrated by two example models and their graph approximations in Fig. 6.3.

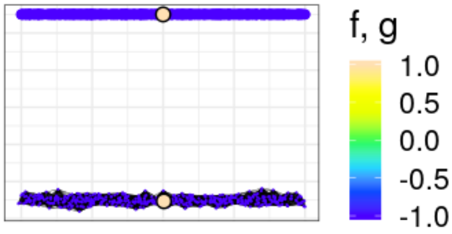
In the first example (Fig. 6.3a), we constructed the fully connected graph G on a translated sample Y of a continuous linear-structured metric space (X, d_X). Due to the absence of curvature, the metric space $(V(G), d_G)$ well-approximates (X, d_X) in the sense of an ϵ-correspondence (we omit an actual value of ϵ as we believe the concept is clear). Since G is fully connected, one connected component will be born in the filtration, and it will never die. This is illustrated by the persistence diagram in Fig. 6.3b, where we defined the filtration through the negative eccentricity function of G. Both for the ground truth model, as well as for G, the eccentricity function provides a smooth transition from the (underlying) leaves towards the center. However, the sublevel filtration for (X, d_X) will start at two



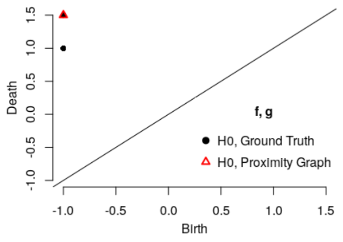
(a) The negative eccentricity for the ground truth (top) and graph approximation (bottom). The graph connects every pair of nodes.



(b) Persistent homology for the sublevel filtrations of the negative eccentricity functions.



(c) Custom defined functions f and g for the ground truth (top) and Rips graph approximation (bottom), respectively.



(d) Persistent homology for the sublevel filtrations of the custom defined functions.

Figure 6.3: In terms of Th. 6.3.1, these examples show that a (ϵ-)correspondence can preserve the metrics and function values of f and g arbitrarily well (in terms of ϵ), while simultaneously, $\max\{a, b\}$ can be arbitrarily high.

connected components, that only merge at the center of X.

The second example (Fig. 6.3c) illustrates a 'finer' approximation of (X, d_X) through the Rips graph $\mathcal{R}_{0.1}(Y)$ constructed from Y. We now defined a function f (resp. g) on X (resp. Y) that values -1 at every single point, apart from one point near the center where it values 1. Again, the filtration for $\mathcal{R}_{0.1}(Y)$ starts with one connected component (including all but one point), that never dies. The filtration for the ground truth model starts off with two connected components that merge only at the center as before.

The takeaway of the examples above, is that to ensure stability, we need two things. First, we need to formalize how well our graph G approximates the topology of the underlying space, both through the concept of ϵ-correspondences, as well as through a distance measure between nodes connected through an edge. Given a weighting function $\omega : E(G) \to \mathbb{R}^+$, we will use the maximum weight $\omega_{\max} := \max_{e\in E(G)} \omega(e)$ for this purpose. In practice, $\omega_{\max}$ will be low if the data is sufficiently densely sampled and G is a neighborhood graph. Second, the functions used to define the filtration must be such that if ϵ and $w_{\max}$ are small, so are the corresponding constants a and b in Theorem 6.3.1. Inspired by [112, Lemma 3.3], we will consider *Lipschitz* functions.

Definition 6.3.2. *Let (X, d) be a metric space. A scalar function $f : X \to \mathbb{R}$ is called c-Lipschitz if $|f(x) - f(y)| \leq cd(x, y)$ for all $x, y \in X$.*

It easily follows that Lipschitz functions are necessarily continuous.

6.3.2 Stability for Metric Trees

In this Section, we provide two closely related functions to ensure stability for tree-structured topologies through graph approximations. These will be the (negative) *eccentricity* and the *normalized centrality*, the latter of which is scale-independent. The true persistence diagrams for these functions are extremely informative for metric trees. The birth of a component will always occur through a leaf, and its death through either a multifurcation or the center of the tree (Fig. 6.2d).

Theorem 6.3.3. *Let $(\mathcal{M}, d_{\mathcal{M}})$ be a metric tree, and G a positively weighted graph such that there exists an $\epsilon_{\mathcal{M}}$-correspondence C between $(\mathcal{M}, d_{\mathcal{M}})$ and (G, d_G). Let $f : \mathcal{M} \to \mathbb{R}$, $g : V(G) \to \mathbb{R}$, and $\epsilon_f \in \mathbb{R}_{\geq 0}$ be such that for all $(x, u) \in C$, $|f(x) - g(u)| \leq \epsilon_f$, and f is c-Lipschitz. Then*

$$d_b\left(\mathrm{Dgm}_0\left(\bar{\mathcal{F}}(f)\right), \mathrm{Dgm}_0\left(\bar{\mathcal{F}}(g)\right)\right) \leq c \max\left\{\frac{\epsilon_{\mathcal{M}}}{2}, w_{\max}\right\} + c\epsilon_{\mathcal{M}} + \epsilon_f.$$

Proof. Take any $(x, u), (y, v) \in C$, let $P_{x,y} \subseteq \mathcal{M}$ denote the unique path from x to y in $\mathcal{M}$, and let $(u = p_0, p_1, \ldots, p_l = v)$ be a shortest path from u to v in G. For any $0 \leq i \leq l$, take q_i such that $(q_i, p_i) \in C$, with $q_0 = x, q_l = y$.

Suppose first that $x \sim y$ in $\{z \in \mathcal{M} : t \leq f(z)\}$. Let m_i be the closest point from q_i on $P_{x,y}$. If for any i, $d_{\mathcal{M}}(q_i, m_i) > \frac{3\epsilon_X}{2}$, then

$$\begin{aligned} d_{\mathcal{M}}(x,y) &= d_{\mathcal{M}}(x,m_i) + d_{\mathcal{M}}(m_i,y) \\ &= d_{\mathcal{M}}(x,q_i) + d_{\mathcal{M}}(q_i,y) - 2d_{\mathcal{M}}(q_i,m_i) \\ &< d_{\mathcal{M}}(x,q_i) + d_{\mathcal{M}}(q_i,y) - 3\epsilon_{\mathcal{M}} \leq d_G(u,v) - \epsilon_{\mathcal{M}} \leq d_{\mathcal{M}}(x,y), \end{aligned}$$

a contradiction. Now since necessarily $m_i \in \{z \in \mathcal{M} : t \leq f(z)\}$,

$$g(p_i) \geq f(q_i) - \epsilon_f \geq f(m_i) - cd_{\mathcal{M}}(m_i, q_i) - \epsilon_f \geq t - \frac{3c\epsilon_{\mathcal{M}}}{2} - \epsilon_f.$$

This shows that $u \sim v$ in $G\left[\left\{w \in V(G) : t - \frac{3c\epsilon_{\mathcal{M}}}{2} - \epsilon_f \leq g(w)\right\}\right]$.

Now suppose we have $x \nsim y$ in $\{z' \in \mathcal{M} : t \leq f(z')\}$. If $x = y$, then $\max\{g(u), g(v)\} < t + \epsilon_f$. as neither u nor v is included in this subgraph, $u \nsim v$ in $G\left[\{w \in V(G) : t + c(w_{\max} + \epsilon_{\mathcal{M}}) + \epsilon_f \leq g(w)\}\right]$. If $x \neq y$, take any $z \in P_{x,y}$ that minimizes $f(z)$ over $P_{x,y}$. Observe that necessarily $f(z) < t$. Now let

$$i := \max\left\{0 \leq i < l : P_{q_i, P_{x,y}} \cap P_{z,y} = \emptyset \vee z = q_i\right\},$$

where $P_{q_i,P_{x,y}} \subseteq \mathcal{M}$ denotes the unique path from q_i to (its closest point on) $P_{x,y}$ in $\mathcal{M}$. By definition of i, the path from q_i to q_{i+1} necessarily passes through z. Hence, it follows that

$$\begin{aligned} g(p_i) &\leq f(q_i) + \epsilon_f \leq f(z) + cd_{\mathcal{M}}(q_i, z) + \epsilon_f \\ &\leq f(z) + cd_{\mathcal{M}}(q_i, q_{i+1}) + \epsilon_f \\ &\leq f(z) + c(w_{\max} + \epsilon_{\mathcal{M}}) + \epsilon_f < t + c(w_{\max} + \epsilon_{\mathcal{M}}) + \epsilon_f. \end{aligned}$$

Again $u \nsim v$ in $G\left[\{w \in V(G) : t + c(w_{\max} + \epsilon_{\mathcal{M}}) + \epsilon_f \leq g(w)\}\right]$. Since continuous functions on finitely triangulable spaces are tame [40, Proposition 2.3], the result follows from Theorem 6.3.1. □

Remark 6.3.4. *The proof of Theorem 6.3.1 suggests that we can obtain even stronger comparisons by looking at the interleaving distance between the resulting merge trees, instead of the 0-dimensional persistence diagrams. Indeed, [57, Fig. 3] provides an example of two distinct merge trees for which the corresponding functions have the exact same persistence diagram. Unfortunately, computing interleaving distances between merge trees is currently a lot more challenging than computing bottleneck distances between persistence diagrams [115].*

Remark 6.3.5. *For Rips graphs $G = \mathcal{R}_{3\delta}(X)$, the bound in Theorem 6.3.3 reduces to the bound in [112, Lemma 3.3] for zeroth-order persistent homology, whenever $\frac{\epsilon_{\mathcal{M}}}{2} \leq w_{\max} \leq 3\delta$. However, our result applies to any graph, and does not require that $w_{\max}$ dominates $\frac{\epsilon_{\mathcal{M}}}{2}$. Intuitive examples where this may not be true include minimum spanning trees.*

Similar to [112, Lemma 3.3], we expect that our result can be generalized to arbitrary length spaces by bounding $\epsilon_{\mathcal{M}}$ through a function of the convexity radius *$\rho(\mathcal{M})$ of $\mathcal{M}$. The convexity radius $\rho(\mathcal{M})$ states that for any open metric ball in $\mathcal{M}$ of radius less than $\rho(\mathcal{M})$, any two points x, y in this ball are connected by a unique shortest path on $\mathcal{M}$.*

The following result can now be derived.

Corollary 6.3.6. *Let $(\mathcal{M}, d_{\mathcal{M}})$ be a metric tree, and G a positively weighted graph such that there exists an ϵ-correspondence C between $(\mathcal{M}, d_{\mathcal{M}})$ and (G, d_G). Let $\mathcal{E}_{\mathcal{M}} := \max_{x\in\mathcal{M}} d_{\mathcal{M}}(\cdot, x)$ be the* eccentricity *function, and $\mathscr{C}_{\mathcal{M}} := \frac{\mathcal{E}^{\max}_{\mathcal{M}} - \mathcal{E}_{\mathcal{M}}(\cdot)}{\mathcal{E}^{\max}_{X} - \mathcal{E}^{\min}_{\mathcal{M}}}$ be the* normalized centrality *function on $\mathcal{M}$ ($\mathcal{E}_G$ and $\mathscr{C}_G$ are defined analogously). Then*

$$d_b\left(\mathrm{Dgm}_0\left(\bar{\mathcal{F}}(-\mathcal{E}_{\mathcal{M}})\right), \mathrm{Dgm}_0\left(\bar{\mathcal{F}}(-\mathcal{E}_G)\right)\right) \leq \max\left\{\frac{\epsilon}{2}, w_{\max}\right\} + 2\epsilon,$$

and

$$d_b\left(\mathrm{Dgm}_0\left(\bar{\mathcal{F}}(\mathscr{C}_{\mathcal{M}})\right), \mathrm{Dgm}_0\left(\bar{F}(\mathscr{C}_G)\right)\right) \leq \frac{\max\left\{\frac{\epsilon}{2}, w_{\max}\right\} + 5\epsilon}{\mathrm{rad}(\mathcal{M})},$$

where the last inequality holds if $\mathscr{C}_{\mathcal{M}}$ and $\mathscr{C}_G$ are well-defined.

Proof. Take $x, y \in X$ and suppose $(x, u) \in C$. It holds that

$$\mathcal{E}_{\mathcal{M}}(x) = \max_{z\in\mathcal{M}} d_{\mathcal{M}}(x, z) \leq \max_{w\in V(G)} d_G(u, w) + \epsilon = \mathcal{E}_G(u) + \epsilon.$$

Analogously, $\mathcal{E}_G(u) \leq \mathcal{E}_{\mathcal{M}}(x) + \epsilon$, and we may take $\epsilon_{\mathcal{E}_{\mathcal{M}}} = \epsilon_{\mathcal{M}} = \epsilon$ in Theorem 6.3.3. Furthermore, with $z = \arg\max_{z'\in\mathcal{M}} d_{\mathcal{M}}(x, z')$ we have

$$\begin{aligned}\mathcal{E}_{\mathcal{M}}(x) - \mathcal{E}_{\mathcal{M}}(y) &= d_{\mathcal{M}}(x, z) - \max_{z'\in\mathcal{M}} d_{\mathcal{M}}(y, z') \leq d_{\mathcal{M}}(x, z) - d_{\mathcal{M}}(y, z)\\ &\leq d_{\mathcal{M}}(x, y).\end{aligned}$$

Analogously, $\mathcal{E}_{\mathcal{M}}(y) - \mathcal{E}_{\mathcal{M}}(x) \leq d_{\mathcal{M}}(x, y)$, so that $\mathcal{E}_{\mathcal{M}}$ is 1-Lipschitz. The stability result for $\mathcal{E}_G$ now follows from Theorem 6.3.3.

For the proof of the stability result for the normalized centrality function, we will use the intermediate diagram

$$\mathcal{D}' := \left\{\frac{\mathcal{E}^{\max}_{\mathcal{M}} + (b, d)}{\mathcal{E}^{\max}_{\mathcal{M}} - \mathcal{E}^{\min}_{\mathcal{M}}} : (b, d) \in \mathrm{Dgm}_0\left(\bar{\mathcal{F}}(-\mathcal{E}_G)\right)\right\},$$

where the addition is vectorized. Due to our previous result,

$$d_b\left(\mathrm{Dgm}_0\left(\bar{\mathcal{F}}(\mathscr{C}_{\mathcal{M}})\right), \mathcal{D}'\right) \leq \frac{\max\left\{\frac{\epsilon}{2}, w_{\max}\right\} + 2\epsilon}{\mathrm{rad}(\mathcal{M})}.$$

Now take any $u \in V(G)$. It holds that

$$\begin{aligned}\frac{\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_G(u)}{\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_{\mathcal{M}}^{\min}} - \frac{\mathcal{E}_G^{\max} - \mathcal{E}_G(u)}{\mathcal{E}_G^{\max} - \mathcal{E}_G^{\min}} &\leq \frac{\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_G(u)}{\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_{\mathcal{M}}^{\min}} - \frac{\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_G(u) - \epsilon}{\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_{\mathcal{M}}^{\min} + 2\epsilon} \\ &= \frac{2\epsilon\left(\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_G(u)\right) + \epsilon\left(\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_{\mathcal{M}}^{\min}\right)}{\left(\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_{\mathcal{M}}^{\min}\right)^2 + 2\epsilon\left(\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_{\mathcal{M}}^{\min}\right)} \\ &\leq \frac{3\epsilon\left(\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_{\mathcal{M}}^{\min} + \epsilon\right)}{\left(\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_{\mathcal{M}}^{\min}\right)^2 + 2\epsilon\left(\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_{\mathcal{M}}^{\min}\right)} \\ &= \frac{3\epsilon + \frac{3\epsilon^2}{\mathrm{rad}(\mathcal{M})}}{\mathrm{rad}(\mathcal{M}) + 2\epsilon} \leq \frac{3\epsilon}{\mathrm{rad}(\mathcal{M})}.\end{aligned}$$

We furthermore find that

$$\begin{aligned}\frac{\mathcal{E}_G^{\max} - \mathcal{E}_G(u)}{\mathcal{E}_G^{\max} - \mathcal{E}_G^{\min}} - \frac{\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_G(u)}{\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_{\mathcal{M}}^{\min}} &\leq \frac{\mathcal{E}_G^{\max} - \mathcal{E}_G(u)}{\mathcal{E}_G^{\max} - \mathcal{E}_G^{\min}} - \frac{\mathcal{E}_G^{\max} - \mathcal{E}_G(u) - \epsilon}{\mathcal{E}_G^{\max} - \mathcal{E}_G^{\min} + 2\epsilon} \\ &= \frac{2\epsilon\left(\mathcal{E}_G^{\max} - \mathcal{E}_G(u)\right) + \epsilon\left(\mathcal{E}_G^{\max} - \mathcal{E}_G^{\min}\right)}{\left(\mathcal{E}_G^{\max} - \mathcal{E}_G^{\min}\right)^2 + 2\epsilon\left(\mathcal{E}_G^{\max} - \mathcal{E}_G^{\min}\right)} \\ &\leq \frac{3\epsilon}{\mathcal{E}_G^{\max} - \mathcal{E}_G^{\min} + 2\epsilon} \\ &\leq \frac{3\epsilon}{\mathcal{E}_{\mathcal{M}}^{\max} - \mathcal{E}_{\mathcal{M}}^{\min} - 2\epsilon + 2\epsilon} = \frac{3\epsilon}{\mathrm{rad}(\mathcal{M})}.\end{aligned}$$

This shows that $d_b\left(\mathcal{D}', \mathrm{Dgm}_0\left(\bar{\mathcal{F}}(\mathscr{C}_G)\right)\right) \leq \frac{3\epsilon}{\mathrm{rad}(\mathcal{M})}$. The result follows by combining the bottleneck matchings. □

Illustrating Stability for Metric Trees through a Toy Example To illustrate how Corollary 6.3.6 can be applied in practice, we considered four tree-structured topologies embedded in $\mathbb{R}^2$, and sampled 600 observations from each of them, by sampling uniformly from each branch a number of points proportional the length of this branch. For each of these data sets, we applied a small amount of random 2-dimensional Gaussian noise, as well as a random rotation, three times. From each of these twelve resulting data sets, we constructed a Euclidean minimum spanning tree (MST) [116], and computed the normalized centrality function. The resulting functions, MSTs, as well as the ground truth models, are shown in Fig. 6.4.

The persistence diagrams obtained for the sublevel filtrations of the normalized centrality functions are shown in Figure 6.5. Note that there may be overlapping points. As can be expected, there are many points in the persistence diagrams for the MSTs near the diagonal. This is a result from the MST not including any triangles (in the graph theoretical—not the simplicial—sense). Nevertheless, we observe that the highly elevated points in all our diagrams identify important structural information of the ground truth models.

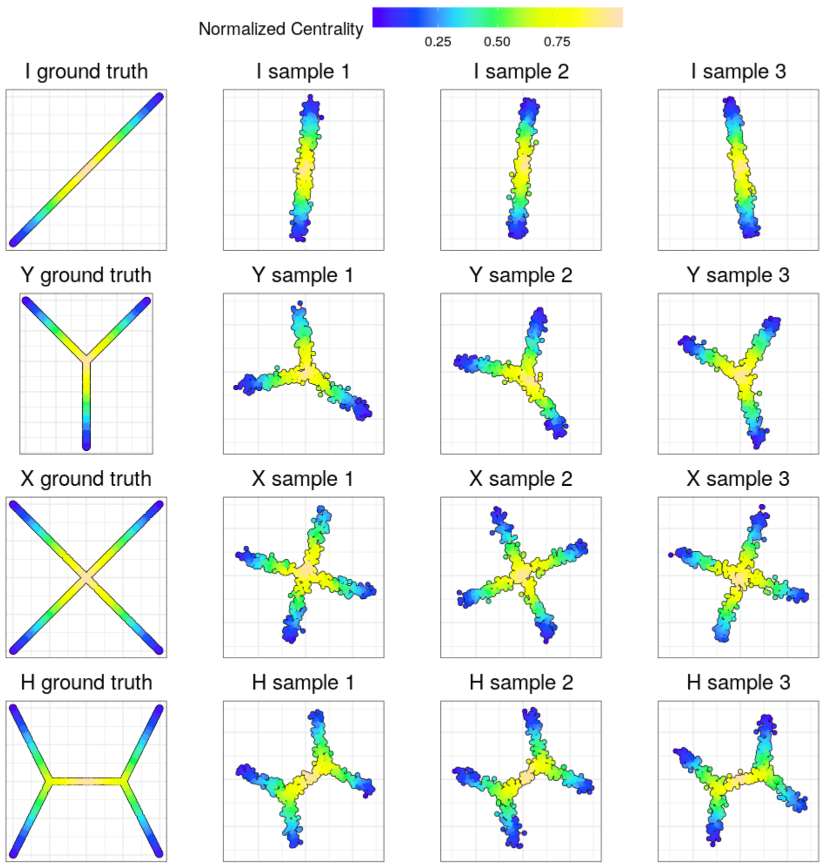


Figure 6.4: Synthetic data sampled from the metric trees in the first column. The samples and their (MST) normalized centralities are shown in columns 2-4.

Figure 6.6a visualizes the pairwise bottleneck distances between all diagrams. Figure 6.6b shows a Multi-Dimensional Scaling (MDS) plot of this distance matrix. We see that similar shapes are clustered well together. We also note that the H-structured topologies are somewhat in the middle of the other topologies. This is as expected. E.g., the longer the middle branch of the corresponding model is, the closer this pattern is to a I-pattern. The shorter this branch is, the closer it is to an X-pattern. This illustrates that topological inference may be difficult depending on the level of noise relative to the shortest branch length, as also discussed in Section 6.2 Nevertheless, in the following Section we will show that these signatures through graph approximations may serve as a tool for exploratory data analysis, rather than topological inference, in case of real world cell trajectory data.

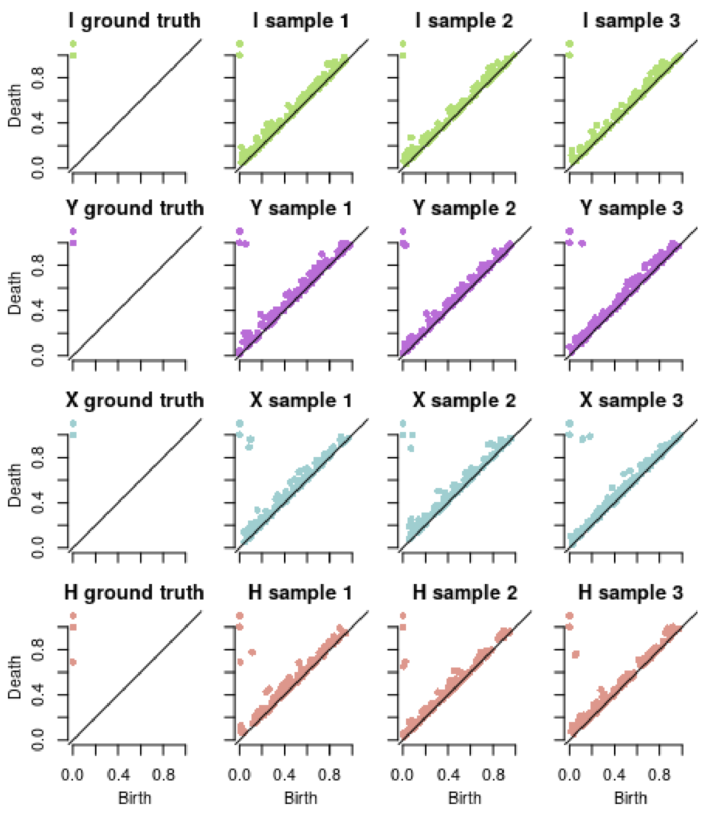
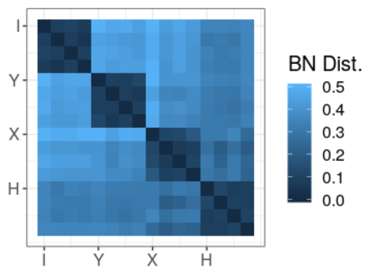
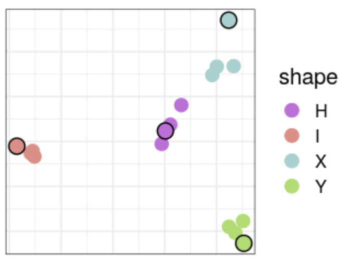


Figure 6.5: The ground truth and empirical persistence diagrams are computed using the normalized centrality to define the filtration.



(a) Pairwise bottleneck distances between all our true and experimental diagrams. The ground truths are marked by their corresponding shape.



(b) MDS plot of the pairwise bottleneck distances. The points corresponding to the ground truth models are marked by a black contour.

Figure 6.6: Visualizing the bottleneck distances between the diagrams.

6.4 Charting Cell Trajectory Data Sets through Topological Signatures

Cell trajectory inference is overall a very difficult task. Even the top ranked methods have a low performance on many data sets, and find it difficult to correctly infer important topological properties such as the number of leaves (Section 5.8). The purpose of this section is not to propose the use of our signatures (Corollary 6.3.6) as a new topological inference method for this type of data, but rather to use these to study why this problem is essentially so difficult. To this end, we proceed with an analysis similar to the one above.

We consider 131 synthetic and 57 real cell trajectory data sets with an underlying tree-structured model [65]. The number of cells ranged from 59 to 5018, and the number of genes from 373 to 23 658. A two-dimensional diffusion map embedding was computed for each data set, both for visualization purposes, as well as to reduce the effects of the *curse of dimensionality* on our neighborhood graph approximation [107]. A 10NN graph and its normalized centralities were computed from each embedding.

Fig. 6.7 visualizes all cell trajectory data sets by means of an MDS plot of the pairwise bottleneck distances we obtained through topological persistence of our 10NN graphs. We illustrate twelve 'landmark' embeddings of cell trajectory data sets, as well as their ground truth models on these embeddings, and their obtained empirical persistence diagrams in Fig. 6.8.

First, observe that all linear cell trajectories are located near a linear curve on top of the MDS plot. This means that our chosen data representation does not artificially create more leaves than truthfully present. However, many nonlinear trajectories are located near this curve as well. Near the right side of this curve, this is mainly due to branches being relatively short compared to a main linear trajectory (e.g., MDS (0, 0.6) in Figure 6.8). These trajectories are indeed theoretically close to linear according to our chosen metric. On the left side of this curve, we find the more noisy data sets, where we fail to provide a good representation. Their persistence diagrams represent more 'blob'-like patterns (Figure 6.8). Below this curve, we find the trajectories where we truthfully manage to identify additional branches. However, we note that it appears to be difficult to identify more than three leaves. This explains why cell trajectory inference methods commonly underestimate the true number of leaves (Section 5.8). Note that the 'boomerang' shape made up by all data sets in Figure 6.7, also coincides with what we theoretically expect for our chosen metric. We note a continuous transmission of blob-like patterns, towards linear patterns, towards patterns with leaves. The fact that this shape takes a turn near the right, can be theoretically explained through the definition of the bottleneck distance. As we only look at the maximal distances of a matching, the number of 'high' distances in such matching does not

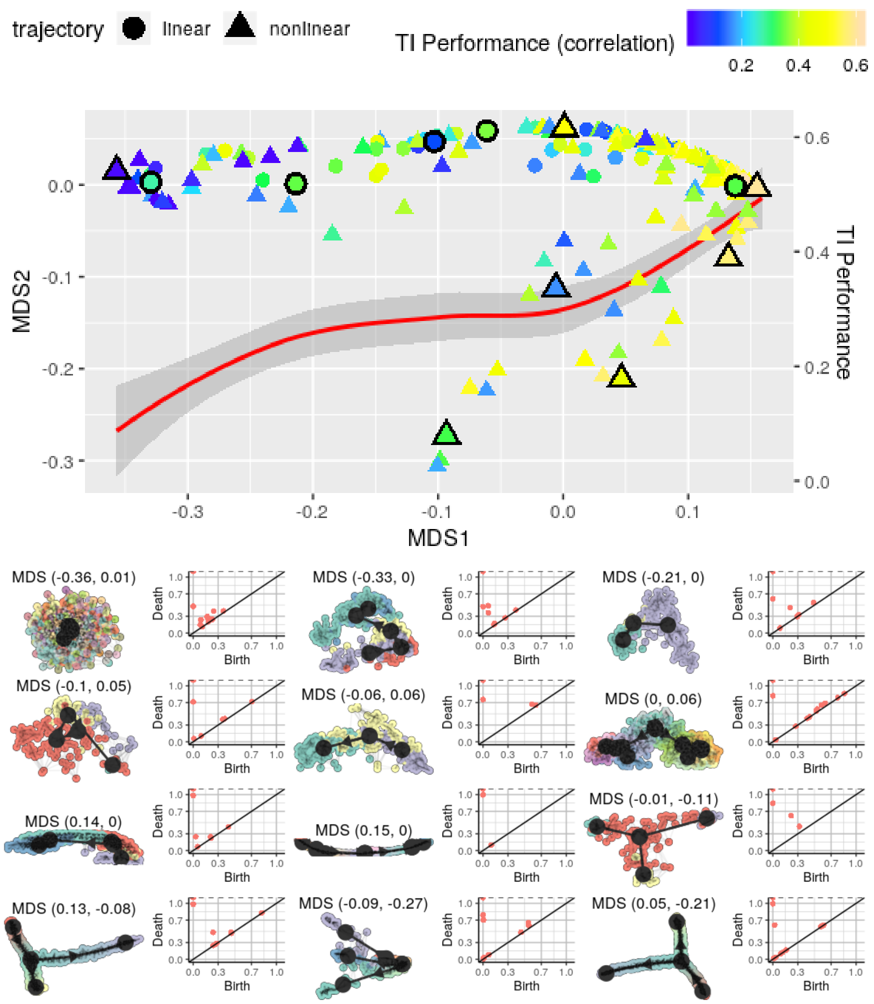


Figure 6.8: Twelve example data sets and their corresponding empirical persistence diagram. The coloring corresponds to the ground truth grouping of cells.

matter. Hence, blob-like patterns are as distant from linear patterns as they are from patterns with leaves, according to this metric.

Finally, we fitted a loess curve (standard settings in R) using the MDS1 coordinate as the independent and the average performance over 45 different cell trajectory inference methods as the dependent variable. This performance is measured through the geodesic distance preservation (correlation) metric introduced by [9] (Section 5.8). Figure 6.7 shows a positive correlation (0.58) between these variables. Note that the choice of using the MDS1 coordinate is arbitrary in general. However, this choice supports our findings that on the left side of our MDS plot, we mainly find noisy data sets. Since every cell trajectory inference method uses a different algorithm or data representation (such as the type of dimensionality reduction or neighborhood graph), this can be seen as a quality measure of the data itself, independent of our chosen data representation.

6.5 Discussion and Conclusion

We provided a novel foundation for quantifying topological patterns in metric trees through graph approximations, which led to new stability results. This is often a more useful concept than guaranteeing a correct topological reconstruction. When aiming for such reconstructions, this results in very restrictive assumptions, as well as algorithms that are sensitive to the used parameters and the amount of noise. In contrast, our results are direct, and independent of unknown properties of the underlying topology.

We verified the theoretical behavior of our topological signatures—the persistence diagrams from Corollary 6.3.6—on an experimental level. We furthermore developed novel insights into cell trajectory inference, and provided the first charting of such data sets that explains some of the difficulties this field is confronted with. We also provided a new way of quality measurement, that does not require ground truth knowledge.

Though our stability result currently only holds for metric trees, we opened up new possibilities to study which functions ensure stability by means of Theorems 6.3.1 & 6.3.3, and Remark 6.3.5. This may lead to further theoretical justification of recognizing a wider variety of patterns through graph approximations.

7 Topological Object Detection in Images

This chapter is based on the following publication.

- Robin Vandaele, Guillaume Adrien Nervo, and Olivier Gevaert. *Topological image modification for object detection and topological image processing of skin lesions.* Scientific Reports, 10(1):21061, 2020. [2]

7.1 Introduction

In this chapter, we consider an application of TDA that is rather distinct from our previous chapters. We will *extend the applicability of TDA for real world image processing and object recognition*. Recall that a 2D image can be seen as a real valued-function defined on a graph, which can be seen as a 1-dimensional abstract simplical complex (Sections 2.5.1 & 2.5.2.4). Furthermore, for the applications we consider in this chapter, we will not require including higher dimensional simplices than nodes and edges. Hence, in some sense, we will still conduct topological inference *on* graphs in this chapter. However, although mathematically our inferred objects will be subgraphs, they will represent well-defined 2D areas in the real-world, i.e., we do not conduct topological inference *of* graphs in this chapter.

Our current work builds upon the idea that persistent homology can be used to detect objects in images [21]. However, many real-world images contain outliers, as well as irrelevant objects, which complicate the use of persistent homology for this purpose. We enclose this gap by introducing ***T**opological **I**mage **M**odification* (TIM). TIM targets improving ***T**opological **I**mage **P**rocessing* (TIP)—processing an image based on the aggregation of its topological information—by *filtering out significant but irrelevant topological information.* We consider the use of TIP for enhancing the ability to identify and segment important objects on images, increasing the performance of existing models and algorithms for this purpose.

Unlike existing TDA methods for object detection or segmentation, through TIM and TIP we are able to discard significant but irrelevant objects [21, 117], which is the main purpose for which we designed this method. Furthermore, although we will consider skin lesion images for illustrating the effectiveness of our method, we do not require any specific textural assumptions restricted to this domain [117]. We also mark the relevant objects in images in a robust manner, rather than producing a parameter sensitive oversegmentation of the entire image [118, 119]. Finally, we may pass our resulting processed image to any segmentation algorithm, and do not target active contour based segmentation methods in particular [119].

Our case-study will consider skin lesion images [120, 121], ideal for illustrating the intuition behind our method. Nevertheless, TIM (and as result, TIP) generalizes to many types of real-world images through its generic assumptions, without requiring any supervision [122, 123]. Furthermore, although any existing method for segmentation or object detection (or that might be modified for this purpose) [124–129] could lead to a possibly generic method for TIP as a replacement of Algorithm 7 which we introduce in this work, we will make clear that these are unfit for this purpose in our experiments. To the best of our knowledge, neither TIM, the concept of TIP, nor the flow 'TIM $\rightarrow$ TIP $\rightarrow$ Segmentation' has been introduced or studied before.

Figure 7.1 illustrates an example of our proposed approach. The original image depicts a centered skin lesion (the relevant object of this image). The image also depicts other irrelevant objects, such as strands of hair and a part of another lesion connecting to the border of the image. Both the irrelevant and relevant objects, as well as the border of the image, are all included in the result of the *Chan-Vese* segmentation algorithm (an unsupervised segmentation algorithm for single channel images [124]) on the corresponding grayscale image. However, the same segmentation algorithm on the topological processed image provides a much better segmentation, that only includes the relevant object. This processed image was obtained from the topological information of our topologically modified image. In the following sections, we describe our approach in more detail.

We discuss the existing difficulties of TDA for object detection in real-world images in Section 7.2, and introduce *topological image modification* (TIM) to overcome these in Section 7.3. Also in Section 7.3, we discuss how image smoothing can be regarded as a *destructive* way of TIM, and introduce a new and more

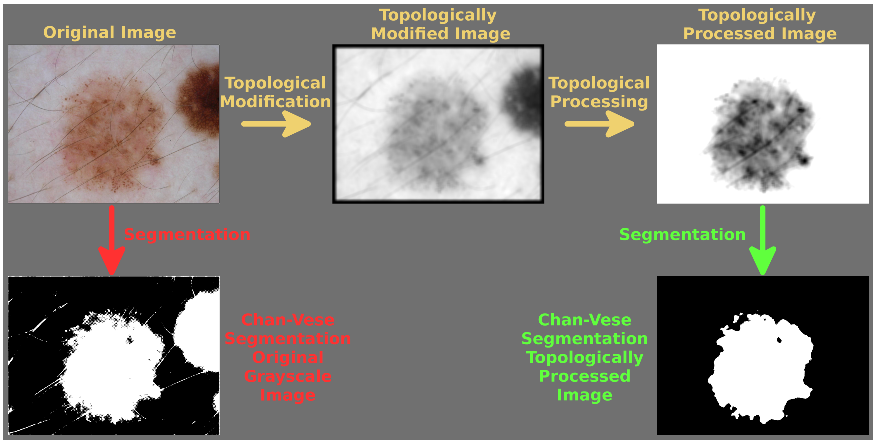

Original Image
Topologically Modified Image
Topologically Processed Image
Topological Modification
Topological Processing
Segmentation
Segmentation
Chan-Vese Segmentation Original Grayscale Image
Chan-Vese Segmentation Topologically Processed Image

Figure 7.1: An overview of how topological image modification and processing improves the performance of the completely unsupervised Chan-Vese segmentation algorithm [124].

constructive way, i.e, *border modification*, in Section 7.3. In Section 7.4 We show how TIM leads to a powerful new method for *topological image processing* (TIP), for which we introduce a new algorithm that marks objects in an image consistent with the (number of) inferred components from its *persistence diagram* (Algorithm 7). We demonstrate how TIM and TIP effectively improves three different generic unsupervised segmentation algorithms in Section 7.5, as well as three other methods in Section 7.5, through a case-study of the ISIC 2018 skin lesion images [130, 131].

7.2 Persistent Homology for Object Detection

Consider the 0-dimensional persistence diagram in Figure 2.13 of the image I shown in Figure 2.12. More formally, this persistence diagram is obtained through the sublevel filtration of the function

$$f : \Delta_I^1 \to \mathbb{R} : \sigma \mapsto \max_{p\in\sigma} \operatorname{gray}_I(p),$$

defined on the graph Δ_I^1 that is obtained by connecting all (vertically, horizontally, and diagonally) neighboring pixels (nodes) in the image, where $\operatorname{gray}_I(p)$ denotes the grayscale value

$$\operatorname{gray}_I(p) = \frac{1}{1000}\left(299\mathrm{R}_I(p) + 587\mathrm{G}_I(p) + 114\mathrm{B}_I(p)\right) \qquad (7.1)$$

given to pixel p in the RGB image I. Note that this is just the standard linear color to grayscale converter implemented in the PIL library in Python.

The darker pixels in I making up two different objects in the image, i.e., the '1' and '8' component, correspond to the most persisting components. Hence, persistent homology is well able to identify that there are two distinct important objects on this *toy* image.

We are now going to perform a similar analysis, but on a *real world* image instead, obtained from the ISIC 2018 data set. This data set can be obtained through `https://challenge2018.isic-archive.com/`. Figure 7.2 shows a skin lesion image I that contains multiple *true* objects: strands of hair, (part of) a non-relevant lesion connecting to the boundary, and a centered lesion: the object of interest. Since these objects are darker than the skin tissue, they should correspond to persisting components in the sublevel filtration of I. Again, we attempt to identify the important objects of I, is through the points in its persistence diagram marking components with a high persistence (Figure 7.2, Middle). Note that apart from the single component with infinite persistence, which will always be present for images, we also note two other relatively long persisting components, with a lifetime above 75. Including the component with infinite persistence, these correspond to three components that are all alive right before the lowest of their death-times (Figure 7.2, Middle).

This example illustrates the first problems that arises from applying topological persistence as a method for object detection in real-world images. Though topological persistence is stable in terms of noise, it is not robust to outliers. In the case of images, this means that a single or insignificant cluster of pixels can be identified as a significant component through persistent homology. This is the case for the component born at time $\sim$78 in Figure 7.2 (Right). Furthermore, persistent homology also identifies true but irrelevant objects, such as the lesion connecting to (and born through) the border of the image.

Although this example shows that topological persistence is sensitive to outliers, it is a well-known and important fact that it is stable under noise (Theorem

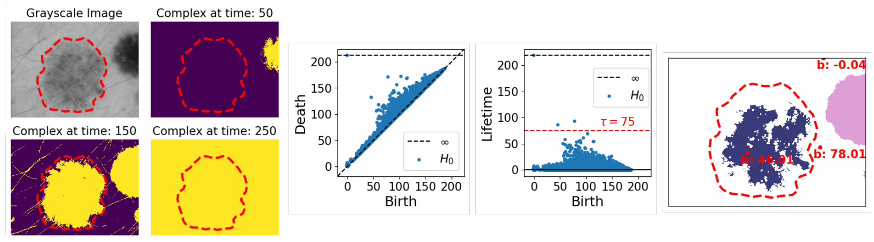


Figure 7.2: (Left). Three complexes $\bar{\mathcal{F}}(\mathrm{gray}_I)_t$ at different time steps in the original image I. (Middle). The resulting persistence diagram and lifetimes obtained by rotating the diagram. (Right). The identified components—those with lifetimes above the thresholds marked by the red striped line—right before their lowest death-time, as well as their birth-pixel and value. Ground truth segmentation borders are marked in red on all images.

2.5.18). Intuitively, this means that if there is only a small pixel-wise difference between two images of the same dimensions, their resulting diagrams will be close according to the bottleneck distance. This result is furthermore important to our current work, as our chosen implementation based on the Ripser library in Python for computing persistence does not allow to track the pixels through which particular components are born or die [132]. However, if each pixel in our image has a unique value, we can just match the birth time of a component to the corresponding pixel in the image. Hence, in practice, we apply a small amount of random noise to our image for this purpose. Due to the stability of persistence diagrams, this will not affect the performance of our method. Other implementations that do allow to track birth and death pixels can e.g. be found in the Dionysus 2 library in Python. Note that Algorithm 7 which we present below requires one of these methods to match diagram points to birth and death pixels.

7.3 Topological Image Modification

In this section, we introduce *topological image modification* (TIM): artificially altering the topological properties of an image, making it significantly more easy to extract relevant topological information. We will consider two types of topological image modification methods in this thesis: *image smoothing* (Section 7.3.1), and *border modification* (Section 7.3.2).

7.3.1 Image Smoothing

The first type of method for TIM we consider is *image smoothing*. Note that this method has previously been used in conjunction with persistent homology of images [21]. However, its true potential within the context of TIM remained unnoticed.

For each pixel p of an image I and $k \in 2\mathbb{N}+1$, we may consider a $k \times k$ square pixel neighborhood $\mathcal{N}_k(p)$ centered at p, and restricted to I, i.e., undefined beyond its borders. We can then define a new image I' from I by averaging over the neighborhood $\mathcal{N}_k(p)$ for each pixel p.

Image smoothing *destructs* topological properties resulting from outliers, decreasing their persistence or even preventing their birth. Furthermore, instead of destructing such insignificant objects, image smoothing may also destruct significant but irrelevant objects of an image. E.g., by smoothing the strands of hair in our example image I (Figure 7.2), they blend in with the surrounding tissue (Figure 7.3).

From the persistence diagram of the smoothed image I', we now only deduce one object in the image with a relatively long finite persistence (Figure 7.3). The corresponding component, as well as the component with infinite persistence, are

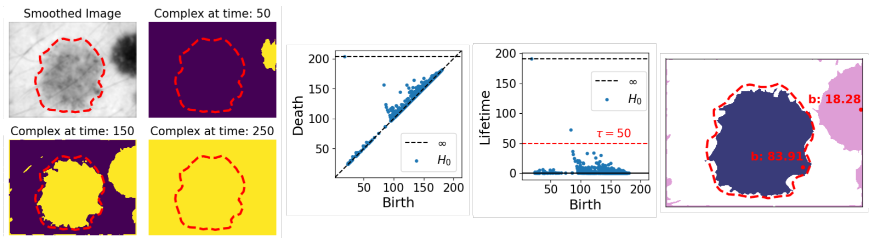


Figure 7.3: (Left). Three complexes $\bar{\mathcal{F}}(\text{gray}_{I'})_t$ at different time steps in the smoothed image I' ($k = 25$). (Middle). The resulting persistence diagram and lifetimes obtained by rotating the diagram. (Right). The identified components—those with lifetimes above the thresholds marked by the red striped line—right before their lowest death-time, as well as their birth-pixel and value. Ground truth segmentation borders are marked in red on all images.

also displayed in Figure 7.3. Note that there is no longer a component corresponding to a cluster of outlying pixels. Furthermore, the component with infinite persistence is now born through a true—although not the relevant—object, instead of the border.

Remark 7.3.1. *The fact that image smoothing can be regarded as a destructive way of TIM, can also be noted by observing the significant decrease in the number of points in the diagrams in Figures 7.2 & 7.3 after smoothing the image.*

7.3.2 Border Modification

We showed how image smoothing was able to destruct insignificant and irrelevant topological features in our image. However, depending on the prominence of irrelevant objects, image smoothing is insufficient for this purpose. This is shown in Figure 7.3, where the most persisting component actually corresponds to an irrelevant object of the image.

In the case of real-world images, some may have borders, some may have irrelevant objects, and some may only display the actual objects of interest. Hence, even with (a possibly different method for) TIM, in a generic setting, it becomes difficult to guarantee that the most persisting components correspond to the most important objects of an image, without prior information on their location in the image. Instead, we apply a simple, intuitive, yet powerful ‘trick’. More specifically, through TIM, we guarantee that the most persisting component does *not* correspond to the important object(s) in the image.

Border modification builds upon this idea through the generic property that in many real-world images, the object(s) of interest do not connect to the border, but the background does. Note that this is a strictly weaker assumption than assuming that the object(s) of interest are near the center of the image. More formally, border

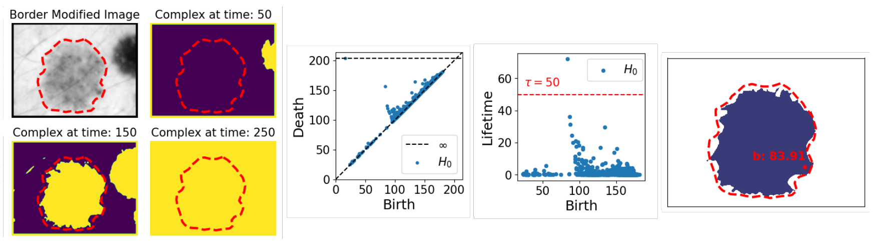


Figure 7.4: (Left). Three complexes $\bar{\mathcal{F}}(\mathrm{gray}_{I'_b})_t$ at different time steps in the border modified smoothed image ($l = 25$). (Top Right). The resulting persistence diagram and finite lifetimes obtained by rotating the diagram. (Bottom Right). The identified component—the single component with finite lifetimes above the thresholds marked by the red striped line—right before its lowest death-time, as well as its birth-pixel and value. Ground truth segmentation borders are marked in red on all images.

modification constructs a new image I_b from an image I, by ensuring that every pixel within a distance l of the border of I_b reaches the lowest value, while other values remain unchanged. Due to the elder rule (Section 2.5.2.1), this ensures that every object connecting to this border will be born through the border. Hence, all of these irrelevant components correspond to the single point with infinite lifetime in the persistence diagram of I_b. As such, we may restrict the analysis of our persistence diagram for identifying objects to the points corresponding to components with finite persistence. This is illustrated in Figure 7.4.

The advantages of border modification are the following.

- There is no bias towards the single point with infinite death-time in the persistence diagram (there will always be one for any image). This is especially useful when the birth (pixel) of the corresponding component marks an irrelevant or insignificant object (Figure 7.2), or when there are more than one relevant objects (possibly making up one 'meta object') displayed by the image (Figure 7.5).

- By ensuring relevant objects have finite persistence, we may automatically infer nontrivial thresholds to mark objects in the image through their death-times (Algorithm 7).

- By restricting our analysis to the points of the persistence diagram with finite persistence, any existing outlier detection method can be applied to automatically infer the number of objects displayed by an image based on their persistence (Section 7.4).

Remark 7.3.2. *Border modification can both be regarded as a constructive and destructive way for modifying topological features of an image. On the one hand, we construct a border such that the existence of a corresponding component with* *infinite persistence is ensured. On the other hand, this process discards all other points in the diagram corresponding to components born through a pixel of this border in the original filtration.*

7.4 Topological Image Processing

In the previous section, we showed how TIM results in persistence diagrams from which one may more efficiently extract both significant and relevant topological features of an image. In this section, we present *topological image processing* (TIP), i.e., image processing based on this topological information.

Our first step is to decide how many components are displayed by the image, through the distribution of the lifetimes. Note that we may restrict the diagram to only include finite lifetimes by applying topological image (border) modification.

In Section 7.3, we manually decided a threshold τ to identify significant components, as those with lifetime greater than τ. Without supervision, any standard outlier detection tool may be used to select such thresholds. However, we will use a method previously described by [80]. This method is based the fact that relevant peaks (of the function defining the filtration) can be extracted from the persistence diagram if it contains a band of a certain width that does not contain any points, as shown by [133]. More specifically, we look for the the largest empty region parallel to the diagonal we can draw into the persistence diagram. To achieve this, we simply iterate over all lifetimes in decreasing order, and track the difference between consecutive lifetimes. A threshold is then obtained by taking any τ between the two lifetimes where the largest of these differences is achieved. Note that our manually selected threshold in Figure 7.4 suffices this criterion. Furthermore, this procedure is especially useful in conjunction with TIM. If any consecutive difference in the ordered lifetimes would be infinite, it would always be selected. However, in Section 7.3, we showed that this is inefficient for real-world images, and may lead to irrelevant components. Furthermore, selecting the single component with infinite persistence in images is insufficient when multiple objects are of interest.

Remark 7.4.1. *[80] observed that this procedure results in a stable threshold τ, if the ratio of the width of the largest empty region to the mean width of all empty regions in the persistence diagram is greater than four. After border modification, smaller ratios indicate the absence of contrast between components in images, where it may be difficult to infer the objects trough an automatic procedure. Though we will not consider this step in our current work, this can be especially useful in medical applications, to identify the images from which it is difficult to identify the object(s) of interest (Figure 7.10a).*

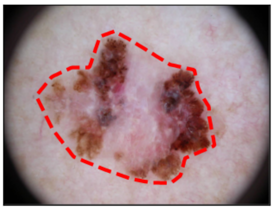
Once a threshold τ has been selected, we process our images as to increase the

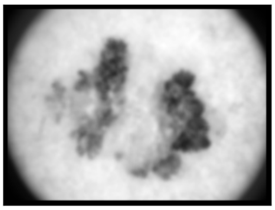
```
Data: Image I, persistence diagram D of I (infinite persistence is assumed
      to mark the border), threshold $\tau$
Result: Binary image J marking objects in I.
J, ds = zeros_like(I), list()
lifetimes = D.death - D.birth #obtain the lifetimes directly from D
obj_idxs = where($\tau <$ lifetimes $< \infty$)
    #identify diagram points marking significant finite lifetimes
obj_idxs = obj_idxs[argsort(D.death[obj_idxs]), 'desc')]
    #sort identified diagram points by decreasing death times
for idx in obj_idx do
    b = birth_pixel(idx) #identify the image pixel that corresponds to
                         #the birth time of this diagram point
    ds.append(death_pixel(idx)) #store the image pixel that corresponds to
                                #the death time of this diagram point
    C = component(I[I[b] $\leq$ I $<$ D.death[idx]], b)
        #get pixels connected to b right before its death
    new_dval = min(I[intersect(ds, C)], D.death[idx])
        #check if C contains death pixels of previous components
    C = component(I[I[b] $\leq$ I $<$ new_dval], b)
        #ensure C does not overlap with previous components
    J[C] = 1 #mark the component for this diagram point in the output
end
return J
```

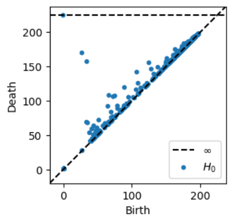
Algorithm 7: Pseudocode for an algorithm that marks objects in an image based on its persistence diagram.

contrast between the objects with a lifetime above τ, and the rest (the background) of the image. For this, observe that if any component with birth-time b and lifetime L dies through another component, due to the elder rule, the latter component has birth-time $b' \leq b$ and lifetime $L' \geq L$. This means that if any component is identified to be significant, the component causing its death is as well. This implies that Algorithm 7 provides a binary image, marking objects of the original image consistent with the number of inferred components through its diagram.

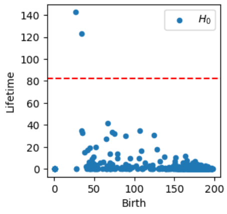
Finally, we apply multivariate interpolation to fill in the background pixels. More formally, for every pixel p, we determine the closest pixel $p_1, \ldots, p_k$, in each of the k identified components. p is then assigned to an interpolation of the values of pixels $p_1, \ldots, p_k$, by means of inverse distance weighting [134]. By applying this on our topological modified (smoothed) image, we obtain a smooth transition between our object(s) and the background, as well as between different parts of the background. Additional smoothing may be required if the result of Algorithm 7 is applied as a mask to the original image. Our method for TIP is illustrated through two examples images of skin lesions in Figures 7.5 & 7.6.


(a) *Original skin lesion image I_1. The ground truth border is marked in red.*

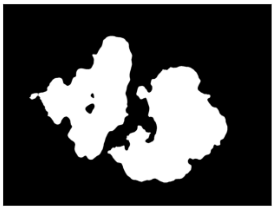

(b) *Topologically modified image I'_{1b} ($k = l = 25$).*



(c) *The persistence diagram of I'_{1b}.*



(d) *The finite lifetimes are used to select a threshold.*


(e) *Components in I'_{1b} right before their lowest death-time.*

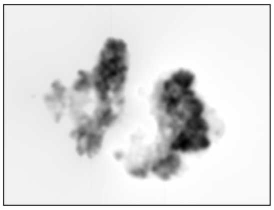
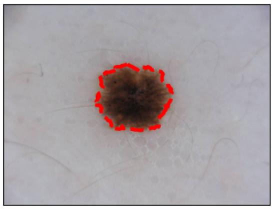
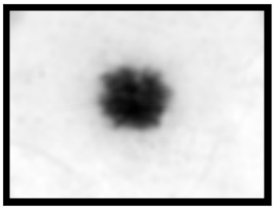
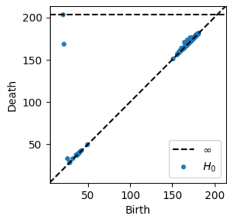

(f) *The topologically processed image for I'_{1b}.*

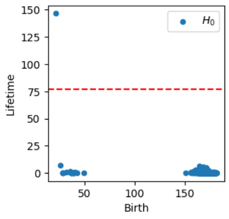
Figure 7.5: A first example overview of TIP.


(a) *Original skin lesion image I_2. The ground truth border is marked in red.*

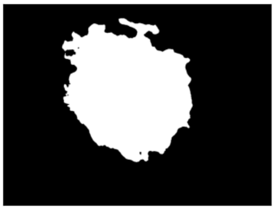

(b) *Topologically modified image I'_{2b} ($k = l = 25$).*



(c) *The persistence diagram obtained from I'_{2b}.*



(d) *The finite lifetimes are used to select a threshold.*


(e) *The component in I'_{2b} right before its death-time.*

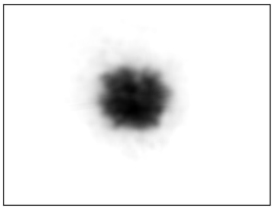

(f) *The topologically processed image for I'_{2b}.*

Figure 7.6: A second example overview of TIP.

When TIP is used as a first step for segmentation, one may wonder why we conduct our last step. E.g., Figure 7.5e already provides a reasonable segmentation of the components making up the skin lesion. This is because there is no clear gradient between the background and the border of the image. In this case, whenever we start including any background pixel, we rapidly include the majority of background pixels, resulting in the death of the relevant components through the border of the image, guaranteed to be included through TIM. However, when there is a particular gradient in the background that is darker near the object(s) of interest, such as on the images in Figures 7.6a & 7.6b, the identified component(s) will generally include many more pixels than those of the actual object (Figure 7.6e), only marking the area that *includes* the the object(s) of interest. Nevertheless, the relevant objects are significantly more highlighted in the topological processed images (Figures 7.5f & 7.6f).

7.5 Unsupervised Segmentation of ISIC 2018 Skin Lesion Images

In this section, we qualitatively and quantitatively show how our method effectively improves the performance of three generic unsupervised binary segmentation models on the ISIC 2018 skin lesion images. In Section 7.5.1 we will discuss the data and (topological) image processing steps. In Section 7.5.2 we present the binary segmentation algorithms that we will consider for improvement through TIM + TIP. In Section 7.5.3 we discuss four metrics that we will use to evaluate how our method improves binary image segmentation. Finally, our experimental results will be discussed in Section 7.5.4.

7.5.1 Topological Image Processing of Skin Lesion Images

We consider 2594 skin lesion images from the ISIC 2018 data set. The relevant object on each image was a skin lesion, for which a ground truth segmentation was available. We will convert each image to grayscale to construct the scalar filtration function (7.1), after which we apply random normal noise ($\sigma^2 = 0.01$), and topologically modify each image through smoothing and border modification. Note that the purpose of the addition of noise is not to affect our performance, but to identify birth-pixels through Algorithm 7, as discussed in Section 7.2. For each image I with diagonal length $\Delta(I)$, we set the smoothing parameter $k \sim \Delta(I)/25$ and the border width $l \sim \Delta(I)/100$, while satisfying the integer requirements. Topological image processing is then performed on the topological modified images. We will evaluate three generic unsupervised segmentation methods—presented in the following section—on all three of the original image, the smoothed only image (using the same image, prior to border modification and TIP), as well as the topologically processed image.

7.5.2 Presenting the Binary Segmentation Algorithms

We will consider the following three algorithms straightforwardly lead to binary segmentations for all three of the original skin lesion images, the smoothed images, and the topologically processed images.

Chan-Vese Segmentation First, we will consider the **Chan-Vese** segmentation algorithm [124]. This is a very generic segmentation algorithm designed to segment objects without clearly defined boundaries, not particularly targeted towards skin lesion, or even biased towards darker objects. The algorithm is based on level sets that are evolved iteratively to minimize an energy function. We used the standard settings of the algorithm implemented in the SCIKIT-IMAGE library in Python.

ISODATA Threshold Segmentation Next, we will consider an unsupervised threshold segmentation where the segmentation is composed by the pixels with a value below, i.e., darker than a certain threshold. The threshold was selected based on the **ISODATA** method [125], using the standard settings of the algorithm implemented in the SCIKIT-IMAGE library in Python.

Isocontour Segmentation We furthermore consider a segmentation algorithm based on identifying **isovalued contours** in the image, i.e., contours in the image where the pixel value remains constant. For this, we use a special case of the marching cubes algorithm implemented in the SCIKIT-IMAGE library in Python [126]. The final segmentation is then obtained by filling in the obtained contours—regarded as polygons in the Euclidean plane—using the OPENCV library. This method differs from a threshold segmentation, in that the isovalued contours are always closed (unless they intersect the border of the image), and that lighter patches enclosed by the contour(s) will also be filled in. The main hyperparameter is the constant which the isocontours should value to. We will simply consider the mean value of the image for this purpose.

7.5.3 Metrics for Evaluating Binary Segmentations

Marking pixels included in the segmentation as positives, we will consider the following metrics for evaluation:

- The **Accuracy**
$$\frac{TP+TN}{TP+TN+FP+FN} \in [0,1],$$

a common validation metric for binary classification.

- The **Sørensen-Dice Coefficient** [135, 136]

$$\frac{2TP}{2TP + FP + FN} \in [0, 1],$$

a statistic assessing the similarity of two samples.

- **Matthews Correlation Coefficient** [137]

$$\frac{TP * TN - FP * FN}{(TP + FP)(TP + FN)(TN + FP)(TN + FN)} \in [-1, 1],$$

measuring the correlation between truth and predicted.

- The **Inclusion Score** (in general machine learning also known as *recall*)

$$\frac{TP}{TP + FN} \in [0, 1],$$

assessing how well the predicted encompasses the truth.

7.5.4 Experimental Results

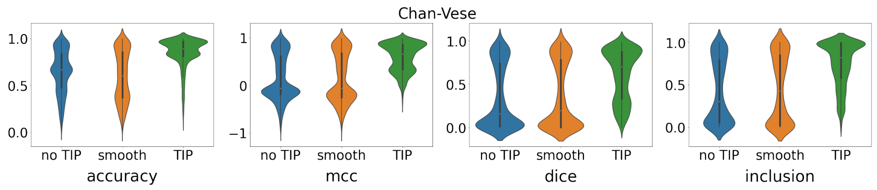
Table 7.1 shows that the average performance for each of the considered metrics, and each of the considered segmentation methods, before TIP, after smoothing, and after TIP. Figure 7.7 shows the respective distributions. We consistently observe strong improvements of the segmentations after TIP, with two exception where the inclusion score is better with smoothing only for the ISODATA and Isocontour method. However, as can be deduced from the other metrics, this is accompanied by a large number of false positives. This is exactly as expected from our method, as through TIP, we disregard the irrelevant objects in the image.

Figure 7.8a illustrates why the inclusion score most significantly increases for the Chan-Vese segmentation algorithm after TIP. Without TIP, the algorithm appears to often segment the inverse of the actual the lesion. Furthermore, note that in this example, the algorithm did not converge well (after the standard set number of iterations), even when TIP was applied. This results in a checkerboard-like pattern (used to initialize the algorithm) surrounding the actual segmentation, and greatly affects the accuracy, Dice, and Mcc. score. This occurred rather commonly (in approximately 600 topologically processed images), explaining the bimodality of the corresponding distributions in Figure 7.7a. Nevertheless, we observe that TIP successfully fulfills its purpose, identifying the lesion in the image and increasing its contrast with the background (Figure 7.8a).

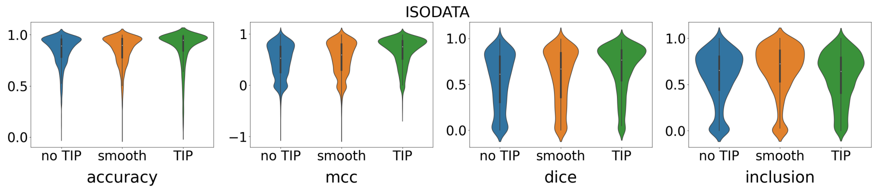
Figure 7.8b illustrates how TIP improves (ISODATA) based threshold segmentation. Without TIP, the darkest parts of the image include the irrelevant strands of

	metric	no TIP	smooth	TIP	minimal improvement
Chan-Vese	Acc.	0.643	0.599	**0.847**	+0.204
	Dice	0.346	0.367	**0.614**	+0.247
	Mcc.	0.152	0.156	**0.590**	+0.434
	Inc.	0.420	0.454	**0.746**	+0.292
ISODATA	Acc.	0.850	0.851	**0.875**	+ 0.024
	Dice	0.543	0.587	**0.672**	+0.085
	Mcc.	0.481	0.528	**0.657**	+0.129
	Inc.	0.587	**0.660**	0.584	-0.076
Isocontour	Acc.	0.680	0.798	**0.893**	+0.095
	Dice	0.439	0.532	**0.704**	+0.172
	Mcc.	0.335	0.492	**0.687**	+0.195
	Inc.	0.757	**0.825**	0.785	-0.040

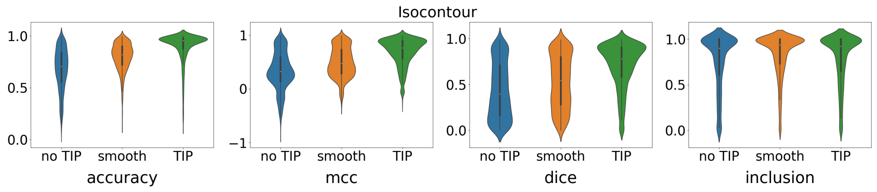
Table 7.1: Averaged performances of different segmentation algorithms before TIP, after smoothing, and after TIP.



(a) Performance distributions of the Chan-Vese segmentation algorithm.

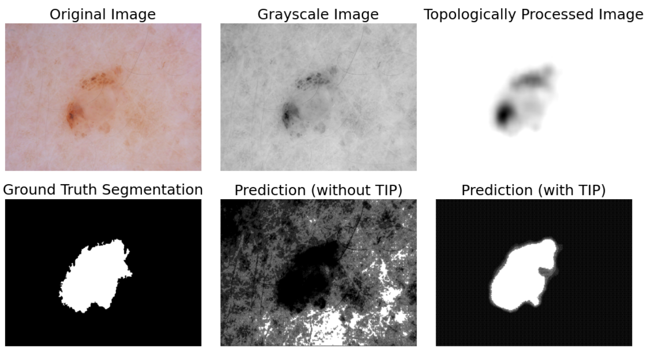


(b) Performance distributions of the ISODATA threshold segmentation algorithm.

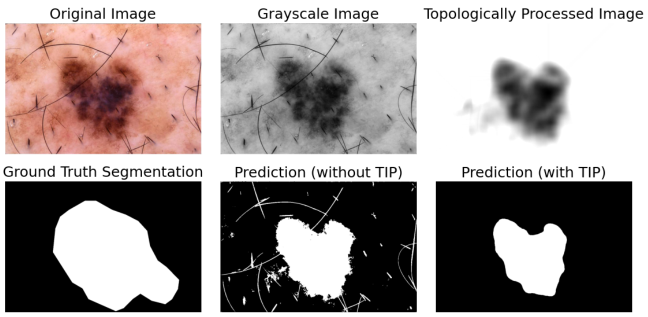


(c) Performance distributions of the Isocontour segmentation algorithm.

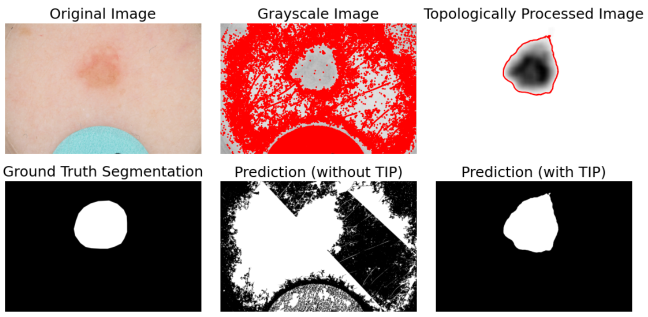
Figure 7.7: Performance distributions of different segmentations algorithm before TIP, after smoothing, and after TIP.



(a) The Chan-Vese segmentation algorithm segments the inverse of the relevant object in the image without TIP, and the correct object with TIP. Note that the algorithm did not converge well after the standard set number of iterations. ***Without TIP:*** *Acc. = 0.570, Dice = 0.020, Mcc. = -0.219, inclusion = 0.039.* ***With TIP:*** *Acc. = 0.938, Dice = 0.767, Mcc = 0.742, Inclusion = 0.890.*



(b) The ISODATA based threshold segmentation algorithm segments many dark strands of hair along with the lesion when no TIP is applied. These strands are disregarded through TIP, resulting in a much better final segmentation. ***Without TIP:*** *Acc. = 0.811, Dice = 0.657, Mcc = 0.571, Inclusion = 0.526.* ***With TIP:*** *Acc. = 0.809, Dice = 0.618, Mcc = 0.588, Inclusion = 0.447.*



(c) Isocontours (red) valuing to the image mean capture the lesion on the image much better after TIP. ***Without TIP:*** *Acc. = 0.553, Dice = 0.259, Mcc = 0.262, Inclusion = 0.964.* ***With TIP:*** *Acc. = 0.986, Dice = 0.919, Mcc = 0.913, Inclusion = 0.965.*

Figure 7.8: Three examples illustrating how each one of our considered segmentation algorithms benefits from TIP.

hair, and there does not exist any threshold that can be used to include the pixels of the lesion and only those. However, after TIP, the darkest object on the image is the lesion itself, as the strands of hair are disregarded. In this case, (ISODATA) based threshold segmentation does lead to a good result. One may also argue whether the 'ground truth' segmentation is actually better than the provided segmentation after TIP in this example. Clearly, TIP significantly improves the segmentation. However, three of the four considered metrics point otherwise (Figure 7.8b).

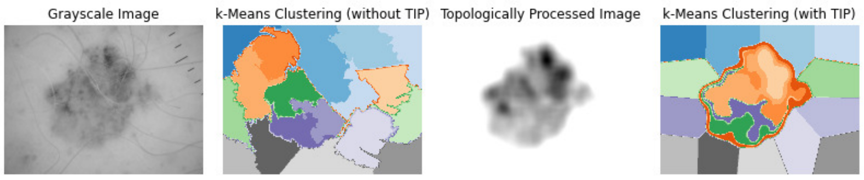
Finally, Figure 7.8c illustrates how isocontour based segmentation benefits from TIP. First, after TIP, due to the high contrast between the object and the background, and the homogeneity of the background, it becomes easy to select an appropriate value the isocontours should value to, as the mean of the image values simply suffices. Second, without TIP, there are often many such isocontours, whereas there are commonly only one or a few (correctly) identified contours after TIP (Figure 7.8c).

7.6 Improving Other Generic Methods

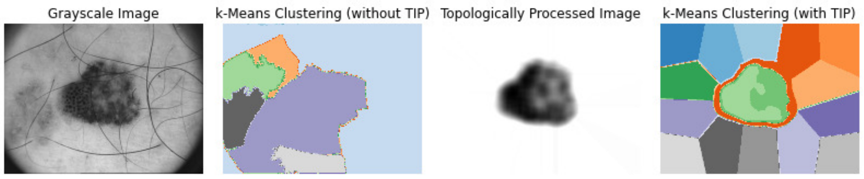
In this section, we discuss some other generic models for which TIP may result in an effective increase in performance. Unlike the previous three models, they will not be quantitatively evaluated on a large scale, as they are not necessarily binary segmentation methods, or we lack a fair and consistent comparison to the non-topologically processed images.

Clustering Based Segmentation Superpixel segmentation algorithms use clustering algorithms in the color space to produce *oversegmentations* (more segments than necessary), and are generally less effective when straightforwardly applied for the task of binary segmentation. Different segments (clusters) in the result are referred to as *superpixels*. Figures 7.9a & 7.9b show the result of a k-means clustering based superpixel segmentation algorithm [127], to segment a skin lesion image in 20 different superpixels, before and after TIP. Only after TIP, we observe clear superpixels belonging to the lesion, and only to the lesion. Hence, TIP provides a promising approach towards constructing effective cluster-based (binary) segmentation algorithms.

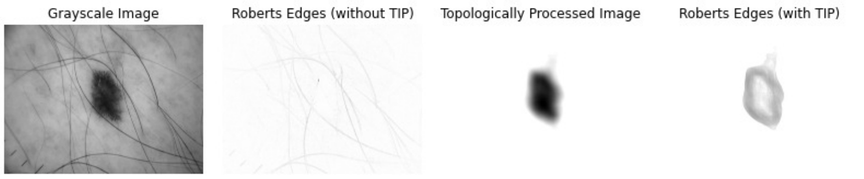
Edge Detection Edge detection methods searches for linear segments that correspond to edges and borders in an image. They differ from segmentation methods in that they do not target the output of well-defined (2D) areas. Figures 7.9c & 7.9d show the result of *Roberts' cross operator* for edge detection [128], on a skin lesion image before and after TIP. Before TIP, we infer some strands of hair (Figure 7.9c), or even no edges are clearly deduced (Figure 7.9d). After TIP, we infer many edges characterizing the lesion, and only those.


Grayscale Image
k-Means Clustering (without TIP)
Topologically Processed Image
k-Means Clustering (with TIP)

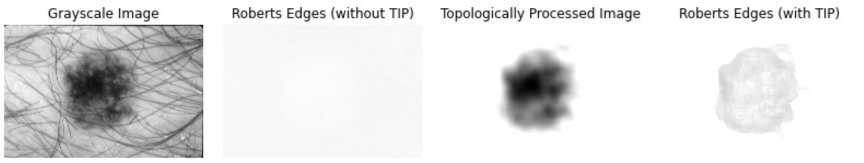
(a) A k-means superpixel segmentation algorithm marks clear segments belonging to the lesion only after TIP (first example).


Grayscale Image
k-Means Clustering (without TIP)
Topologically Processed Image
k-Means Clustering (with TIP)

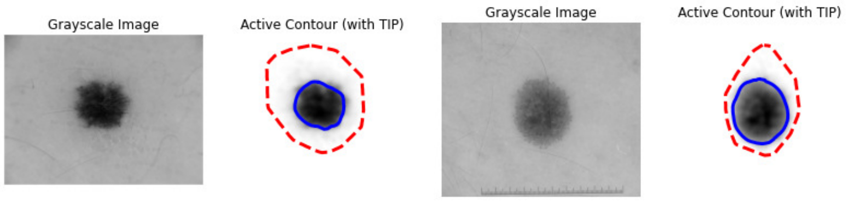
(b) A k-means superpixel segmentation algorithm marks clear segments belonging to the lesion only after TIP (second example).



(c) Roberts' cross operator for edge detection marks edges characterizing the lesion only after TIP (first example).



(d) Roberts' cross operator for edge detection marks edges characterizing the lesion only after TIP (second example).



(e) Two example images where Algorithm 7 is used to initialize the active contour model to provide an effective segmentation of the lesion (interior of the blue line). The initialization (red line) of the first example is the boundary of the convex hull of the mask in Figure 7.6.

Figure 7.9: Examples of three other generic and unsupervised models to which TIP provides an effective method to improve their results.

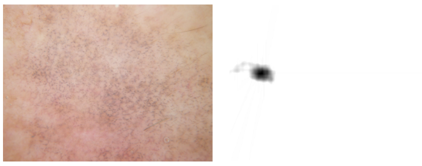
Active Contour Segmentation The active contour model is a method to fit open or closed splines, referred to as *snakes*, to lines or edges in an image [129]. It is based on the minimization of an energy function, similarly to the Chan-Vese segmentation algorithm. Unlike the previous two examples, this method does lead to a straightforward binary segmentation, by taking the interior of the resulting snake. It requires an initial estimate surrounding the object of interest, making it difficult to apply to the original (skin lesion) images in a consistent and effective way, without additional supervision. However, through TIM and TIP, we are able to provide both an effective initialization and segmentation. More specifically, as discussed in Section 7.4, Algorithm 7 does not provide a straightforward segmentation of the objects, rather it marks a surrounding area of the object of interest, containing no other significant objects. Hence, we can use the areas marked by Algorithm 7 to initialize the active contour segmentation algorithm. Figure 7.9e shows the result of this method, where we used the boundary of the convex hull of this area as the initialization, leading to effective segmentations of the topologically processed images.

7.7 Discussion and Conclusion

We introduced topological image modification (TIM) as a method for enhancing the ability to extract both relevant and significant topological information from an image. Although image smoothing has been applied in conjunction with persistent homology of images before, its true potential as a destructive topological image modifier has not been studied in detail. Furthermore, we introduced a powerful new method for TIM, i.e., border modification, sensible on three different levels. First, we discard all bias towards the single component with infinite persistence. Second, we may automatically and consistently separate objects from the background through their finite death-times (Algorithm 7). Third, any outlier detection method to automatically infer the number of components through the persistence diagram of the image becomes well-defined.

We introduced the concept of, as well as a new method for topological image processing (TIP). We showed how this method significantly increased the performance of six different generic and unsupervised models and algorithms through a wide variety of of skin lesion images from ISIC 2018. Furthermore, this increase in performance was extensively quantified on all 2594 skin lesion images, for the three algorithms that led to a straightforward binary segmentation method before and after TIP. Though this is a very domain-specific application, our method for TIP is very generic, resting on the assumptions that outliers can be destructed through smoothing the image, and that the relevant object(s) are away from the border of the image.

In any of the shown examples, we observe that our method of TIP fulfills its


(a) (Left) The original lesion image. (Right) Topologically processed image.

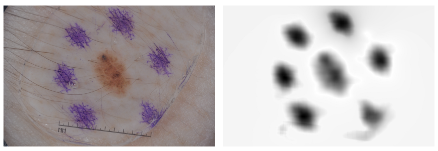

(b) (Left) The original lesion image. (Right) Topologically processed image.

Figure 7.10: Two example images where our current version of TIM + TIP does not capture the full relevant object and only the relevant object.

purpose, correctly identifying the relevant objects in the image and increasing their contrast with the background. Interestingly, none of the segmentation methods we considered, showed to be effective for the task of skin lesion segmentation prior to TIP. Furthermore, even in conjunction with TIP, these methods maintained their genericity. Naturally, these methods can be further improved for the particular task of skin lesion segmentation, by trading off their genericity with their performance. E.g., though the ground truth lesions always connected, none of our considered segmentation algorithms necessarily outputs connected segmentations. Their results may be post-processed to accommodate for this restriction.

Since we only considered unsupervised models in our experiments, it may be unrealistic to expect similar performances as the state-of-the-art supervised models, such as convolutional neural networks [122, 123]. Rather, we evaluated our work in an unsupervised context to show that TDA using TIM and TIP improves the task of skin lesion segmentation using generic unsupervised segmentation algorithms. It is left to investigate how TIP may enhance the ability to learn in a supervised setting, e.g., as an additional channel to a convolutional neural network based model, or through combined architectures, for either segmentation or classification problems [117].

As with any method, there are some limitations to our method, both for skin lesion images, as well as for more general applications. We assumed that the relevant objects of our image were darker than the surrounding background, i.e., they have a lower pixel value. In applications where they are actually lighter, one can easily apply our method by constructing the *superlevel filtration* instead of the sublevel filtration, capturing topological information equivalent to the sublevel filtration of the image after negating its pixel values.

A more difficult problem is when there is little to no contrast between our object of interest and the background (Figure 7.10a). We argue that any learning model, unsupervised or supervised, will find it challenging to correctly identify objects in such images. However, we may automatically recognize these particular types of images based on their persistence diagrams, as stated in Remark 7.4.1.

Another difficulty is when more prominent but irrelevant objects are separated from the boundary of the image. E.g, in Figure 7.10b, many irrelevant objects, such as the corners of the image, the strands of hair, and the ruler, are destructed in our topologically processed image. However, the surgical marker surrounding the lesion still remain. A different function defining the filtration on the image than the customary grayscale (7.1), possibly nonlinear in the color channels, that e.g. accounts less or not for the purple colors of the image, may be more appropriate in this case.

The fact that death times commonly occur after the full lesion has been included, prevents us from using persistence diagrams of the topologically modified images for a direct segmentation algorithm. However, in practice, a gradient between the lesion and the background of the image that results in such 'late' death-time may also indicate a region of inflammation around the lesion (Figure 7.6). Unfortunately, these regions are often disregarded in the 'ground truth' (Figure 7.6a), and further exploration of this interesting property is required.

Our method works well on images displaying one or few objects of interest on a uniform, noisy, or textured background. This makes skin lesion images an ideal application. For images fully composed of many objects (e.g., a street, cars, houses, trees, ...), other types of models may be more applicable.

8 Concluding Remarks and Future Work

In this chapter, we conclude upon our work (Section 8.1), and discuss further directions for and applications of topological inference in graphs and images (Section 8.2). Finally, we discuss our work in the broader context of machine learning and real-world applications in Section 8.3.

8.1 Conclusion

In this thesis, we thoroughly illustrated that simple graph-structured models occur naturally in many real-world graphs. However, this does not necessarily imply that many graphs are the (causal) result of these models. This makes it difficult to mathematically formalize the concept of topological models in graphs, as discussed in Chapter 3. Yet, we presented a variety of methods for effectively inferring these models across a broad spectrum of realistic applications.

In Chapter 4, we showed how local topological data analysis (LTDA) can provide local topological characterizations of metric data approaching metric graphs, in terms of degrees. Furthermore, we showed that one may also practically deduce the presence of cycles without the need of 1-dimensional persistent homology. By both storing and using this topological information, we were able provide effective reconstructions of metric graph models by means of clustering algorithms. These methods are inspired by the fact that exact knowledge of the degrees of all points in the theoretical model, or even less specific whether they are different from 2 or not, allows one to exactly reconstruct the model. Due to this reason, for metric graph reconstruction methods to work well, local topological information needs to be inferred correctly near every data point. Therefore, they are ideally applied to more clean and/or processed data, that approximates the underlying model well. The advantage of these methods is that they can directly model cycles, and mark important regions through the used clustering algorithms.

In Chapter 5, we introduced a completely different approach towards topological inference in graphs. Here, we inferred topological subgraph models, called backbones, based on the assumption that in real-world graphs 'noise' and 'outliers' surround the actual topological model underlying the graphs. Unlike reconstruction methods, these assumptions apply to the empirical data, rather than the ground truth model. Hence, our method for backbone inference satisfies much better generalization and robustness properties. In contrast to the degree of a node, our introduced boundary coefficient (BC) captures the coreness of a node well in both the small-world network model, as well as the non-small-world network model. This coefficient allowed us to construct effective forest-representations, i.e., f-pines, for inferring backbones through the Constrained Leaves Optimal subForest (CLOF) problem. We qualitatively and quantitatively confirmed its effectiveness for many types of graphs, ranging from social networks, to earthquake locations scattered across the Earth, and high-dimensional cell trajectory data.

In Section 5.8 we showed that even high-ranked state-of-the-art approaches in the field of cell trajectory inference (CTI) struggle to infer trajectories in many examples. They often fail to capture the geometry of the model, or predict the number of leaves correctly. In Chapter 6 we developed topological signatures accompanied by theoretical justification, that allow their use to study and quantify this particular problem in more detail. We furthermore showed that these signatures correlate well with the performance of current CTI methods, and as such provide a new way of performing data quality control within this field.

Finally, in Chapter 7 we applied TDA for a different kind of graph-structured data, this being images. We showed that although 0-dimensional persistent homology can be used for object detection in images, a direct approach towards this lacks robustness properties. We therefore introduced topological image modification (TIM) to enhance the ability to extract both significant as well as relevant topological information from real-world images. This information was then consecutively used for topological image processing (TIP) to increase the contrast between important objects and irrelevant objects or the background of the image. We furthermore qualitatively and quantitatively evaluated how TIM and TIP improve a wide variety of image segmentation methods in an unsupervised setting.

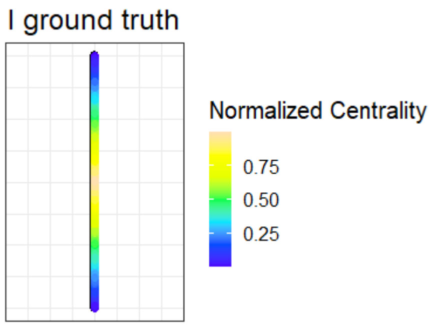
8.2 Future Work

In this section, we discuss a variety of interesting research topics that arise from the results and observations in this thesis. We will subdivide these according to chapter(s) and topic.

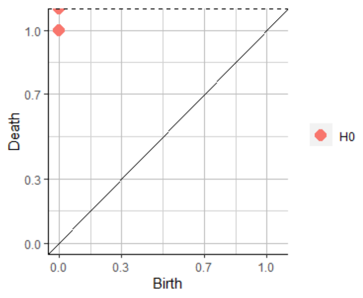
Formalizing Topological Models in Graphs As mentioned in Section 3.5, how to mathematically formalize the concept of topological models in graphs is one of the most important open problems resulting from this thesis. E.g., one might formalize these through random graphs for different 'classes' of graphs separately. As discussed in Section 3.4.2, these classes could specify whether the graphs are given or derived from point cloud data, whether there is a causal relationship between them and the model, whether they satisfy the small-world network model, or how nodes in the model relate to the nodes they represent. Furthermore, these classes often largely overlap (although not necessarily completely). E.g., proximity graphs derived from low-dimensional point cloud generally do not satisfy the small-world network model. However, whether such a finite classification of formalized topological models in graphs is even possible without limiting their applicability, is an open problem. Furthermore, even though we pointed out the difficulties for providing a universal formalization of topological models in graphs, we showed that there do exist methods that may effectively infer such models in graphs that would be distinct according to such classification. This makes studying how to formalize these models that more interesting.

Topological Libraries [Chapters 4 & 6] As discussed above, LTDA and the graph reconstruction methods derived thereof area ideally applied to clean, processed data, or data with a low underlying complexity of the model. Accompanied with a (number of leaves) inference method [80], the signatures introduced in Chapter 6 may provide some additional robustness to these methods. More specifically, one would not require to introduce an inner radius r' parameter anymore. Furthermore, one could develop a *topological library* of possible ground truth local topological signatures to match empirical local neighborhoods. One may even just perform these matchings to the persistence diagram of a line topology to identify edge and non-edge points, which is (theoretically) sufficient to reconstruct the global model. The restriction to constructing (local) Rips graphs will then not be required for inferring local topologies. Indeed, instabilities in terms of the number of connected components in punctuated neighborhoods of different type of proximity graphs may then be identified through points close to the diagonal of the diagram, e.g., as in Figure 6.5.

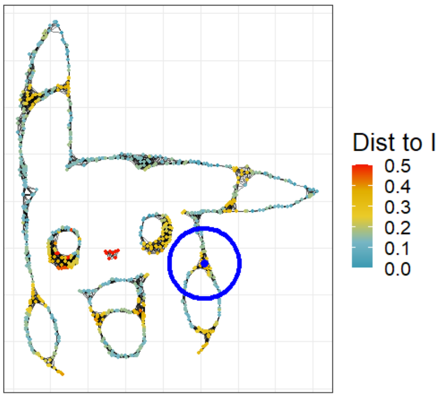
Figure 8.1 illustrates of how this may work in practice. First, we construct a topological library $\mathfrak{L}$ containing one persistence diagram $\mathcal{D}$ (Figure 8.1b), ob-



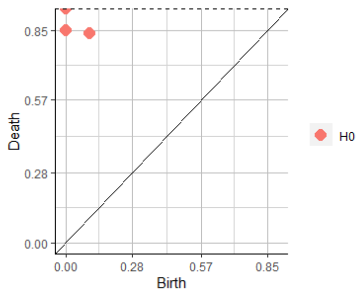
(a) A possible ground truth local topological model and its normalized centrality are used to construct a topological library containing one diagram.



(b) The single persistence diagram $\mathcal{D}$ in the topological library $\mathfrak{L}$, which captures a local linear model.



(c) A Rips graph G constructed from a point cloud data set Pikachu. Each node v is mapped to the bottleneck distance between the persstence diagram of the sublevel filtration of the normalized centrality function on $G[\bar{B}_{d_G^{\text{unw}}}(v,3)]$ and the diagram in the topological library.



(d) The local signature obtained from the node and its neighborhood shown in blue on Figure 8.1c.

Figure 8.1: Within the context of LTDA, topological libraries may provide a more robust way to discover edges, leaves, multifurcations, and isolated points, with better generalization properties.

tained from the sublevel filtration of the normalized centrality function on a linear metric graph (Figure 8.1a). Note that the resulting signature is independent of the length or any curvature of the ground truth model of which it is computed. More precisely, it always equals the set $\{(0,\infty),(0,1)\} \cup \{(x,x) : x \in \mathbb{R}\}$. In Figure 8.1c we consider the same Rips graph G constructed from the metric graph representing Pikachu in Section 5.5.4. For every node $v \in V(G)$, we consider a neighborhood $B_{d_G^{\mathrm{unw}}}(v,r) = \bar{B}_{d_G^{\mathrm{unw}}}(v,r-1)$. The local signature of v, which is the persistence diagram obtained through the sublevel filtration of the normalized centrality function on $G[\bar{B}_{d_G^{\mathrm{unw}}}(v,r-1)]$, is then matched to $\mathcal{D}$ through the bottleneck distance. An example of such signature for the node v marked in blue in Figure 8.1c is shown in Figure 8.1d. By mapping each node v to the such obtained bottleneck distance, we are able to identify nodes representing edges (low bottleneck distance) and nodes representing non-edges (high bottleneck distance), as shown in Figure 8.1c.

Nevertheless, further exploration how to decrease the sensitivity of to the choice of the (outer) radius r under a non-uniform amount of noise may still be required. This can e.g. be noted from the thicker lower parts of Pikachu's eyes in Figure 8.1c, which for the chosen radius represent 'blobs' rather than edges.

Scalability of the Boundary Coefficient [Chapter 5] Our method for inferring backbones shows to scale well to thousands of nodes. At first sight, this may not be enough for many practical applications, such as large network embedding [102, 138]. Nevertheless, there are many practical examples, such as cell trajectory data, where identifying the underlying topology for graphs of this order remains an important problem. Furthermore, our method may scale to larger order graphs by optimizing or even parallelizing our current implementation to compute boundary coefficients, the most expensive part of our method. Algorithms for approximating these coefficients may be investigated on a theoretical, probabilistic, and experimental level as well. E.g., fast algorithms for all-pairs approximate shortest paths are discussed in [139].

Generalization to more Complex Backbones [Chapter 5] One may increase the *local scope* of the boundary coefficient. I.e., the BC currently only averages over pairs of neighbors, but it may as well consider neighbors of neighbors, and so on. In this way, we may be able to effectively mine *topological skeletons* in data with a a locally higher intrinsic dimension. An example of this would be the Swiss Roll from Section 5.7, where we increase the width of the manifold (along the z-axis), while also increasing the sparsity of the proximity graph.

We experimentally demonstrated that modeling the topology of or graph by means of a (sub)forest is not a severe limitation when it comes to cycles. These may be more efficiently discovered through persistent homology if our backbone provides a significant size reduction, while remaining spread out across the underlying topology. However, we have yet to provide an effective method for 'lifting' the holes found by means of topological persistence to our forest-structured backbone, i.e., closing the gaps corresponding to these holes. Persistent homology has also been connected to minimum spanning trees and their higher-dimensional analogues of minimum spanning acycles on a theoretical level [140]. Connecting these results to CLOF may lead to new theoretically well-founded approximation algorithms for computing persistence [40, 141].

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

Applications of Backbones [Chapter 5] Considering backbone inference in graphs, we only focused on applications within the field of topological data analysis. Correctly identifying the graph-structured model is the exact purpose of CTI methods. For other graphs such as social networks, we also showed that we can meaningfully identify backbone structures, both on a qualitative and quantitative level. Our procedure provides a way to visualize or obtain insight into their underlying structure. Nevertheless, this is often not the end goal for these networks. Hence, a wide variety of new applications of backbones, such as community detection, subgroup discovery, and graph embeddings, is yet to be discovered. E.g., [102] introduced a heuristic algorithm to iteratively simplify a graph, as to increase the performance of any existing graph embedding method through better initializations. Hence, initializing the embedding through a well-chosen backbone can lead to a graph embedding method that respects topological properties of the graph.

Finally, one may investigate how backbone extraction improves existing models for graph-structured data, and vice versa. E.g., *graph convolutional networks* (GCNs) have recently led to many new applications for graphs [142]. On the one hand, prior knowledge of nodes near or on the core structure of the graph may enhance the ability to learn from graphs through better initializing a GCN. This is similar to the method described above for improving graph embeddings, which is also one of the many applications of GCNs. On the other hand, similar to how an initial dimensionality reduction improves the performance of cell trajectory inference, graph autoencoders [143] may serve as a tool to find a latent or denoised representation of the graph, prior to the final backbone extraction.

Cell Trajectory Inference [Chapters 5 & 6] In case of CTI, we noted that even high-ranked cell trajectory methods, according to [9], often struggle to capture the geometry of the underlying model, or its exact number of leaves (Figures 5.31 &

5.33). From Chapter 6, we conclude that for many data sets, branches are difficult to separate from each other due to a high amount of noise (even after a dimensionality reduction), or that branches which are relatively short according to some main trajectory are left undetected (intuitively, our 'elbow' in Figure 5.10b will occur too soon). In Section 5.8, we also showed how our method for backbone inference may benefit from a more accurate leaf inference method for the task of cell trajectory inference, increasing its performance under this knowledge (Figure 5.31). Again, from Chapter 6 we conclude that developing such method based on purely topological information may be difficult. However, the signatures introduced in this chapter allow do us to qualitatively and quantitatively evaluate which data representation (dimensionality reduction and/or proximity graph) better captures topological properties of cell trajectory data. Naturally, finding more effective representations will lead to a better performance of both our as well as other methods. This is the formal subject of [6].

Topological Image Modification and Processing [Chapter 7] The idea behind topological image modification—altering the topological properties of an image to improve the detection of both relevant and significant objects—is very generic. Hence, a wide variety of topological image modifiers, as well as new applications of topological image processing to supervised learning and domains other than (segmenting) skin lesions are yet to be discovered. A simple example of an additional topological image modifier would be *color modification*. E.g., through a nonlinear function on the color channels we may decrease the prominence of the purple surgical markers of the image in Figure 7.10b. This may prevent their detection through the resulting persistence diagram. Furthermore, adding extra channels that capture topological information to images may prove to be valuable for convolutional neural networks models. Unfortunately, many current state-of-the-art pre-trained layers have been optimized for 3-channel (RGB) images. Hence, we might not be directly able to exploit their advantage of having been trained on sometimes millions of images [144], and require further investigation into novel model architectures that benefit from this additional topological information.

8.3 Our Work in a Broader Context

We both qualitatively and quantitatively showed that we developed effective tools for topological inference and analysis across a wide variety of real-world graphs and images. Naturally, not every of our considered 'applications' appears to be equally important at the time of writing. E.g., it might be obvious that cell trajectory inference is currently a more relevant and difficult problem, than inferring the underlying model of the Harry Potter network. The purpose of this section is hence to discuss how our work may impact society and show real-world value.

Something that we did not discuss is that methods particularly targeting CTI often deal with *pseudotime analysis*, which corresponds to ordering cells along the (inferred) trajectory [9]. Given a (inferred) trajectory, cells can be directly ordered in various ways, e.g., by projecting them onto the trajectory. In a matter of fact, this is exactly implemented in the wrapper provided by [9]. Note that early methods for CTI prioritized ordering cells correctly over inferring the actual trajectories [9,58]. Cell trajectories and these pseudotimes offer a transcriptome-wide understanding of dynamic processes [9]. In cancer, they can be used to identify (dis)continuity in cell states, and indicate the mode of tumor evolution [58–61]. In immunology, understanding the cellular transition dynamics modeled by cell trajectories and how they can be modified to improve human health is one of the central objectives [62]. These examples accompanied by the ongoing efforts to construct transcriptomic catalogs of whole organisms [9, 145, 146], clarify the importance of effective and scalable CTI methods.

Nevertheless, we merely showed that our method for CTI through backbone inference (Section 5.8) is competitive to, but does not outperform the state-of-the-art. Although our provided method for CTI may show to be a complementary and useful tool over the 45 others evaluated by [9], and those that have been and will be developed thereafter [147, 148], we argue that our real contribution for the particular case of CTI lies in Chapter 6. Indeed, by using our topological signatures (Corollary 6.3.5) for exploratory data analysis rather than topological inference in Section 6.4, we were able to explain the difficulties accompanied by CTI (which we observed in Section 5.8) on both a theoretical and practical level within a topological context. To the best of our knowledge, neither such an application of TDA, nor such an analysis within the field of CTI, had been considered before. Furthermore, we observed that the performance averaged over all CTI methods evaluated by [9] is consistent with the information captured through our topological signatures, allowing for a novel method for cell trajectory data quality control. This may eventually lead to new methods for evaluating the data prior to the actual trajectory inference, and for comparing and developing new representations (dimensionality reductions and/or proximity graphs) to improve the effectiveness of both our or other CTI methods. We recently investigated and discussed this in more detail in [6].

In a more general context beyond CTI, applications of topological inference *of* graphs *from* graphs are still unclear. E.g., the locations of earthquakes in Figure 5.23 can be straightforwardly visualized, which does not require the underlying model. Similarly, while connoisseurs may confirm that our method qualitatively provides effective models for the Harry Potter and Game of Thrones network, this is subject to opinion, and what we can use these models for remains vague. Yet, it is for these very reasons we included extensive experiments on a variety of different graphs, rising from artificial graphs to fields such as social networks, geology, and biology. This enabled us to thoroughly illustrate that we are able to provide effective graph-structured models in many fields of science, without any particular application currently in mind.

Similar to dimensionality reductions of point cloud data, the direct applications of inferring graph-structured models in graphs lie within topics such as exploratory data analysis, and visualization. Furthermore, as do dimensionality reductions find applications for improving machine learning models, e.g., by reducing overfitting, so may constrained graph-structured models lead to future applications for improving machine learning models on graphs, such as GCNs, graph embeddings, or classification models. As an example (which we also discussed above), [102] showed that simplified graphs obtained from the original graph—which are exactly the models we consider in a broader context—may prevent any existing graph embedding method from getting stuck in local minima during optimization. The applications of these various machine learning problems on graphs range from link prediction and recommender systems [149–151] (think of Netflix predicting which series you will like, Amazon which items might interest you, or Facebook your potential friends) to classification of graphs arising from social networks, toxicology, and protein functions or structures [39, 152–155].

In case of topological image modification and image processing, and in the particular case of skin lesion images, effective segmentations allow for automatic and more accurate predictions [117], and hence fast and tailored interventions when necessary. However, applications must not be restricted to biomedicine, or even have any social value at all. One may e.g. think of a cool new selection tool or filter in Photoshop, with only expressing your creativity as an application in mind.

Bibliography

[1] Robin Vandaele, Yvan Saeys, and Tijl De Bie. *Mining Topological Structure in Graphs through Forest Representations*. Journal of Machine Learning Research, 21(215):1–68, 2020.

[2] Robin Vandaele, Guillaume Adrien Nervo, and Olivier Gevaert. *Topological image modification for object detection and topological image processing of skin lesions*. Scientific Reports, 10(1):21061, 2020.

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

[4] Robin Vandaele, Yvan Saeys, and Tijl De Bie. *The Boundary Coefficient: a Vertex Measure for Visualizing and Finding Structure in Weighted Graphs*. In Proceedings of the 15th International Workshop on Mining and Learning with Graphs (MLG), 2019.

[5] Mira Bernstein, Vin De Silva, John C Langford, and Joshua B Tenenbaum. *Graph approximations to geodesics on embedded manifolds*. Technical report, Citeseer, 2000.

[6] Robin Vandaele. *Topological Data Analysis of Metric Graphs for Evaluating Cell Trajectory Data Representations*. Master's thesis, Ghent University, 2020.

[7] Andrea S LaPaugh and Ronald L Rivest. *The subgraph homeomorphism problem*. Journal of Computer and System Sciences, 20(2):133–149, 1980.

[8] Michael Joswig, Frank H Lutz, and Mimi Tsuruga. *Heuristics for sphere recognition*. In International Congress on Mathematical Software, pages 152–159. Springer, 2014.

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

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

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

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

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

[14] Gabor Csardi, Tamas Nepusz, et al. *The igraph software package for complex network research.* InterJournal, complex systems, 1695(5):1–9, 2006.

[15] Gunnar Carlsson. *Topology and data.* Bulletin of the American Mathematical Society, 46(2):255–308, jan 2009.

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

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

[18] Mathieu Carrière, Steve Y. Oudot, and Maks Ovsjanikov. *Stable Topological Signatures for Points on 3D Shapes.* Computer Graphics Forum, 34(5):1–12, 2015.

[19] Chad Giusti, Robert Ghrist, and Danielle S. Bassett. *Two's company, three (or more) is a simplex.* Journal of Computational Neuroscience, 41(1):1–14, Aug 2016.

[20] Tamal Krishna Dey, Sayan Mandal, and William Varcho. *Improved Image Classification using Topological Persistence.* In Matthias Hullin, Reinhard Klein, Thomas Schultz, and Angela Yao, editors, Vision, Modeling & Visualization. The Eurographics Association, 2017.

[21] Rabih Assaf, Alban Goupil, Valeriu Vrabie, Thomas Boudier, and Mohammad Kacim. *Persistent homology for object segmentation in multidimensional grayscale images*. Pattern Recognition Letters, 112:277 – 284, 2018.

[22] Kenneth Kunen. *Set theory an introduction to independence proofs*. Elsevier, 2014.

[23] M.A. Armstrong. *Basic Topology*. Undergraduate Texts in Mathematics. Springer New York, 2013.

[24] Ulrich Stelzl, Uwe Worm, Maciej Lalowski, Christian Haenig, Felix H Brembeck, Heike Goehler, Martin Stroedicke, Martina Zenkner, Anke Schoenherr, Susanne Koeppen, et al. *A human protein-protein interaction network: a resource for annotating the proteome*. Cell, 122(6):957–968, 2005.

[25] Michael Hecker, Sandro Lambeck, Susanne Toepfer, Eugene Van Someren, and Reinhard Guthke. *Gene regulatory network inference: data integration in dynamic models—a review*. Biosystems, 96(1):86–103, 2009.

[26] Xiaoming Liu, Johan Bollen, Michael L Nelson, and Herbert Van de Sompel. *Co-authorship networks in the digital library research community*. Information processing & management, 41(6):1462–1480, 2005.

[27] Stephen P Borgatti, Martin G Everett, and Jeffrey C Johnson. *Analyzing social networks*. Sage, 2018.

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

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

[30] Herbert Edelsbrunner. *Persistent Homology in Image Processing*. In Walter G. Kropatsch, Nicole M. Artner, Yll Haxhimusa, and Xiaoyi Jiang, editors, Graph-Based Representations in Pattern Recognition, pages 182–183, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[31] Korte Bernhard and Jens Vygen. *Combinatorial optimization: theory and algorithms*. Springer-Verlag Berlin Heidelberg, 2012.

[32] Wikipedia. *Unit disk graph — Wikipedia, The Free Encyclopedia*. `http://en.wikipedia.org/w/index.php?title=Unit%20disk%20graph&oldid=970510290`, 2020. [Online; accessed 27-November-2020].

[33] Brent N Clark, Charles J Colbourn, and David S Johnson. *Unit disk graphs*. In Annals of Discrete Mathematics, volume 48, pages 165–177. Elsevier, 1991.

[34] David Eppstein, Michael S Paterson, and F Frances Yao. *On nearest-neighbor graphs*. Discrete & Computational Geometry, 17(3):263–282, 1997.

[35] Timothy F Havel, Irwin D Kuntz, and Gordon M Crippen. *The combinatorial distance geometry method for the calculation of molecular conformation. I. A new approach to an old problem*. Journal of theoretical biology, 104(3):359–381, 1983.

[36] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. Applied Mathematics. American Mathematical Society, 2010.

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

[38] Christoph Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl. *Deep Learning with Topological Signatures*. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, pages 1633–1643, USA, 2017. Curran Associates Inc.

[39] Bastian Rieck, Christian Bock, and Karsten Borgwardt. *A Persistent Weisfeiler-Lehman Procedure for Graph Classification*. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, Proceedings of the 36th International Conference on Machine Learning, volume 97 of *Proceedings of Machine Learning Research*, pages 5448–5458, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[40] Steve Y. Oudot. *Persistence Theory: From Quiver Representations to Data Analysis*. Number 209 in Mathematical Surveys and Monographs. American Mathematical Society, 2015.

[41] Nikhil Singh, Heather D. Couture, J. S. Marron, Charles Perou, and Marc Niethammer. *Topological Descriptors of Histology Images*. In Guorong Wu, Daoqiang Zhang, and Luping Zhou, editors, Machine Learning in Medical Imaging, pages 231–239, Cham, 2014. Springer International Publishing.

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

[43] Kathryn Garside, Robin Henderson, Irina Makarenko, and Cristina Masoller. *Topological data analysis of high resolution diabetic retinopathy images*. PLOS ONE, 14(5):1–10, 05 2019.

[44] Allen Hatcher. *Algebraic topology*. Cambridge University Press, 2002.

[45] Wikipedia. *Simplicial complex — Wikipedia, The Free Encyclopedia*. `http://en.wikipedia.org/w/index.php?title=Simplicial%20complex&oldid=970115769`, 2020. [Online; accessed 24-November-2020].

[46] Kairui Glen Wang. *The basic theory of persistent homology*. `http://math.uchicago.edu/~may/REU2012/REUPapers/WangK.pdf`, 2012.

[47] Clément Maria. *Algorithms and data structures in computational topology*. Theses, Université Nice Sophia Antipolis, October 2014.

[48] Alexander Nabutovsky and Shmuel Weinberger. *Algorithmic aspects of homeomorphism problems*. Contemporary Mathematics, 231:245–250, 1999.

[49] Yngve Sundblad. *The Ackermann function. a theoretical, computational, and formula manipulative study*. BIT Numerical Mathematics, 11(1):107–119, Mar 1971.

[50] NJ Cavanna, M Jahanseir, and DR Sheehy. *A geometric perspective on sparse filtrations*. arXiv preprint arXiv:1506.03797, 2015.

[51] Gunnar Carlsson, Anjan Dwaraknath, and Bradley J Nelson. *Persistent and Zigzag Homology: A Matrix Factorization Viewpoint*. arXiv preprint arXiv:1911.10693, 2019.

[52] Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. *Geometry helps to compare persistence diagrams*. Journal of Experimental Algorithmics (JEA), 22:1–20, 2017.

[53] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. *Stability of persistence diagrams*. Discrete & computational geometry, 37(1):103–120, 2007.

[54] Fionn Murtagh and Pedro Contreras. *Algorithms for hierarchical clustering: an overview*. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2(1):86–97, 2012.

[55] Xu Liu, Zheng Xie, Dongyun Yi, et al. *A fast algorithm for constructing topological structure in large data.* Homology, Homotopy and Applications, 14(1):221–238, 2012.

[56] Tamal K Dey, Facundo Mémoli, and Yusu Wang. *Multiscale mapper: topological summarization via codomain covers.* In Proceedings of the twenty-seventh annual acm-siam symposium on discrete algorithms, pages 997–1013. SIAM, 2016.

[57] Dmitriy Morozov, Kenes Beketayev, and Gunther Weber. *Interleaving distance between merge trees.* Discrete and Computational Geometry, 49(22-45):52, 2013.

[58] Justine Jia Wen Seow, Regina Men Men Wong, Rhea Pai, and Ankur Sharma. *Single-Cell RNA Sequencing for Precision Oncology: Current State-of-Art.* Journal of the Indian Institute of Science, page 1, 2020.

[59] Ankur Sharma. *Hiding in plain sight: epigenetic plasticity in drug-induced tumor evolution.* Epigenetics Insights, 12:2516865719870760, 2019.

[60] Ankur Sharma and Ramanuj DasGupta. *Tracking tumor evolution one-cell-at-a-time.* Molecular & cellular oncology, 6(3):1590089, 2019.

[61] Ankur Sharma, Elaine Yiqun Cao, Vibhor Kumar, Xiaoqian Zhang, Hui Sun Leong, Angeline Mei Lin Wong, Neeraja Ramakrishnan, Muhammad Hakimullah, Hui Min Vivian Teo, Fui Teen Chong, et al. *Longitudinal single-cell RNA sequencing of patient-derived primary cells reveals drug-induced infidelity in stem cell hierarchy.* Nature communications, 9(1):1–17, 2018.

[62] Daniel J Kunz, Tomás Gomes, and Kylie R James. *Immune cell dynamics unfolded by single-cell technologies.* Frontiers in immunology, 9:1435, 2018.

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

[64] James Eberwine, Jai-Yoon Sul, Tamas Bartfai, and Junhyong Kim. *The promise of single-cell sequencing.* Nature methods, 11(1):25–27, 2014.

[65] Robrecht Cannoodt, Wouter Saelens, Helena Todorov, and Yvan Saeys. *Single-cell -omics datasets containing a trajectory*, October 2018.

[66] Ronald R Coifman, Stephane Lafon, Ann B Lee, Mauro Maggioni, Boaz Nadler, Frederick Warner, and Steven W Zucker. *Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps*. Proceedings of the national academy of sciences, 102(21):7426–7431, 2005.

[67] Daniel Marbach, James C Costello, Robert Küffner, Nicole M Vega, Robert J Prill, Diogo M Camacho, Kyle R Allison, Manolis Kellis, James J Collins, and Gustavo Stolovitzky. *Wisdom of crowds for robust gene network inference*. Nature methods, 9(8):796–804, 2012.

[68] Béla Bollobás. *Random graphs*. In Modern graph theory, pages 215–252. Springer, 1998.

[69] Alan Frieze and Michał Karoński. *Introduction to random graphs*. Cambridge University Press, 2016.

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

[71] Paul Bendich, Bei Wang, and Sayan Mukherjee. *Local homology transfer and stratification learning*. In Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms, pages 1355–1370. SIAM, 2012.

[72] Paul Bendich, David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Dmitriy Morozov. *Inferring local homology from sampled stratified spaces*. In 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), pages 536–546. IEEE, 2007.

[73] Primoz Skraba and Bei Wang. *Approximating local homology from samples*. In Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms, pages 174–192. SIAM, 2014.

[74] Paul Bendich. *Analyzing stratified spaces using persistent versions of intersection and local homology*. Duke University, 2008.

[75] Timothy M Chan. *All-pairs shortest paths for unweighted undirected graphs in o (mn) time*. ACM Transactions on Algorithms (TALG), 8(4):1–17, 2012.

[76] Lior Rokach and Oded Maimon. *Clustering methods*. In Data mining and knowledge discovery handbook, pages 321–352. Springer, 2005.

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

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

[79] Louis L McQuitty. *Hierarchical linkage analysis for the isolation of types.* Educational and Psychological Measurement, 20(1):55–67, 1960.

[80] Bastian Rieck and Heike Leitte. *Agreement Analysis of Quality Measures for Dimensionality Reduction.* In Hamish Carr, Christoph Garth, and Tino Weinkauf, editors, Topological Methods in Data Analysis and Visualization IV, pages 103–117, Cham, 2017. Springer International Publishing.

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

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

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

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

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

[86] Bernard Chazelle. *A Minimum Spanning Tree Algorithm with inverse-Ackermann Type Complexity.* J. ACM, 47(6):1028–1047, November 2000.

[87] Ryuhei Uehara and Yushi Uno. *Efficient Algorithms for the Longest Path Problem.* In Rudolf Fleischer and Gerhard Trippen, editors, Algorithms and Computation, pages 871–883, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[88] Andreas Krause and Daniel Golovin. *Submodular Function Maximization.* Tractability, 3:71–104, 01 2011.

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

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

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

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

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

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

[95] T.G. Crainic and G. Laporte. *Fleet Management and Logistics*. Centre for Research on Transportation. Springer US, 1998.

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

[97] Y.P. Aneja and K.P.K. Nair. *Location Of A Tree Shaped Facility In A Network*. INFOR: Information Systems and Operational Research, 30(4):319–324, 1992.

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

[99] Yingpeng Hu, Kaixi Zhang, Jing Yang, and Yanghui Wu. *Application of Hierarchical Facility Location-Routing Problem with Optimization of an Underground Logistic System: A Case Study in China*. Mathematical Problems in Engineering, 2018:1–10, 09 2018.

[100] Pasquale Avella, Maurizio Boccia, Antonio Sforza, and Igor Vasil'Ev. *A Branch-and-Cut Algorithm for the Median-Path Problem*. Computational Optimization and Applications, 32(3):215–230, Dec 2005.

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

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

[103] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. *A Global Geometric Framework for Nonlinear Dimensionality Reduction.* Science, 290(5500):2319–2323, 2000.

[104] J. Hartigan. *The K-means algorithm.* Clustering algorithms, 4, 1975.

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

[106] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. *On the Surprising Behavior of Distance Metrics in High Dimensional Space.* In Jan Van den Bussche and Victor Vianu, editors, Database Theory — ICDT 2001, pages 420–434, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[107] Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. *Nearest neighbors in high-dimensional data: The emergence and influence of hubs.* In Proceedings of the 26th Annual International Conference on Machine Learning, pages 865–872, 2009.

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

[109] F. Fouss, A. Pirotte, J. Renders, and M. Saerens. *Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation.* IEEE Transactions on Knowledge and Data Engineering, 19(3):355–369, March 2007.

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

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

[112] Mathieu Carriere, Steve Oudot, and Maks Ovsjanikov. *Local Signatures using Persistence Diagrams.* working paper or preprint, June 2015.

[113] Frédéric Chazal, David Cohen-Steiner, Leonidas J Guibas, Facundo Mémoli, and Steve Y Oudot. *Gromov-Hausdorff stable signatures for shapes using persistence.* In Computer Graphics Forum, volume 28, pages 1393–1403. Wiley Online Library, 2009.

[114] Victor Patrangenaru and Leif Ellingson. *Nonparametric Statistics on Manifolds and Their Applications to Object Data Analysis.* CRC Press, Inc., USA, 1st edition, 2015.

[115] Elena Farahbakhsh Touli and Yusu Wang. *FPT-algorithms for computing Gromov-Hausdorff and interleaving distances between trees.* arXiv preprint arXiv:1811.02425, 2018.

[116] Kevin Buchin and Wolfgang Mulzer. *Delaunay triangulations in O (sort (n)) time and more.* Journal of the ACM (JACM), 58(2):1–27, 2011.

[117] Yu-Min Chung, Chuan-Shen Hu, Austin Lawson, and Clifford Smyth. *TopoResNet: A hybrid deep learning architecture and its application to skin lesion classification.* arXiv preprint arXiv:1905.08607, 2019.

[118] S. Paris and F. Durand. *A Topological Approach to Hierarchical Segmentation using Mean Shift.* In 2007 IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8, 2007.

[119] Hengrui Luo, Justin Strait, and Abhijoy Saha. *Combining Geometric and Topological Information in Image Segmentation.* arXiv preprint arXiv:1910.04778, 2019.

[120] Roberto A Novoa, Olivier Gevaert, and Justin M Ko. *Marking the Path Toward Artificial Intelligence–Based Image Classification in Dermatology.* Jama Dermatology, 155(10):1105–1106, 2019.

[121] D. I. Schlessinger, G. Chhor, O. Gevaert, S. M. Swetter, J. Ko, and R. A. Novoa. *Artificial intelligence and dermatology: opportunities, challenges, and future directions.* Semin Cutan Med Surg, 38(1):31–37, Mar 2019.

[122] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation.* In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, pages 234–241, Cham, 2015. Springer International Publishing.

[123] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. *Mask R-CNN.* In The IEEE International Conference on Computer Vision (ICCV), Oct 2017.

[124] Tony Chan and Luminita Vese. *An Active Contour Model without Edges.* In Mads Nielsen, Peter Johansen, Ole Fogh Olsen, and Joachim Weickert, editors, Scale-Space Theories in Computer Vision, pages 141–151, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[125] F. R. D. Velasco. *Thresholding using the ISODATA clustering algorithm.* IEEE Transactions on Systems Man and Cybernetics, 10:771–774, November 1980.

[126] William E. Lorensen and Harvey E. Cline. *Marching Cubes: A High Resolution 3D Surface Construction Algorithm.* In Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH ’87, page 163–169, New York, NY, USA, 1987. Association for Computing Machinery.

[127] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. *SLIC Superpixels Compared to State-of-the-Art Superpixel Methods.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(11):2274–2282, 2012.

[128] Lawrence G Roberts. *Machine perception of three-dimensional solids.* PhD thesis, Massachusetts Institute of Technology, 1963.

[129] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. *Snakes: Active contour models.* International Journal Of Computer Vision, 1(4):321–331, 1988.

[130] Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. *Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic).* arXiv preprint arXiv:1902.03368, 2019.

[131] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. *The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions.* Scientific data, 5:180161, 2018.

[132] Christopher Tralie, Nathaniel Saul, and Rann Bar-On. *Ripser.py: A Lean Persistent Homology Library for Python.* The Journal of Open Source Software, 3(29):925, Sep 2018.

[133] Frédéric Chazal, Leonidas J. Guibas, Steve Oudot, and Primoz Skraba. *Persistence-Based Clustering in Riemannian Manifolds.* In ACM Annual Symposium on Computational Geometry, pages 97–106, Paris, France, June 2011.

[134] J.C. Dooley. *Two-dimensional interpolation of irregularly spaced data using polynomial splines.* Physics of the Earth and Planetary Interiors, 12(2):180 – 187, 1976.

[135] Lee R Dice. *Measures of the amount of ecologic association between species.* Ecology, 26(3):297–302, 1945.

[136] Th A Sorensen. *A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons.* Biol. Skar., 5:1–34, 1948.

[137] Brian W Matthews. *Comparison of the predicted and observed secondary structure of T4 phage lysozyme.* Biochimica et Biophysica Acta (BBA)-Protein Structure, 405(2):442–451, 1975.

[138] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. *LINE: Large-scale Information Network Embedding.* In Proceedings of the 24th International Conference on World Wide Web, WWW '15, pages 1067–1077, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

[139] Surender Baswana and Telikepalli Kavitha. *Faster algorithms for all-pairs approximate shortest paths in undirected graphs.* SIAM Journal on Computing, 39(7):2865–2896, 2010.

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

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

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

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

[144] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. *Imagenet large scale visual recognition challenge.* International journal of computer vision, 115(3):211–252, 2015.

[145] Aviv Regev, Sarah A Teichmann, Eric S Lander, Ido Amit, Christophe Benoist, Ewan Birney, Bernd Bodenmiller, Peter Campbell, Piero Carninci, Menna Clatworthy, et al. *Science forum: the human cell atlas.* Elife, 6:e27041, 2017.

[146] Tabula Muris Consortium et al. *Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris.* Nature, 562(7727):367, 2018.

[147] Thinh N Tran and Gary D Bader. *Tempora: cell trajectory inference using time-series single-cell RNA sequencing data.* PLOS Computational Biology, 16(9):e1008205, 2020.

[148] Helena Todorov, Robrecht Cannoodt, Wouter Saelens, and Yvan Saeys. *TinGa: fast and flexible trajectory inference with Growing Neural Gas.* Bioinformatics, 36(Supplement_1):i66–i74, 2020.

[149] László Grad-Gyenge, Attila Kiss, and Peter Filzmoser. *Graph embedding based recommendation techniques on the knowledge graph.* In Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization, pages 354–359, 2017.

[150] Jun Ai, Yayun Liu, Zhan Su, Hui Zhang, and Fengyu Zhao. *Link prediction in recommender systems based on multi-factor network modeling and community detection.* EPL (Europhysics Letters), 126(3):38003, 2019.

[151] Seyed Mehran Kazemi and David Poole. *Simple embedding for link prediction in knowledge graphs.* In Advances in neural information processing systems, pages 4284–4295, 2018.

[152] Christoph Helma, Ross D. King, Stefan Kramer, and Ashwin Srinivasan. *The predictive toxicology challenge 2000–2001.* Bioinformatics, 17(1):107–108, 2001.

[153] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. *Protein function prediction via graph kernels.* Bioinformatics, 21(suppl_1):i47–i56, 2005.

[154] Pinar Yanardag and SVN Vishwanathan. *Deep graph kernels.* In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1365–1374, 2015.

[155] Paul D Dobson and Andrew J Doig. *Distinguishing enzyme structures from non-enzymes without alignments.* Journal of molecular biology, 330(4):771–783, 2003.