Title
Extracting and visualizing topological information from large high-dimensional data sets

Author
Beketayev, Kenes

Publication Date
2014-05-08

Peer reviewed

Extracting and Visualizing Topological Information from Large High-dimensional Data Sets

By

KENES BEKETAYEV
B.S. (Northwestern Polytechnic University) 2007
Eng.D. (Kazakh-British Technical University) 2008

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Bernd Hamann, Chair

Gunther H. Weber

Zhaojun Bai

Committee in Charge
2013

Extracting and Visualizing Topological Information from Large High-dimensional Data Sets

To my parents

Baidaulet and Bayan

Acknowledgments

I am enormously grateful to my advisor, Professor Bernd Hamann, for guiding me through the hurdles of the graduate life with a great patience and enthusiasm. His advising style and professional approach impressed me from the start and inspired me every step of the way. I will always remember Thanksgiving dinners hosted by Bernd and Cathryn, as they gave me glimpses of a warmth and comfort that only home can provide.

Next, I wish to thank Dr. Gunther H. Weber, whose kind and patient insistence continually pushed me beyond my comfort zone to places I would have never reached on my own. His ability to perform ginormous amounts of work will always serve as a benchmark against which I will measure myself. I am also grateful to Professor Zhaojun Bai for his presence on my dissertation committee. Undoubtedly, I learned a great deal from him and his course on scientific computing, and I thoroughly enjoyed our interesting yet quite coincidental chats over the years.

I thank Professors Kenneth I. Joy and John D. Owens for their generous provision of knowledge and advice, and their invaluable comments that helped me to improve my dissertation. I thank Professor Phillip Rogaway for the academic advising, and Jessica Stoller for always being there, when you have questions. I thank professors from whom I had the good fortune to receive an instruction, as they taught me plenty and more.

I am grateful to the staff at Lawrence Berkeley National Laboratory, especially Dr. Dmitriy Morozov, Dr. Maciej Haranczyk, and Dr. Damir Yeliussizov, with whom I collaborated extensively and fruitfully, and Dr. E. Wes Bethel, who made it possible for me to join the Visualization group, and gain exposure to exciting frontiers of science. To others, Dr. Jörg Meyer, Dr. Hank Childs, Dr. Prabhat, Dr. Daniela Ushizima, Dr. Harinarayan Krishnan, Dr. Burlen Loring, Dr. Oliver Rübel, Rachel Lance, Barbara Salisbury, Dr. Richard Martin, Dr. Hasan Metin Aktulga, I say thank you for your support, advice, and conversations. It was a pleasure to work with all of you.

I want to personally thank Dr. Horst D. Simon for advising me and helping me through some

rough times during my graduate studies. I am also grateful to Dr. Mervyn Wong, who got me involved in research projects at LBNL and later encouraged me to enter a graduate program, same as Dr. Aleks Göllü and Dr. Nuri Dagdeviren, who helped me to get started. I want to express my personal appreciation to Dr. Kanat Baigarin and Mr. Aslan Sarinzhipov, for supporting my endeavors and for providing me with incredible opportunities.

As my good friend Xiang puts it, a big shout out goes out to him, Jishang, Matthew, Stanley, Thomas, Brian, Gabriel, Andrew, Anjul, Shubho, Yuan, and Jianqiang for all the fun times and interactions that kept me sane.

Dr. Zhanybek Alpichshev, Dr. Igor Ganichev, Ruslan Kenzhebekov, Olzhas Makhambetov, Medet Nokhatov, and Shamil Arsunukayev; you guys helped me more than you could possibly imagine, and I would not be writing these lines without you. Rasim abi, Alibek A., Alibek B., Zhanibek K., Yasin, Adilbek, Bakhytzhan, Ivan, Gaziz, Aidos A., Nadya, Dinara, and many, many others, thank you.

Finally, the love and perseverance of my parents, Baidaulet Beketayev and Bayan Sakhipova, and their belief in me, has allowed me to be where I am today. And my gratitude goes out to my brother Serik, who contributed substantially by forcing me to aspire to be a better model for him. For all of this I am extremely grateful, and I love them endlessly. Grandpa Zarkesh Sakipov, thanks for inspiring me to do science, and grandma, I miss you. Karlygash, Nurlan, Guldana, Serzhan, Zhamilya, Bekzar, you are all in my heart.

My work was supported by the Director of the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 (Lawrence Berkeley National Laboratory), by the Department of Computer Science and the Office of Graduate Studies, of the University of California at Davis, by the National Scientific Foundation through grant CCF-0702817, and by Program 055 of the Republic of Kazakhstan's Ministry of Education and Science under the contract with the Center for Energy Research, Nazarbayev University.

Contents

List of Figures viii

List of Acronyms xiii

Abstract xiv

1 Introduction 1

2 Background 4
- 2.1 Computational Topology . . . 4
  - 2.1.1 Basic Definitions . . . 5
  - 2.1.2 Morse Theory . . . 5
  - 2.1.3 Level Set Topology . . . 6
  - 2.1.4 Cell Complexes . . . 6
  - 2.1.5 Structures for Point Sets . . . 7
  - 2.1.6 Persistent Homology . . . 8
- 2.2 Topological Structures . . . 9
  - 2.2.1 Reeb Graph . . . 9
  - 2.2.2 Morse-Smale Complex . . . 11
  - 2.2.3 Simplification . . . 12
  - 2.2.4 Persistence Diagram . . . 12
- 2.3 Topology-based Visualization . . . 13
  - 2.3.1 Graph-based Representations . . . 13
  - 2.3.2 Visualization Metaphors . . . 14

3 Topology–based Analysis of Transformation Pathways in Chemical Systems 15
- 3.1 Introduction . . . 15
- 3.2 Contributions . . . 17
- 3.3 Related Work in Chemistry . . . 17
- 3.4 Algorithm . . . 18
  - 3.4.1 Morse Complex . . . 20

3.4.2 Simplification . . . 20
3.4.3 Periodicity . . . 22
3.4.4 Graph Layout . . . 22
3.5 Results . . . 24
3.5.1 Dimer of Formic Acid and Acetic Acid . . . 25
3.5.2 Free Energy of a Guest Molecule in a Porous Material . . . 30

4 Comparative Geometry–preserving Topological Landscapes 35
4.1 Introduction . . . 35
4.2 Contributions . . . 37
4.3 Related Work . . . 37
4.3.1 Contour Tree Visualization . . . 37
4.3.2 Dimension Reduction . . . 38
4.3.3 Graph Drawing . . . 38
4.4 Algorithm . . . 39
4.4.1 Contour Tree Computation . . . 40
4.4.2 Contour Tree Layout . . . 40
4.4.3 Landscape Construction . . . 41
4.4.4 Comparable Landscapes . . . 44
4.5 Results . . . 45

5 Measuring Distance between Merge Trees 48
5.1 Introduction . . . 48
5.2 Contributions . . . 51
5.3 Related Work . . . 51
5.3.1 Persistence Diagrams . . . 51
5.3.2 Distance between Graphs . . . 52
5.3.3 Using Topology of Real Functions for Shape Matching . . . 52
5.4 Distance between Merge Trees . . . 53
5.4.1 Definition . . . 53
5.5 Naïve Algorithm . . . 56
5.6 Optimized Algorithm with Memoization . . . 57
5.7 Number of Branch Decompositions of a Merge Tree . . . 60

6 Measuring Error in Approximating Sublevel Set Topology 62
6.1 Introduction . . . 62
6.2 Contributions . . . 66
6.3 Related Work . . . 67
6.3.1 Sampling and Mesh Types . . . 67
6.3.2 Evaluation of Mesh Types . . . 68
6.4 General Evaluation . . . 69
6.4.1 Ground Truth Function . . . 70
6.4.2 Measure Normalization . . . 71

6.4.3 Evaluation and Analysis 72
6.4.4 Nonlinear Manifolds 79
6.4.5 Evaluation Summary 82
6.5 Evaluation of Proposed Topology-based Techniques 83

7 Conclusions **85**

References **88**

List of Figures

2.1 2D critical points: minimum, saddle, maximum. (Image courtesy of Attila Gyulassy) 6

2.2 Left: The cover is a union of ε-balls, the nerve is the Čech complex. Right: The cover is Voronoi diagram, the nerve is Delaunay complex. 8

2.3 The Reeb graph of a torus. Left: Critical points correspond to changes in the topology of contours. For example, the minimum corresponds to the birth of a component of a contour. The lower saddle corresponds to the split of a contour component into two, etc. Right: The Reeb graph, which in addition to critical points, encodes the information about the evolution of components of each contour, thus serving as a topological skeleton of the surface. (Image courtesy of Gunther H. Weber [82]) 9

2.4 Contour tree records (a) birth, (b) merger, (f) split, and death of contour components in a simply connected domain. (Image courtesy of Hamish Carr [21]) . . . 10

2.5 Morse-Smale complex. Given a 2D manifold with maxima colored by red, minima colored by blue, and saddles colored by green colors, the middle left image shows a stable segmentation of the manifold with corresponding Morse complex, while the middle right image shows an unstable segmentation of the manifold with corresponding Morse complex. The image on the right is a Morse-Smale complex, obtained by combining stable and unstable Morse complexes. (Image courtesy of Attila Gyulassy) 11

3.1 A 2D graph showing the topology of the 3D potential energy function of a complex of formic and acetic acids, which depends on the positions of constituting atoms. Blue and red dots represent minimum energy configurations and lowest barriers connecting neighboring minima, respectively. Edges represent the energy cost of a particular transformation, with darker and wider edges corresponding to transformations through lower barriers (more likely transformations). Two vertical branches corresponding to different positions of protons are visible on the left and right side of the graph. Energy barriers for transforming the right branch into left one are lower than for the reverse transformation. 16

3.2 Left: Chemical system configuration of a dimer of formic and acetic acid. Middle left: Energy of the system in Hartrees (y-axis) as a function of the methyl group rotation angle between 0 to 360 degrees (x-axis). Blue and red dots mark energy minima and barriers, respectively. Middle right: Naive graph representation of transformation pathways. Right: Proposed graph representation, where the top row of a energy minima label displays its coordinate and the bottom row its energy value. 19
3.3 Energy functions of DFA along its four dimensions (top), and respective graphs for the third dimension (bottom left), showing the periodic nature of the dimension, and (bottom right) the fourth dimension, where it correctly handles the special case of periodicity. Labels for latter two use the convention from Figure 3.2. 24
3.4 Two-dimensional energy function landscapes of DFA and corresponding graph: (a) $F_{12} - M_{12}$ (b) $F_{13} - M_{13}$ (c) $F_{14} - M_{14}$ (d) $-M_{34}$. The graph representation uses the same labeling as the one used in the previous figures. 28
3.5 Three-dimensional energy function volume renderings of DFA and corresponding graphs: (a) $F_{123} - M_{123}$ (b) $F_{124} - M_{124}$. 29
3.6 Graph of the 4D energy function of DFA. Nodes are labeled A1 to A9 and B1 to B6 to highlight two subgraphs. A1-A9 correspond to minima with coordinates of 0.95Å along the first and second dimension. The remaining third and fourth coordinates are (45,345), (180,345), (60,180), (285,345), (180,180), (60,0), (300,180), (180,0), and (300,0) for A1 to A9, respectively. Similarly, the first and second coordinate for B1-B6 are 1.75Å. The remaining coordinates are (60,345), (180,345), (300,345), (60,0), (180,0), (300,0) for B1 to B6, respectively. 30
3.7 Picture of (A) methane, (B) LTA zeolite structure. The orange isosurface highlights the closest distance to which the center of the guest molecule center can approach. The large cage is located in the center of the unit cell. The small cage is shared among eight cells and is visible in the corners of the unit cell. 31
3.8 Persistence diagram of the energy function of a CH_4 guest molecule in porous material. The large drop in number of minima (y-axis) hints the optimal threshold value for simplification, in this particular case around 0.005. 32
3.9 Top: Volume rendering of the energy function of a CH_4 molecule in an LTA zeolite and lowest energy paths connecting neighboring minima. Bottom: Corresponding graph showing lowest energy paths (an edge going through a face of the periodic box marked with arrows). 33

4.1 **The contour tree layout process.** First the vertices are projected as a point set. Then, the edges are drawn iteratively by connecting the corresponding points. The initial three edges $(f,d),(c,d)$, and (d,b), are drawn as a straight line. Then, the edge (e,b) is routed via Voronoi diagram. Finally, the edge (b,a) is drawn through intersections of Voronoi diagram and bounding box, since no path through Voronoi diagram itself is available. 40
4.2 **Contour map construction process.** The process starts from a root branch (f,a). It has children, so they are sorted in descending manner, and the process continues with the child branch (c,d), as it corresponds to the highest saddle d. The upper (processed) section of this branch is an edge (f,d), so the triangle contour is used. Then, its child branch (c,d) is a leaf, so the triangle contour is used again. However, the upper section for the next saddle b is complex, so the offset contour is used. Next, the corresponding child branch (e,b) is a leaf, so the triangle contour is used. Finally, the offset contour is applied to the branch (f,a), and that concludes the contour map construction process. 42
4.3 **Triangle contour.** Constructing a triangle contour (black) for the branch (c,d) creates a valley with inner contours (red) corresponding to the given branch. . . . 43
4.4 **Offset contour.** Constructing an offset contour (black) around the upper section of the saddle b creates a continuation of the peak f with the valley c on it. 44
4.5 **Trilinear ray sampling option data set.** Volume rendering (top) and the corresponding geometry-preserving topological landscape (bottom). 45
4.6 **Geometry-preserving landscapes.** Top: Trilinear ray sampling option data set, for which poor performing configurations are closely located (circle A). Bottom: Nearest-neighbor ray sampling option data set, for which poor performing configurations are spread around (circles A, B). From landscapes, one can easily see how close/far the poor performing configurations are from each other. To derive the same observation from the direct view, one has to search for specific, often different, isosurfaces that would encompass such configurations. Furthermore, this search might be impossible when the dimensionality of the function exceeds three. 47

5.1 Consider two scalar functions (left), one of which is a slightly shifted version of the other. Comparing them directly, e.g., via L_∞ norm, results in a large difference. On the other hand, their persistence diagrams are the same (right), thus capturing the topological similarity of these functions. 49
5.2 Consider two "close" scalar functions, shown on the left, where one contains additional noise. If to construct their persistence diagrams and find the bottleneck distance (which corresponds to the longest black line segment between paired points on the right), the result is small, correctly reflecting the closeness of the functions. In fact, the difference is the same as the level of the noise, which in this example is small. 49

5.3 Consider two scalar functions on the left. The bottleneck distance between persistence diagrams on the right equals zero, as points of two diagrams overlap. However, comparing corresponding merge trees in the middle reveals a difference, since they cannot be matched exactly. This difference highlights an existence of additional nesting information in merge trees. Quantifying it is the main goal of this chapter. 50
5.4 Merge trees T_f (top) and T_g (bottom), all their possible branch decompositions, and corresponding rooted tree representations. Root branches are colored in red, demonstrating the mapping of branches to vertices. 54
5.5 As long as s_r is paired with a minimum that is outside of the subtree T_{s_i}, the set of branch decompositions of that subtree remains the same. 57
5.6 Splitting the higher degree (> 2) saddle always creates more branch decompositions. For example, the saddle s induces three branch decompositions, however after being split into two saddles s^1 and s^2, now there are four branch decompositions. 61

6.1 (Left) Molecule of dimer of formic and acetic acid. (Right) Rotation of its methyl group produces the potential energy function. 63
6.2 Although the first approximated function (green) deviates from the real function (blue), it still preserves the correct number of minima — three — thus bearing no error on the count of minima. However, the second approximated function (red) contains only one minimum, leading to an error of $2/3$. 64
6.3 (Left) Given a set of nine samples of a two-dimensional function, one can construct various meshes with potentially different approximation error. For example, following meshes are generated by (middle left) Gabriel graph, (middle right) Delaunay triangulation, (right) K-Nearest-Neighbors, with $K = 2$, the lowest value for which the mesh becomes connected. 65
6.4 The edge between any two points p and q exists, if their "region" is empty, i.e., free of other points. Left: For a Gabriel graph, the "region" is a circle with an edge (p,q) as a diameter. Right: For a relative neighborhood graph, the "region" is an intersection of two half-circles, with centers at p and q, and a radius (p,q). . 68
6.5 The ground truth function is indicated by the bold black line, a supremum of maxima generators. Red nodes are maxima and blue nodes are possible saddles. Note that saddle $s_{3,4}$ belongs to the given function, while $s_{2,4}$ do not. A function value and coordinates of any saddle can be explicitly computed from the information about the maxima (dotted lines). 71
6.6 The increase of the number of samples leads to the decrease of error values of $\varepsilon_B, \varepsilon_M$. 73

6.7 The error measure ε_B for functions with varying number of maxima. As expected, more maxima slow the error decrease rate, as it takes more samples to discover all maxima. Indeed, for max3 function (left), the error drops below 0.1 around 400 samples, for max20 function (middle) around 1000 samples, for max50 function (right) around 3000 samples. Note an irregular increase of the error for the DEL mesh, between 2100 and 6900 samples. 74
6.8 Increase in dimensionality does not affect the expected decreasing error trend. Again, the performance of the DEL mesh is relatively bad. 75
6.9 Detailed investigation of an irregular increase of the error in Figure 6.4.3. The DEL mesh before (left) and after (middle) the error drop around 6900 samples. The DEL mesh before the error drop has many, potentially over-connecting, long edges along the boundaries of the domain. The error drop happens, as some of them are reduced due to the newly inserted samples. For comparison, the RNG mesh before the error drop is provided (right), showing that it is not affected by the problem of over-connecting boundary edges. 76
6.10 Evaluation of the relaxed versions of the GAB and RNG meshes shows no significant improvement in terms of the error. 77
6.11 The minimally connected KNN and VR meshes compared to the RNG mesh. The minimally connected VR mesh performs better, as it approximates the function better. 78
6.12 The average number of neighbors for the minimally connected VR mesh in Figure 6.11 equals 11. If to set the number of neighbors for the KNN mesh to 11, and compare it to the minimally connected KNN and VR meshes, the performance is significantly more stable. 79
6.13 The sweep of the parameter k versus the number of samples. The deep blue area on the right side corresponds to the values of k, for which mesh is disconnected, thus no results are available. Almost in all cases, the error decreases and then increases, with growing k value. 80
6.14 The evaluation with the underlying manifold changed from a square $(x,y,z=0)$ in 2D Euclidean space (top left) to the paraboloid $(x,y,z=-x^2-y^2)$ embedded in 3D space (top right); and to the torus $(x,y,z=\pm\sqrt{r^2-(R-\sqrt{x^2+y^2})^2})$ embedded in 3D space (bottom). In all cases, the following parameter values $\{x=y=[0,150],z=[0,75],r=25,R=50\}$ are used. The ground truth function, shown in colors (red indicating high function values, blue indicating low function values), stays the same in all cases. 81
6.15 Performance of the DEL, GAB, and RNG meshes on curved square manifolds, namely on the paraboloid (top two), and on the torus (bottom two) manifolds. The ground truth function is the same as the one used in Figure 6.6. 82
6.16 Performance of the DEL, GAB, and RNG meshes on curved (paraboloid) cube manifold. 83

List of Acronyms

DEL	Delaunay complex (or Delaunay triangulation, depending on context)
GAB	Gabriel graph
KNN	K-Nearest-Neighbors
RNG	Relative Neighborhood Graph
VR	Vietoris-Rips complex

Abstract

This doctoral dissertation explores and advances topology-based data analysis and visualization, a field that concerns itself with creating tools for gaining insights from scientific data, thus supporting the process of scientific knowledge discovery. In particular, the study proposes two novel analytical techniques, inspired by domain-specific problems and presents a study of error of approximation for these techniques.

The first part of the dissertation focuses on a specific problem that arises in computational chemistry. Analysis of transformation pathways is a well-known tool for the investigation of chemical systems and has implications for the design of chemical reactions and materials. However, existing techniques for analyzing transformation pathways either lack the required level of detail for such analyses, or they are limited to low-dimensional data. These issues, complicated by the noise in data and by issues of handling periodic boundaries, are addressed by a novel technique, which involves the extraction of a topological structure, the "Morse complex," and visualizing it as a graph, augmented with additional information, enabling the desired end-user analysis. The technique is then successfully applied to the analyses of two different types of chemical data, which demonstrates its utility.

The second part of the dissertation concentrates on the problem of enabling the comparison of data sets in terms of their topologies. In particular, the focus is on enabling the comparison between different instances of the same topological structure, namely a contour tree. One possible solution to this problem is to correlate contour trees in terms of the geometric proximity of their critical points. In order to visualize this correlation, a novel technique combines the extraction of the contour trees, dimensionality reduction, graph drawing, and contours construction. The technique produces a visual metaphor called a "geometry-preserving topological landscape." The utility of the technique is demonstrated through a comparative analysis of data sets based on their corresponding landscapes.

The remainder of the dissertation is dedicated to studying the problem of error quantification for the proposed techniques, as well as for more general settings. In particular, the focus is on approximation methods used to reconstruct a domain. For example, by studying the ability of these techniques to preserve topological information, one can derive method selection recommendations, which are potentially generalizable to various topological data analysis techniques. To address this problem, a novel definition of a difference

measure for topological abstraction, the merge tree, is presented and subsequently used to evaluate the previously mentioned approximation methods. The resulting recommendations are found to support the selection of approximation methods for the two proposed techniques.

Chapter 1

Introduction

The field of scientific data analysis remains a hotbed for research activity, in many instances as the result of challenges imposed by the increasing size and dimensionality of scientific data. These challenges impede the effective analysis, crucial for scientific knowledge discovery. Overcoming these challenges, often within specific settings, continues to be of utmost importance, and it is therefore, the focus of this dissertation.

In particular, this work focuses on a subarea of scientific data analysis, concerned with a topology-based approach to data analysis and visualization, the background for which is reviewed in greater details in Chapter 2. This approach utilizes the identification and abstraction of topological information from data, e.g., underlying topological structures and features, and allows insight into the systems and phenomena to which this data corresponds. Combined with intuitive visualization, topological information serves as a powerful data analysis tool in many domain sciences, such as chemistry, physics, etc., thus helping researchers in many instances to overcome the impositions of dimensionality and size.

Within the past decade, numerous methods for extraction and visualization of topological information have been proposed. As a result of the parallel advances in our ability to produce larger and more complex data sets, an even greater number of problems have emerged accompanied by

their unique challenges. Thus, to narrow the scope of this dissertation, the first portion of research work is motivated by problems arising from the application studies in different domain sciences. The rationale is straightforward. First, the approach guarantees that the research work is applicable in the real world. Second, creating and improving the theory and methodology of scientific data analysis should have a real impact on domain sciences, and, consequently, it must be guided by these same sciences. Finally, the ability to demonstrate the correctness and usefulness of the produced results serves as a good check for the sanity of such research.

In particular, two problems are considered. In Chapter 3, the problem of analyzing transformation pathways in chemical systems is addressed through combination of topological information extraction, dimension reduction, and visualization methods, resulting in a new technique that enables the desired analysis. In Chapter 4, the problem of enabling comparative analysis is addressed by preserving spatial correlations between interesting configurations (corresponding to extrema in data sets) that are visualized according to a two-dimensional terrain metaphor named the geometry-preserving topological landscape. While this technique is presented in a general context, the motivation for it is demonstrated by the application example that focuses on the comparative performance analysis of a ray casting algorithm.

The remainder of the dissertation is dedicated to quantifying the errors arising from the approximation of topological information. Although the two techniques previously presented solve different problems, both techniques are provided with discrete sets of points as input, and subsequently require a combination of methods (e.g., for mesh construction, interpolation, etc.) for their approximations of the underlying function and topology. One of the main problems that arise in this setting is the study of the introduced approximation error, which is usually left unaddressed. The proposed solution to this problem is two-fold. In Chapter 5, a more generic problem of measuring topological difference/similarity of functions is addressed by introducing a novel definition of the distance between merge trees, and the development of the algorithm for computing it. In Chapter 6, the proposed distance is utilized as an error measure for the evaluation of commonly

used approximation methods, which leads to specific recommendations on the selection of methods, including for the two proposed techniques.

Chapter 2

Background

This chapter provides necessary background for the main body of research work, including a brief overview of computational topology as well as sufficient theory and definitions to formulate the basis for computing topological information. Next, it reviews the actual features and structures that constitute topological information and discusses various algorithms for their computation. Finally, it offers an overview of techniques for the visualization of topological information, which constitutes an important aspect of creating effective analysis tools, especially because such information is abstract and is, therefore, often difficult to comprehend.

2.1 Computational Topology

Computational topology is a relatively new research area that studies the topology using a computer. It is concerned with characterization of the space, mostly in terms of quantifiable or identifiable topological invariants. For example, in computer graphics, the problem of connectivity for surface reconstruction from point sets requires a correct identification of holes and tunnels. In context of this dissertation work, computational topology enables the extraction of topological information. The basic theory and definitions of computational topology, adopted below from the

overview by Afra Zomorodian [2], allow an introduction of Morse theory. In turn, the Morse theory, and its variations, such as discrete Morse theory and Morse theory adaptation for piecewise linear functions, forms the basis for defining and computing topological features and structures used in this work. Furthermore, an alternative interpretation of the underlying domain space is given using cell complexes, used by the homology theory. A brief introduction into the persistent homology is given to provide an alternative basis for working with sub-level sets.

2.1.1 Basic Definitions

The definition of topological space is adopted as follows. For a set of points X, topology is a subset $T \subseteq 2^X$ such that: 1. If $S_1, S_2 \in T$, then $S_1 \cap S_2 \in T$. 2. If $S_J | j \in J \subseteq T$, then $\cup_{j \in J} S_j \in T$. 3. $\emptyset, X \in T$. Here, $S \in T$ is an *open set* and the pair $\mathbb{X} = (X;T)$ is a topological space. It can be viewed as an abstraction of a metric space, where the notions of neighborhoods and continuity are preserved, however an abstract metric is used to identify the relations. A manifold is a topological space M which locally homeomorphic to the connectivity structures in metric space with given dimensionality. For example, a circle is a 1-manifold S_1, since every point on circle is homeomorphic to an open interval in $\mathbb{R}^1$. Finally, a boundary of the d-manifold is considered to be a $(d-1)$-manifold without boundary.

2.1.2 Morse Theory

The representation of a topological space, given above, allows to introduce the base for computing topological information. In particular, consider a smooth manifold M and a smooth map as $h : \mathbb{M} \leftarrow \mathbb{R}$. Point $p \in M$ is a *critical point* (see Figure 2.1), if the derivative at this point is zero, and if non-zero, the point is a *regular point*. A critical point is nondegenerate, if its Hessian is nonsingular. The map h (sometimes called a *height function*) is called a Morse function, if all its critical points are nondegenerate. According to the Morse lemma, it is always possible to change

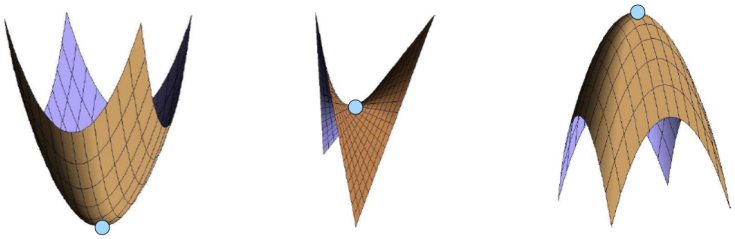

Figure 2.1: 2D critical points: minimum, saddle, maximum. (Image courtesy of Attila Gyulassy)

the map by a small value to create a Morse function. Throughout the dissertation, functions under analysis are considered to be Morse.

2.1.3 Level Set Topology

Given a Morse function f, it is possible to track the evolution of the topology of its isosurfaces. Isosurface is a surface, constructed by connecting all points with the same preset function value called an *isovalue*. Contours are connected components of the isosurface. More formally, isosurface is a level set, defined as $L_c(f) = \{x_i | f(x_i) = c\}$ for an isovalue c, and contours are elements of that set. Level sets and their evolution is the main target of study of level set topology. Various topological structures that encode the information about the level sets are presented more elaborately in Section 2.2.

2.1.4 Cell Complexes

So far, the underlying domain space was interpreted as a smooth manifold, allowing the application of the Morse theory. Alternatively, the space can be interpreted as a finite representation called a *cell complex*. While in the majority of work in level set topology these interpretations are used interchangeably through the work of Robin Forman [37] on the discrete Morse theory and the Morse theory adaptation for piecewise linear functions [71], the exact relationship is case-dependent and at times unclear. Nevertheless, the latter interpretation requires a slightly different

introduction, which is provided further. Essentially, a cell complex is a union of polyhedra glued together along the seams. More formally, a cell complex C should satisfy two conditions:

1. If $c \in C$, then $f \in C$, where f is a boundary face.

2. If $c_1, c_2 \in C$, then $c_1 \cap c_2 = f \in C$.

There are many types of cell complexes defined in computational topology. The simplicial complex is the most popular type of complex due to its structural simplicity. All its cells are d-dimensional simplices, constructed as a convex hull of $d+1$ affinely independent points in $\mathbb{R}^d$. CW-complex carries more relaxed cell conditions, so that simplicial complex satisfies CW-complex requirements, but not vice-versa. It should be noted that a simplicial complex carries a useful property of continuity, which can be generalized for CW-complex as well [37], constituting the link between interpretations given by the discrete Morse theory.

2.1.5 Structures for Point Sets

Cell complexes are naturally derived from the topological space under the assumption that connectivity information is given. But in practice, point sets are obtained by sampling the underlying topological space, and carry no connectivity information. So, in order to approximate the underlying topology, a cell complex have to be somehow constructed. There are several types of abstract and geometric complexes that are commonly used for such approximation, and the selection of a particular type of complex often depends on a given point set. More formally, such complex can defined as a *nerve* in a following manner:

Definition 2.1.1 (Nerve)**.** Let X be a point set embedded in $\mathbb{R}^d$. An *open cover* of X is $U = \{U_i\}_{i \in I}, U_i \in \mathbb{R}^d$, where I is an *index set* and $X \subseteq \cup_i U_i$. The nerve N of U is:

1. $\emptyset \in N$, and

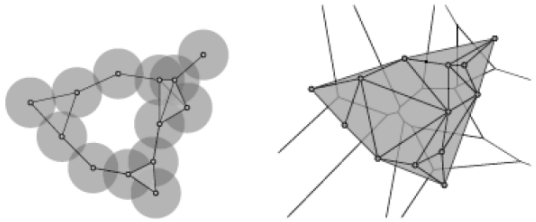

Figure 2.2: Left: The cover is a union of ε-balls, the nerve is the Čech complex. Right: The cover is Voronoi diagram, the nerve is Delaunay complex.

2. If $\cup_{j\in J} U_j \neq \emptyset$ for $J \subseteq I$, then $J \subseteq N$.

Since cover defines the nerve, it is important to consider its properties. U is a good cover, if all U_i are contractible and so are all of their nonempty finite intersections. For a good cover, its nerve is a homotopy equivalent to the cover. Hence, several good covers are considered, whose nerves (i.e., complexes) are often used in practice. The Čech complex is a nerve of the union of ε-balls centered around the input points (see Figure 2.2). It is a good complex for the uniformly distributed point sets. The Vietoris-Rips complex relaxes the Čech complex by permitting a dimension expansion to include simplices with higher dimensions. Another popular complex is a Delaunay complex, which is a nerve of Vonoroi cells $V(p) = \{x \in \mathbb{R}^d | d(x,p) \leq d(x,y), \forall p \in X\}$ (see Figure 2.2). While several other complexes are introduced and used throughout the dissertation (especially in Chapter 6), presented complexes demonstrate the general approach to their interpretation as an approximated underlying domain space.

2.1.6 Persistent Homology

The underlying domain space (i.e., topological space), reconstructed by cell complexes, helps to introduce the concept of homology in algebraic topology. In particular, an approach to studying the topology of the sub-level sets is considered. A sub-level set is defined as $f^{-1}(-\infty, c]$, where the value of c is fixed. See the textbook by Munkres [60] for the detailed introduction to homology. Informally, homology describes the cycles in a topological space: the number of components,

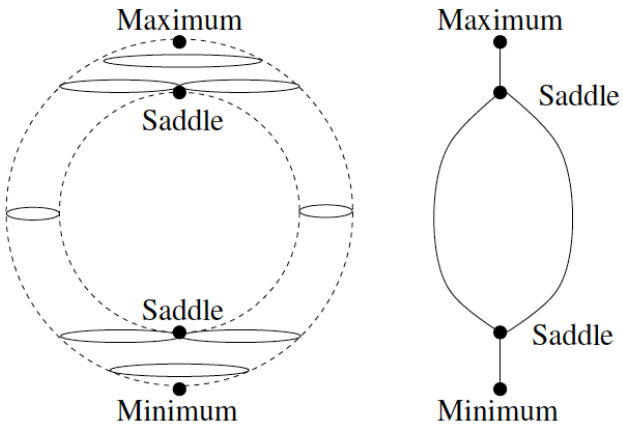

Figure 2.3: The Reeb graph of a torus. Left: Critical points correspond to changes in the topology of contours. For example, the minimum corresponds to the birth of a component of a contour. The lower saddle corresponds to the split of a contour component into two, etc. Right: The Reeb graph, which in addition to critical points, encodes the information about the evolution of components of each contour, thus serving as a topological skeleton of the surface. (Image courtesy of Gunther H. Weber [82])

loops, voids, and so on. This dissertation only considers 0–dimensional cycles, i.e., the connected components. Persistent homology tracks changes to the connected components in sub-level sets $f^{-1}(-\infty, c]$, as the value of c changes.

2.2 Topological Structures

This section reviews topological structures, commonly utilized for capturing the topology of (sub)level sets of functions, including illustrations of structures and an overview of algorithms for computing them.

2.2.1 Reeb Graph

The Reeb graph [70] is a structure that encodes topological changes of contours, as isovalue is varied within the value range of the target function, see Figure 2.3. Accordingly, it preserves

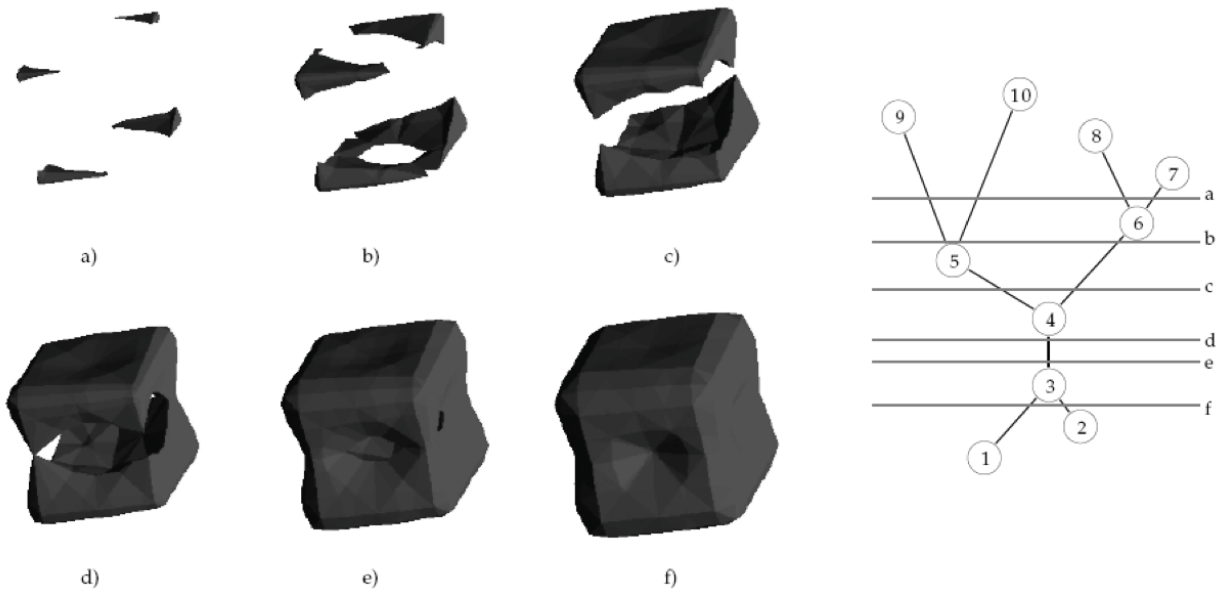

Figure 2.4: Contour tree records (a) birth, (b) merger, (f) split, and death of contour components in a simply connected domain. (Image courtesy of Hamish Carr [21])

complete connectivity information about contour components. Cole-McLaughlin et al. [25] gave a simple algorithm for computing the Reeb graph for triangulated 2-manifolds. Pascucci et al. [69] gave a streaming algorithm that allows an on-line (incremental) computation of the Reeb graph with a complexity of $O(nm)$. Harvey et al. [48] provided a randomized algorithm for computing the Reeb graph with expected running time complexity of $O(m\log m)$, while Salman Parsa [66] gave deterministic algorithm with the worst case complexity of $O(m\log m)$.

Contour tree is a special case of the Reeb graph. It tracks topological changes of contours in a simply connected domain, by recording birth, merge/split, and death of components of contours, see Figure 2.4. First contour tree calculation method that works for dimensions higher that three was introduced by Carr et al. [21]. They constructed contour tree by first building join and split trees, and then merging them. Recently, Chiang et al. introduced an alternative method for contour tree calculation [23], based on precalculating all critical points and building the join and split trees using them, reducing the number of points to be considered.

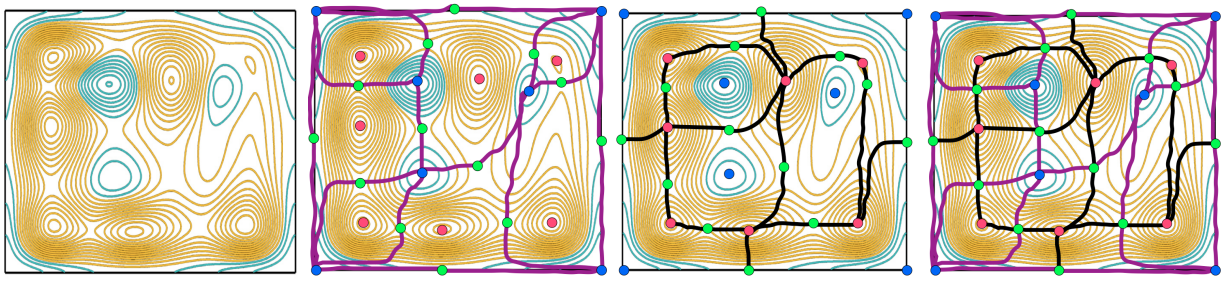

Figure 2.5: Morse-Smale complex. Given a 2D manifold with maxima colored by red, minima colored by blue, and saddles colored by green colors, the middle left image shows a stable segmentation of the manifold with corresponding Morse complex, while the middle right image shows an unstable segmentation of the manifold with corresponding Morse complex. The image on the right is a Morse-Smale complex, obtained by combining stable and unstable Morse complexes. (Image courtesy of Attila Gyulassy)

2.2.2 Morse-Smale Complex

Discrete Morse theory permits the computation of Morse-Smale complexes [37] as partitioning of the space into the steepest flow segments, represented by crystals. These crystals are obtained by first computing complexes of stable and unstable segmentations of a manifold, and then combining them. The stable segmentation of a manifold is obtained by finding the steepest descending gradient flow for each point of the manifold, and associating it with the end point, i.e., a minimum, thus creating a segment for each minimum. Then, for each minimum, steepest gradient paths from saddles to this minimum are computed, as a result constituting a graph called stable Morse complex, see the middle left image in Figure 2.5. The unstable segmentation is computed in a similar fashion by reversing the direction of flow. An example of the Morse-Smale complex computation is given in Figure 2.5.

The method by Edelsbrunner et al. [31] calculates Morse-Smale complexes for a piecewise linear 2-manifold. Their method was extended to handle to 3-manifolds [30]. Gyulassy et al. [44] further improved the algorithm, and discussed scalability issues [45], which remain challenging.

2.2.3 Simplification

The presented structures allow the extraction of useful topological information. However, often they get too large, noisy, and cluttered, impeding the analysis. One solution to this problem is a simplification process. While many importance metrics can be used for simplification, the most widespread metric is a *persistence*. Informally, the persistence of a topological feature signifies how prominent it is in the data. For example, in case of connected components of isosurfaces, the lifetime of a component from its birth to death signifies its persistence. Since the birth and death events correspond to the pair of critical points, either minimum–saddle or saddle–maximum, the persistence itself can be quantified as the function value difference of the pair.

The concept of the persistence was formalized by Edelsbrunner et al. [32] and used for the noise elimination within the context of growing complexes (i.e., filtrations). Carr et al. [20] proposed a simplification of a contour tree by pruning its leafs, i.e., connected contour components, based on their importance. Several importance metrics were considered, including a persistence and local geometric measures (e.g., volume) of connected contour components. Gyulassy et al. [46] described the persistence-based simplification of the Morse-Smale complex, which allowed the authors to handle the noise and extract important topological features. The persistence was also used to produce a hierarchical representation of topological structures, for example Pascucci et al. [67] used a persistence for a multi-resolution hierarchical representation of a contour tree, which helps to reduce the clutter.

2.2.4 Persistence Diagram

The simplest abstraction that records the changes to the connected components in sub-level sets is a persistence diagram. A component is born in the sub-level set $f^{-1}(-\infty, b]$, if its homology class does not exist in any sub-level set $f^{-1}(-\infty, b-\varepsilon]$. This class dies in the sub-level set $f^{-1}(\infty, d]$, if its homology class merges with another class that exists in a sub-level set $f^{-1}(-\infty, b']$ with $b' < b$.

When a component is born at b and dies at d, a pair (b,d) is recorded in the (0–dimensional) persistence diagram of the function f, denoted $D(f)$. Chapter 6 uses persistence diagrams to define alternative error measure for the evaluation.

2.3 Topology-based Visualization

Topological structures are abstract in nature, so their visual representation usually depends on needs of some end-user analysis. Considerations like the level of detail of presented information, as well as visual accessibility and apprehensibility are paramount for enabling and enhancing effective analysis. Some commonly employed methods of visualizing topological structures are covered next.

2.3.1 Graph-based Representations

Since presented topological structures are essentially graphs, the straightforward approach to their visualization is based on presenting them as graphs on the plane or in volume. A number of existing techniques use this approach, employing various graph layout schemes. For example, Pascucci et al. [67] presented an orrery-like hierarchical visualization of contour trees in the volume, created using a radial tree layout [6], the layout that allows avoiding an issue of self-intersections. Gyulassy et al. [44] visualized 2D/3D Morse-Smale complex directly in the original domain space, by embedding vertices into corresponding geometric locations, and connecting them by straight edges. As Morse-Smale complex with dimensions higher than three cannot be visualized in the original domain space, Gerber et al. [43] presented the graph layout that utilizes dimension reduction method to embed vertices and edges (represented as curves) from high-dimensional space onto the plane. In similar fashion, the technique presented in Chapter 3 uses graph-based visualization of Morse complex [9], with vertices projected from original high-dimensional space onto

the plane.

2.3.2 Visualization Metaphors

Graph-based representations are often insufficiently detailed or hard to read. One way of resolving these issues is to link additional visuals, e.g., linking a contour tree with a volumetric view, thus showing isosurface(s) corresponding to selected level(s) in the contour tree [22]. Alternatively, a number of visualization metaphors address these issues in various settings. For example, a visualization metaphor called toporrery [68] is an augmented version of previously mentioned orrery-like hierarchical representation of contour trees, designed to incorporate provided importance metric. Another metaphor that also uses the radial layout to visualize contour trees, is called topological cactus [84], which is able to incorporate additional quantities using cylindrical extrusion of tree edges and spikes. The metaphor called topological landscape [83] went away from tree-like representations, and tackled the reverse problem of constructing a 2D function (i.e., a landscape) which has a given contour tree. Several versions of topological landscapes exists [47, 61, 62, 28], including a geometry-preserving topological landscape [8] presented in Chapter 4 of this dissertation. Finally, Correa et al. [27] proposed topological spines as a visualization of an extremum graph, defined in their work as a simplified substructure of Morse-Smale complex. Their work follows the Morse complex graph-like visualization discussed in Chapter 3.

Chapter 3

Topology–based Analysis of Transformation Pathways in Chemical Systems

This chapter presents a novel topology-based analysis technique that focuses on the analysis of transformation pathways in complex chemical systems [9]. The technique presented here demonstrates the power and utility of the topology-based approach, especially in aiding the scientific knowledge discovery from large high-dimensional data sets. The selection of approximation methods for this technique is supported by the study conducted in Chapter 6.

3.1 Introduction

In chemistry, for example, transformation processes that involve changes of relative positions of atoms in chemical systems are of fundamental interest. Examples of such transformations include internal rotation of fragments of a molecule (e.g., conformation change), translation of atoms or molecules within a chemical system (e.g., diffusion), and shifting of atoms leading to breaking and/or rearrangement of chemical bonds (e.g., chemical reactions). An important factor of any transformation is its cost. Chemists usually focus on transformation between configurations of

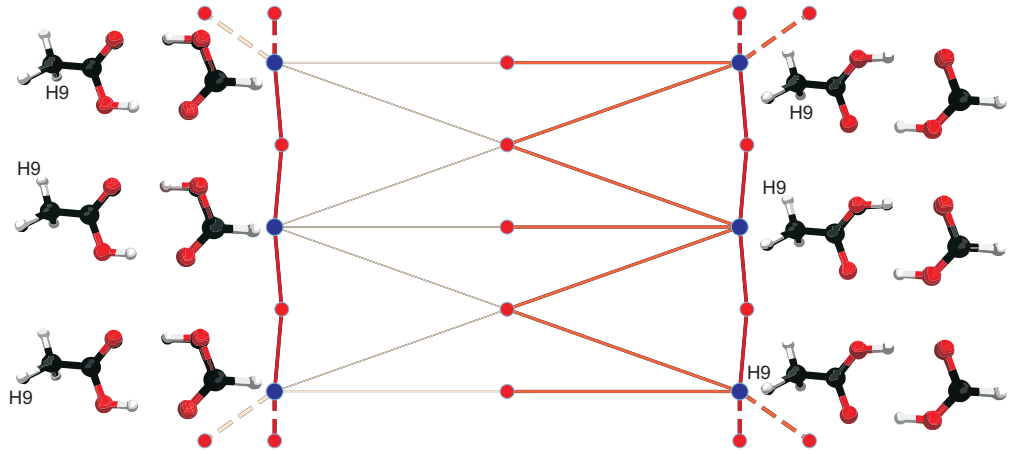

Figure 3.1: A 2D graph showing the topology of the 3D potential energy function of a complex of formic and acetic acids, which depends on the positions of constituting atoms. Blue and red dots represent minimum energy configurations and lowest barriers connecting neighboring minima, respectively. Edges represent the energy cost of a particular transformation, with darker and wider edges corresponding to transformations through lower barriers (more likely transformations). Two vertical branches corresponding to different positions of protons are visible on the left and right side of the graph. Energy barriers for transforming the right branch into left one are lower than for the reverse transformation.

chemical systems corresponding to *energy minima*, representing stable states for the system. The term "configuration" is generalized here and can take on two meanings: It can refer to a set of strictly defined positions of all atoms of a studied system as it is done in the vast majority of quantum chemical calculations relying on the Born-Oppenheimer approximation. On the other hand, it can also refer to a representative or dominantly populated state within a class.

The cost of a transformation between stable states can be defined as the energy difference between the two minima involved in the transformation. In a typical case, a transformation involves a transition through a higher energy configuration, a *barrier* that determines the probability of the transformation (or time required for it to happen). The analysis of *transformation pathways* in a chemical system usually follows the same general scheme. After identifying important energy minima, connecting pathways and corresponding barriers are found. A chemical system has $3n$ degrees of freedom, where n is the number of atoms. Analyzing transformations in such a system

requires identifying minima and transition states in a $3n$-dimensional space called *energy landscape*. Chemists can often reduce the dimensionality of this space by exploiting prior knowledge about the system. For example, when considering a diffusion process, it is often sufficient to consider only translation and rotation of a rigid molecule, and the analysis can be performed for a six-dimensional energy function. When considering only conformational changes of a molecule, it is sufficient to investigate $3n-6$ internal degrees of freedom. The latter can be simplified further by assuming that some coordinates are constant due to very high energy cost associated with their change.

3.2 Contributions

Contributions of this chapter are: a technique that provides sufficient level of detail of transformation pathways in higher dimensions; embedded handling of noise and periodicity; use of dimension reduction and incorporation of additional information to improve visual analysis by end-user.

3.3 Related Work in Chemistry

Flamm et al. [35] proposed the barrier tree concept for the analysis of degenerate energy landscapes, and Heine et al. [49] described a visualization technique for multiple barrier trees. However, a barrier tree is an alternative name of a merge tree, which omits some barriers that are of interest for required analysis. Similarly, James et al. [52] introduced a *disconnectivity graph* to visualize energy landscapes of water clusters in an uniform electric field. This graph is similar to a barrier tree but uses different visual representation.

Recently, Okushima et al. [64] proposed a way to generalize the disconnectivity graph concept as a *saddle connectivity graph*. To construct this graph, their technique locates minima and sad-

dle points and combine them into a schematic line representation. While the saddle connectivity graph shows all relevant barriers, the proposed technique is computationally more efficient and the resulting layout scheme facilitates easier tracking of separate transition pathways.

Another method for the analysis of energy landscapes, proposed by Apaydin et al. [1], is a *stochastic roadmap simulation*. However, this method focuses on obtaining a high-level overview, e.g., finding the global minimum in a protein folding problem. Consequently, it can miss minima and saddles that may be important. This method is able to analyze system with larger number of degrees of freedom at the price of being stochastic, thus not exploring the entire domain of the configuration space. One the contrary, proposed technique is oriented to provide information about the entire energy landscape within the domain and let the user choose the desired level of detail.

3.4 Algorithm

The technique uses a combination of the Morse complex that captures the relationship between energy minima, a dimension reduction technique called multi-dimensional scaling [79] for projecting minima positions to the two-dimensional plane, and, finally, a graph layout incorporating all necessary chemical information. The technique employs simplification schemes both for noise reduction and for focusing attention on the most relevant features of the chemical systems. The technique also introduces a strategy for handling a complexity, added by a possible periodicity in the input data. The graph, produced by this technique, shows relevant transitions between energy minima and provides a succinct summary view of relevant transformations of the considered chemical systems.

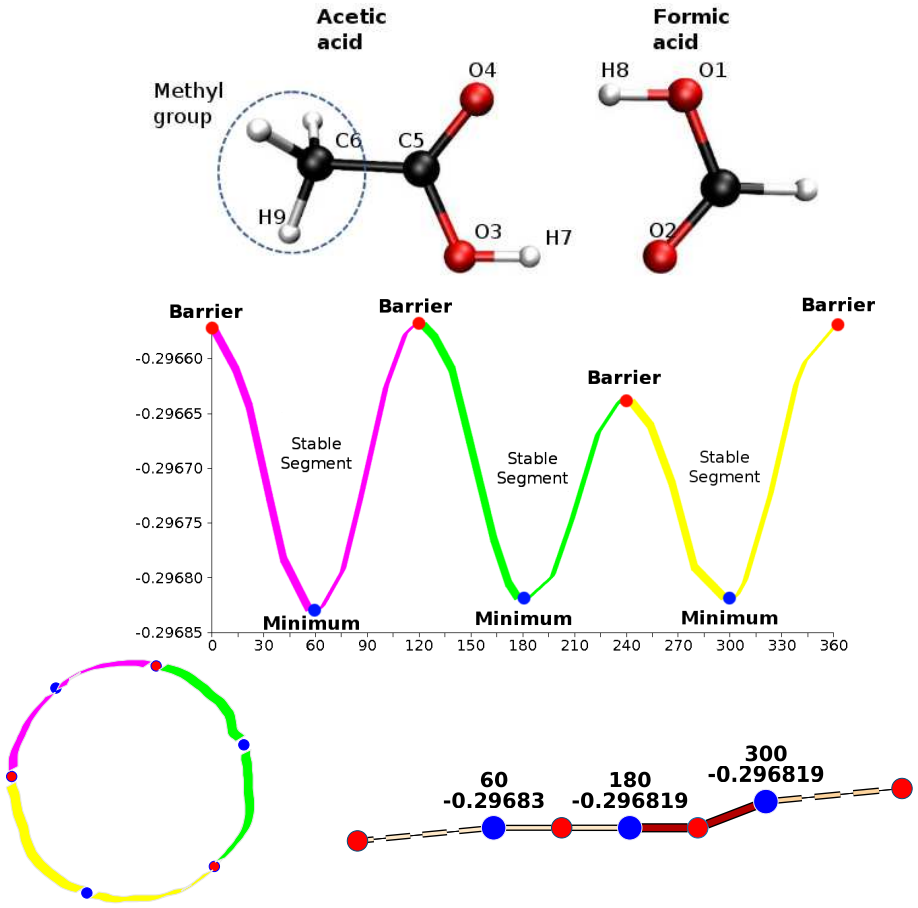

Figure 3.2: Left: Chemical system configuration of a dimer of formic and acetic acid. Middle left: Energy of the system in Hartrees (y-axis) as a function of the methyl group rotation angle between 0 to 360 degrees (x-axis). Blue and red dots mark energy minima and barriers, respectively. Middle right: Naive graph representation of transformation pathways. Right: Proposed graph representation, where the top row of a energy minima label displays its coordinate and the bottom row its energy value.

3.4.1 Morse Complex

To obtain structural transition information, the technique approximates the Morse complex for the energy function f_e by producing a segmentation of the input data into stable descending manifolds. Here, the segmentation scheme is similar to Gerber et al. [43], and is done by finding a neighbor with the steepest descending gradient for each point, and subsequently applying a union-find algorithm to determine its corresponding minimum.

Segmented descending regions in the resulting representation correspond to classes of configurations, each associated with a minimum, i.e., stable state. Each descending region can adjoin several other regions. The saddle between two regions, i.e., the lowest function value along their boundaries, corresponds to the barrier between the states. Since minima and saddles in the Morse complex correspond to energy minima and barriers, chemical transformation pathways are given by the edges connecting two minima through a saddle. Assigning minima and saddles to nodes and connecting pathways as edges create the Morse complex graph (MCG).

The major steps of this algorithm can be summarized as follows: First, perform a sweep over all input points to identify and store the steepest descending neighbor for each point. Then, use a union-find data structure to segment all input points into regions whose gradient flow ends at the same minimum, thus approximating stable manifolds. For each segment, consider its neighboring segments and identify the lowest value along the boundary. Then, construct the MCG by adding or updating three nodes — two minima and saddle, and two connecting edges. Finally, check whether any of the minimum–saddle edges crosses a periodic domain boundary, and if so, determine the direction. The resulting MCG is passed on to the graph layout stage.

3.4.2 Simplification

Noise in scientific data often complicates the analysis, as it might lead to convoluted resulting graphs. To ensure that a generated graph does not clutter the end users, effective means of sim-

plifying the MCG should be applied before laying it out. Simplification of the MCG is done by removing low-persistence minima from the graph. The persistence for each minimum–saddle pair is defined as the absolute difference of energy function values. Since one minimum might belong to several minimum-saddle pairs, and thus have multiple associated values, the lowest among them is selected to decide, whether a minimum is to be removed. This approach gives a good measure to identify the noise, which mostly corresponds to minima that appear due to numerical error. Further results also show that minima with a low persistence correspond to a latent chemical configurations, hence their elimination does not affect the analysis. Finally, to have additional flexibility in this approach, a persistence threshold, under which minima are simplified, can be changed by a user.

A sequential elimination of minima is performed in order of their persistence (see Figure 3.8). First, the minimum with the lowest persistence is identified. Then, a minimum that is connected through the lowest saddle to the minimum that needs to be reduced is found. These two minima are then merged into a new minimum, which inherits all the neighbors of merged minima. Finally, all the saddles (barriers) are recalculated along the border of newly create minimum, which also inherits necessary information, associated with merge minima (e.g., periodicity information).

Two approaches for finding a suitable persistence threshold can be used. The conventional approach for finding such threshold is based on a persistence diagram, which is obtained by gradually increasing the simplification threshold value within the possible value range and determining the current number of minima for each value. Large drops in the number of minima indicate a large numb of related minima and thus are candidates to be simplification threshold. For example, the persistence of the minima, introduced by numerical errors, normally can be easily identified by such big drop. While this approach can be useful due to its generality, it is combined with specialized approach, based on use of chemical domain knowledge pertaining to the considered system. These two approaches complement each other and provide a user with sufficient flexibility to select an appropriate persistence threshold for simplification.

3.4.3 Periodicity

Proper handling of periodicity for different chemical structures and presenting them in a simple way for analysis is key to providing readable graphs. Chemistry datasets can also have some of their dimensions being periodic, and handling and presenting such data sets leads to additional visualization challenges. Periodicity handling in such data sets is elegantly done by marking edges of the MCG that cross the domain boundary of a periodic dimension and presenting them in a graph using dashed edges. Furthermore, directions are explicitly indicated, which could be crucial in the analysis of porous materials.

The interpretation of periodicity in the data is straightforward: The maximumboundary and the minimum boundary along an axis corresponding to a periodic dimension will be virtual neighbors when wrapping around the data set. Thus, a maximum and minimum cell along that axis share a face, and the triangulation must subdivide that face consistently. The use of Freudenthal's subdivision scheme [38] ensures such consistency and avoids possible cracks and hanging nodes in a triangulation that might occur on the domain's boundary faces. While Freudenthal subdivision can be generalized to arbitrary dimensions, the number of edges in a mesh grows exponentially, making this approach infeasible for dimensions larger than three. Thus, to create neighborhood graphs for high-dimensional data sets, the k-nearest-neighbor algorithm [76] is used, and to guarantee the correct triangulation, edges that are not bi-directional are removed from the neighborhood graph. Both methods produce relatively low approximation errors, see Section 6.5 for details.

3.4.4 Graph Layout

The first step of graph layout algorithm is projection of the location of minima to two dimensions using classical metric multidimensional scaling (MDS). While it seems intuitive to apply MDS both to minimum and saddle positions, this approach often leads to a cluttered graph layout. Projecting only minima and placing saddles along lines between minima results in a much cleaner

graph layout and supports encoding of additional information based on the location of barriers.

While the projection at this point is simple to navigate, overlaps of edges and nodes might conceal important information. This issue is addressed by an iterative optimization algorithm that minimizes node/edge overlap in the final layout. If to denote the node's center N_{center} and the node's projection to the edge's mid-axis as $P_{N_{\text{center}}}$, the distance between them tells if the node and the edge overlap or not. Let's define the overlap function as $D_{\text{overlap}} = (N_{\text{radius}} + E_{\text{width}}/2) - \text{distance}(N_{\text{center}}, P_{N_{\text{center}}})$, where N_{radius} is the radius of the node, and E_{width} is the width of the edge. If the D_{overlap} is positive, there is an overlap, otherwise not. Thus, the goal is to iteratively move the node away from the edge along the projection line, until D_{overlap} becomes zero or negative. To ensure the preservation of the overall structure, the center of the node does not move further than the predefined ε radius. Although the convergence rate of the algorithm is found satisfactory, a user can manually set parameters for either faster or clearer view of the final graph layout.

After the final positions for all nodes of the graph are calculated, the chemical information from the model system is added to the graph. Since the most important information is the *energy difference* between a minimum and a barrier, the graph edges are colored according to the energy difference value. The Boltzmann distribution states that the probability of accessing a state decreases exponentially with its energy. The lower the energy difference is, the higher is the probability of a transformation. Accordingly, the paths with smaller energy differences should be displayed more prominently (wider edges with darker colors), see Figure 3.1, as they are of higher importance. On the other hand, a higher energy difference makes the transformation unlikely, hence corresponding edges are very narrow and lightly colored edges, making them almost invisible, signifying that these edges carry little to none importance in analysis. In addition, a user is provided an option to discard such edges, if they are beyond the highest possible and/or interesting energy difference level, which also helps to further clarify the potential visual clutter. *Periodic edges*, identified as edges that cross the periodic boundaries, are dashed and the common saddle of the two connected minima is broken into two nodes (see Figure 3.2). This helps chemist to identify periodic edges

right-away. Finally, all minima are labeled with coordinates (first line) and energy value (second line). This information helps chemists to relate the graph to the state of the model system and it guides analysis.

3.5 Results

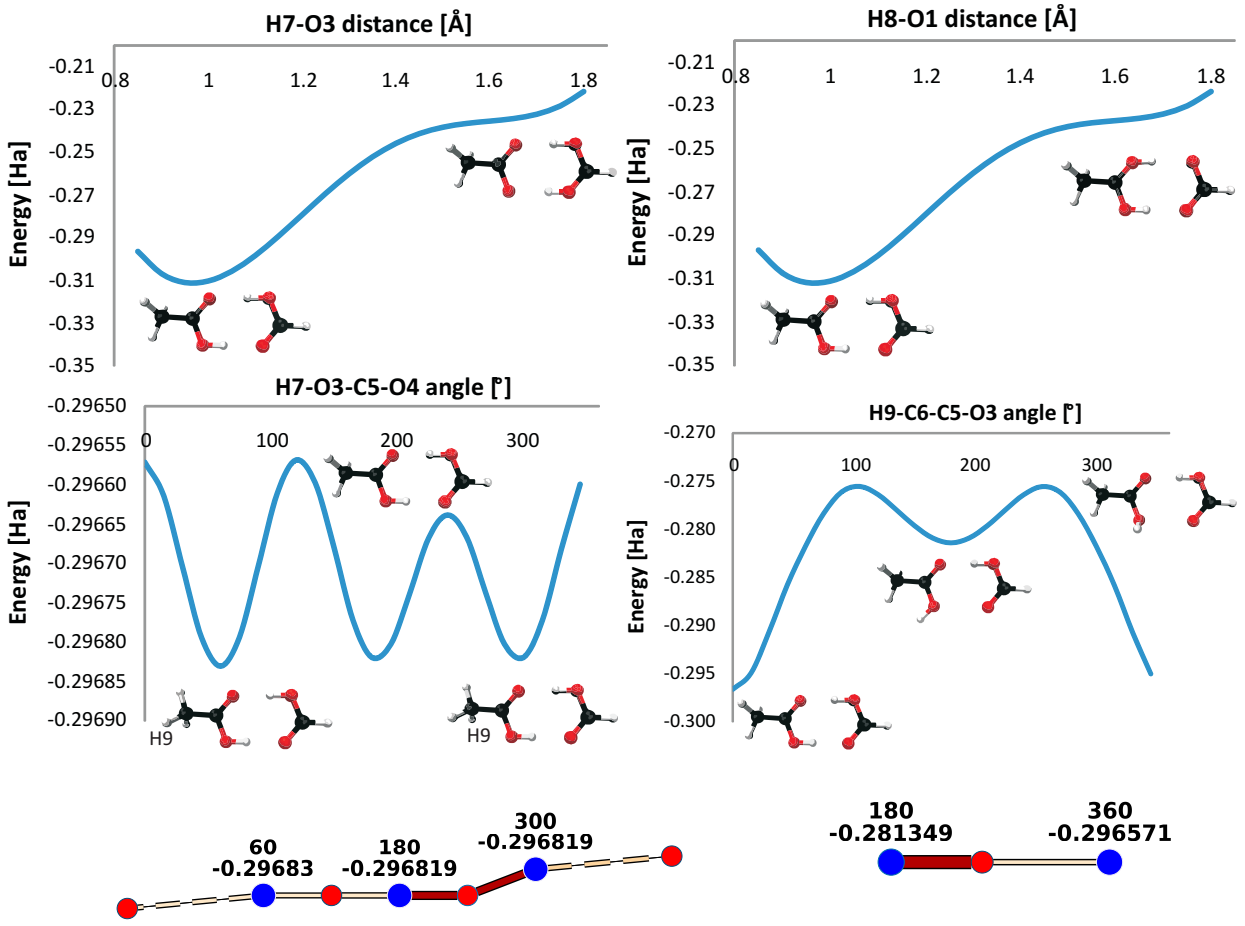

Figure 3.3: Energy functions of DFA along its four dimensions (top), and respective graphs for the third dimension (bottom left), showing the periodic nature of the dimension, and (bottom right) the fourth dimension, where it correctly handles the special case of periodicity. Labels for latter two use the convention from Figure 3.2.

In order to demonstrate the visualization of transformation pathways, two distinct chemical

data sets were studied. The first study concentrates on a dimer of formic acid and acetic acid (DFA), which serves as an example of a small molecular cluster, for which energetics of conformational changes and proton transfer reactions are studied. The second study investigates energies related with diffusion of a small molecule in a crystalline porous material, namely CH_4 molecule in a LTA zeolite.

3.5.1 Dimer of Formic Acid and Acetic Acid

Small model systems are often selected by chemists to study intermolecular interactions such as hydrogen bonds. The small size of such complexes does not only facilitate gas-phase experiments (small systems usually have high vapor pressure) but also enables performing accurate *ab initio* calculations to investigate properties and simulate transformations in these complexes. In this context, a dimer of carboxylic acids can serve not only as model of hydrogen-bonded system but also can be used to study proton transfer reactions [4]. A dimer of formic acid and acetic acid is selected as a model system to demonstrate generation of graph representations of transformation pathways in this chemical system.

The initial geometry of DFA presented in Figure 3.2 was optimized at the semi-empirical PM3 level of theory [74, 75] to find a minimum energy configuration. In particular, up to four degrees of freedom of the system are considered. Therefore, the geometry is optimized to generate a four-dimensional grid with energies of DFA in the considered configurations. The geometries of each configuration were generated in the following manner. All atomic positions (in internal coordinates), except those involved in the considered degree of freedoms, were fixed at their optimal positions. The remaining coordinates of DFA were systematically modified along the four considered directions. Specifically, dimensions of the energy grid are defined as follows:

- First dimension: length of H7–O3 bond. This degree of freedom describes stretching of H7–O3 bond. Its range is scanned between 0.85Å and 1.85Å with a step size of 0.05Å. The

distance unit throughout this example is Ångström (1Å=10^{-10}m).

- Second dimension: length of H8–O1 bond. Its range is scanned from 0.85Å to 1.85Å with a step size of 0.05Å.

- Third dimension: H7–O3–C5–O4 dihedral angle, which is scanned with a step size of 15 degrees (24 samples, periodic). This transformation rotates the H7–O3 group around the C5–O3 axis and has been selected to investigate the energetic effect, associated with disrupting the O3–H7...O2 hydrogen bond.

- Fourth dimension: rotation of C6 methyl group along the C6–C5 axis. It is defined by the H9–C6–C5–O3 dihedral angle, sampled at a step size of 15 degrees. During the rotation of the methyl group, all internal coordinates of atoms of the methyl group are fixed.

For each of the generated geometries, the corresponding energy was calculated at the semi-empirical PM3 level. All calculations were performed with the Gaussian03 package [39]. The resulting 4D energy grid was analyzed and used to generate corresponding graph representations. The energy unit used for this example is Hartree (1Ha≈27.211eV).

To simplify the referencing, an energy function of the system is denoted as F_X, where $X = \{1,2,3,4,12,13,14,23,24,34,123,124,134,234,1234\}$ is a set of dimension combinations. For example F_{124} corresponds to the energy function, defined in space of the first, second and fourth dimensions. Absent dimensions in this case are assumed to have a default (optimal) fixed coordinate. Each graph representation is referred correspondingly M_X. Note that coordinates are scaled between zero and one in all calculations and figures presenting F_X.

Figure 3.3 shows the energy function for each dimension of the system. F_1 and F_2 have only one minimum each, thus no transformations between minima exist in the system. However, F_3 has three minima, thus three transformations are possible. M_3 describes all possible transformations and barriers, and handles the periodic nature of the dimension by showing the barrier at 0 (and

359) as the same node on the left and right sides. Therefore, there are two ways of getting from one minimum to any other, which naturally corresponds to left and right rotations of the methyl group of acetic acid. Now, one would expect that since F_4 is also periodic, M_4 would have two different paths to get from one minimum to the other. However, as only the lowest barrier between two minima is of significance, M_4 shows only the lower of two barriers. Indeed, if the user has to choose the rotation direction, (s)he would choose the one that goes through the lowest barrier.

Figure 3.4 shows the energy function landscapes and corresponding graph representations for each of two selected dimension pairs. One-dimensional analysis is simple, but the analysis of two-dimensional functions introduces more complexity. F_{12} in Figure 3.4 is still relatively simple, with two minima and the lowest point connecting them. However, $F_{13}, F_{14}, F_{23}, F_{24}, F_{34}$ are not so intuitive due to one or two periodic dimensions. It is obvious from the energy function landscape that there might be several transformation paths between each two minima. For example in Figure 3.4 multiple paths are possible between the minimum at (60,180) and the minimum at (300,180).

Figure 3.5 shows two examples of the energy functions for three dimensions. Since the energy functions becomes complicated in three dimensions, the analysis have to rely on graph representations to observe next important characteristics. Thorough analysis of M_{124} in Figure 3.4 shows that in a given system, a much better transformation between the minimum at (0.95,0.95,180) and the minimum at (1.75,1.75,0) can be found by visiting the third minimum at (0.95,0.95,0), although direct transformation is also available.

The final example is the graph representation of the 4D energy function of DFA depicted in Figure 3.6, for which plotting the original energy functions in all four dimensions was not feasible. The resulting graph is fairly easy to analyze. It clearly shows energy separation of the graph into two subgraphs: one corresponding to rotamers of DFA (with minima denoted A1-A9) and another with double proton transfer (minima denoted B1-B6). Transitions between minima within each subgraph are at low cost as shown by heavy edges.

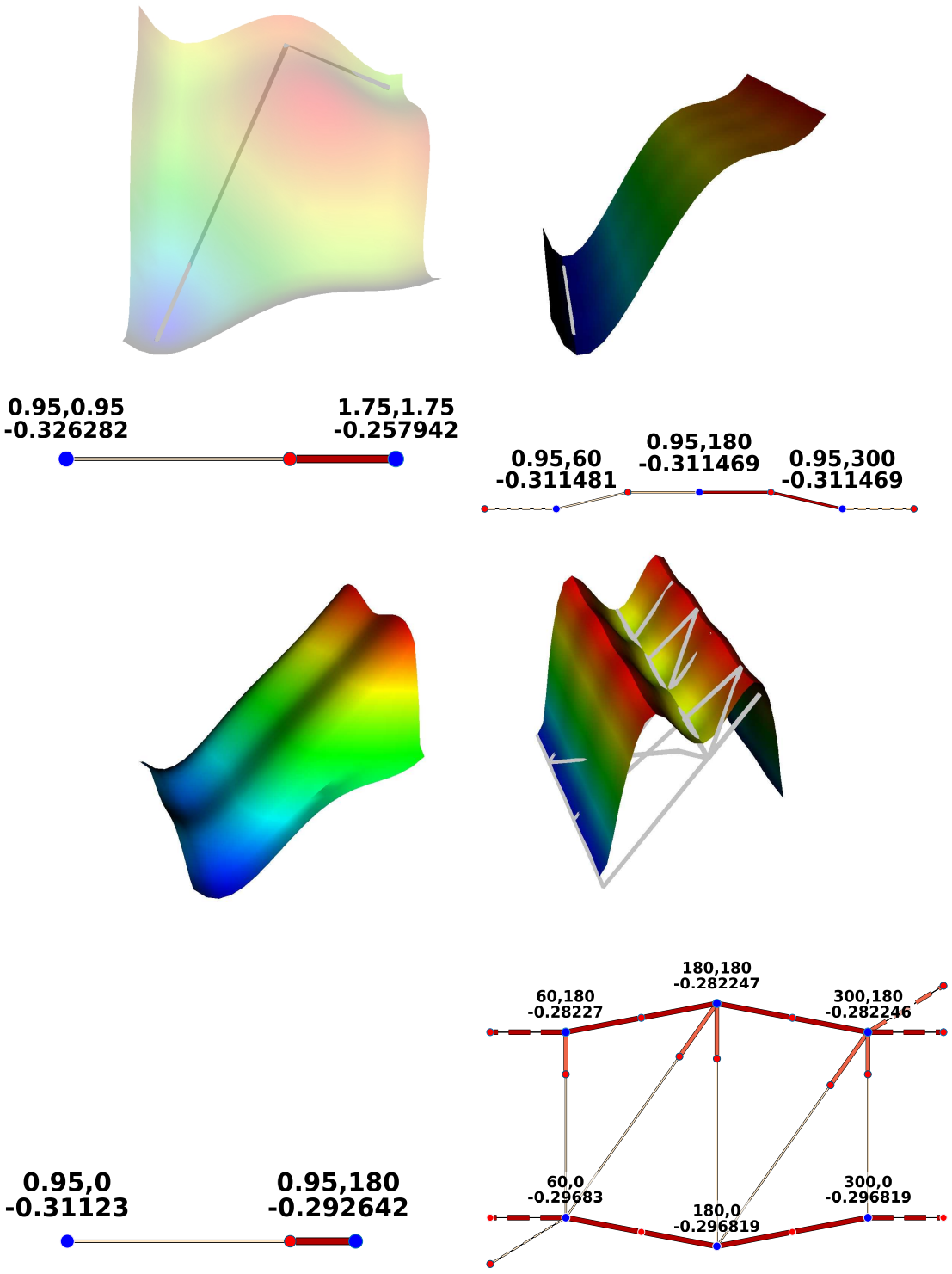

Figure 3.4: Two-dimensional energy function landscapes of DFA and corresponding graph: (a) $F_{12} - M_{12}$ (b) $F_{13} - M_{13}$ (c) $F_{14} - M_{14}$ (d) $-M_{34}$. The graph representation uses the same labeling as the one used in the previous figures.

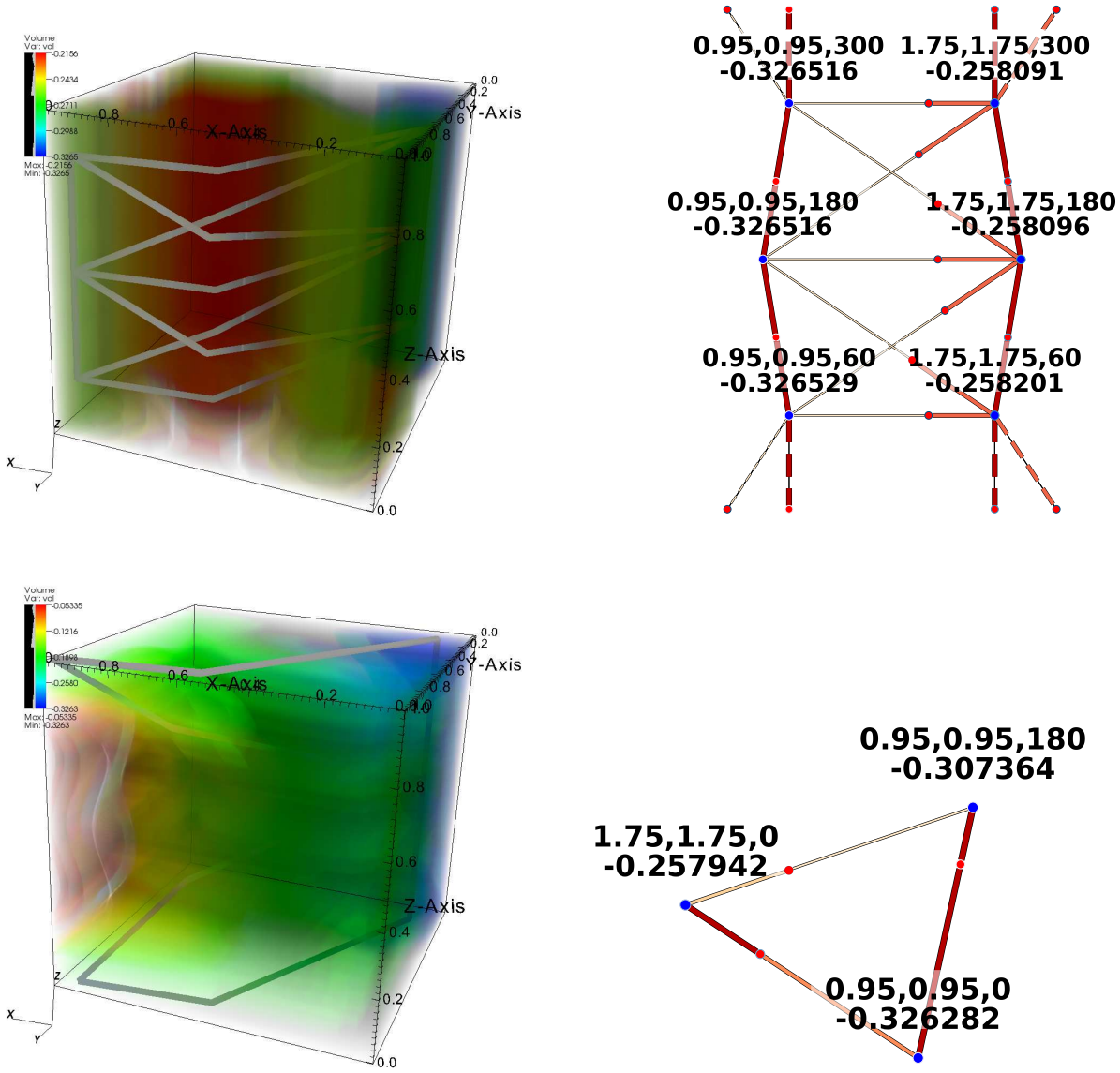

Figure 3.5: Three-dimensional energy function volume renderings of DFA and corresponding graphs: (a) $F_{123} - M_{123}$ (b) $F_{124} - M_{124}$.

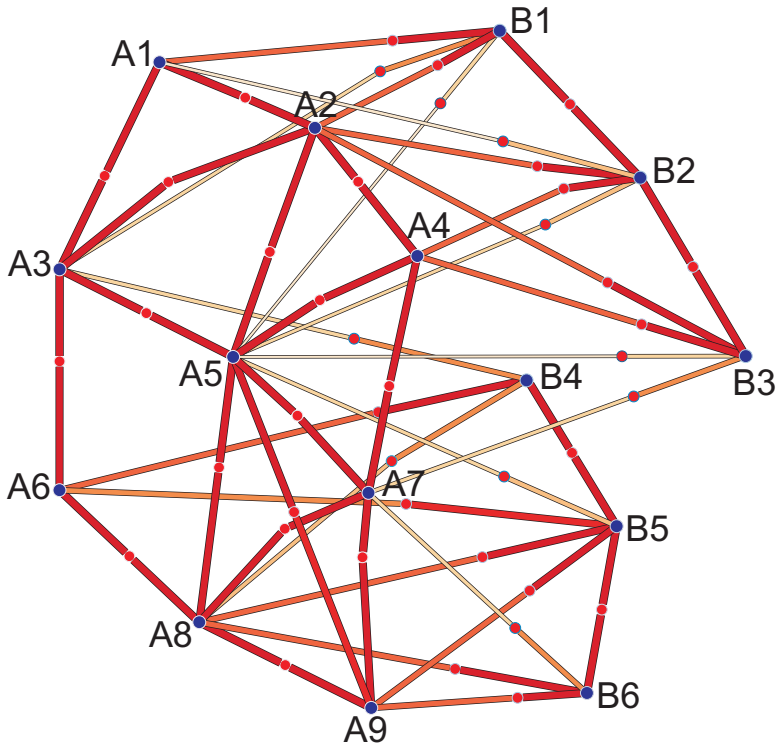

Figure 3.6: Graph of the 4D energy function of DFA. Nodes are labeled A1 to A9 and B1 to B6 to highlight two subgraphs. A1-A9 correspond to minima with coordinates of 0.95Å along the first and second dimension. The remaining third and fourth coordinates are (45,345), (180,345), (60,180), (285,345), (180,180), (60,0), (300,180), (180,0), and (300,0) for A1 to A9, respectively. Similarly, the first and second coordinate for B1-B6 are 1.75Å. The remaining coordinates are (60,345), (180,345), (300,345), (60,0), (180,0), (300,0) for B1 to B6, respectively.

3.5.2 Free Energy of a Guest Molecule in a Porous Material

The second example of application of the technique involves analysis of free energy of a guest molecule inside a porous material. Porous materials contain complex networks of void channels and cages that are exploited in many different industrial applications. Zeolites, probably the most recognized class of crystalline porous materials, have found wide use in industry since the late 1950s. They are commonly used as chemical catalysts, in particular as cracking catalysts in oil refinement, membranes and adsorbents for separations and water softeners [3, 73, 72, 55]. For example, porous materials can be used as membranes to separate carbon dioxide (CO_2) from CH_4 in cleaning up of natural gas.

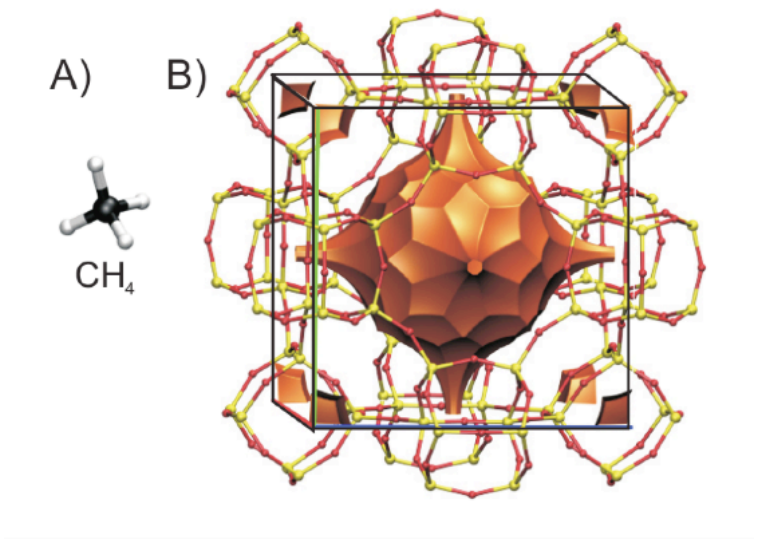

Figure 3.7: Picture of (A) methane, (B) LTA zeolite structure. The orange isosurface highlights the closest distance to which the center of the guest molecule center can approach. The large cage is located in the center of the unit cell. The small cage is shared among eight cells and is visible in the corners of the unit cell.

One of the key processes that determines performance of membranes is diffusion of guest species. Diffusion of gases inside a porous material is controlled by free energy barriers. Analysis of possible diffusion pathways and the associated barriers is therefore critical to understand and design optimal separation devices. Proposed representation is ideally suited to facilitate such understanding.

In the following, the graph representation of free energy of the CH_4 molecule inside LTA zeolite is demonstrated. The corresponding 3D free energy grids were prepared by the following procedure. The 3D space describing a periodic box of LTA zeolite was divided into 239x239x239 volumetric bins. A Monte Carlo simulation was performed to predict an average free energy of a guest molecule inside each bin. The details of this procedure were given by Kefer et al. [54], with the particular force-field approach, developed by García-Pérez et al. [41]. The energies used in this example are expressed in k_BT.

The free energy function of CH_4 in LTA zeolite is substantially different from the energy func-

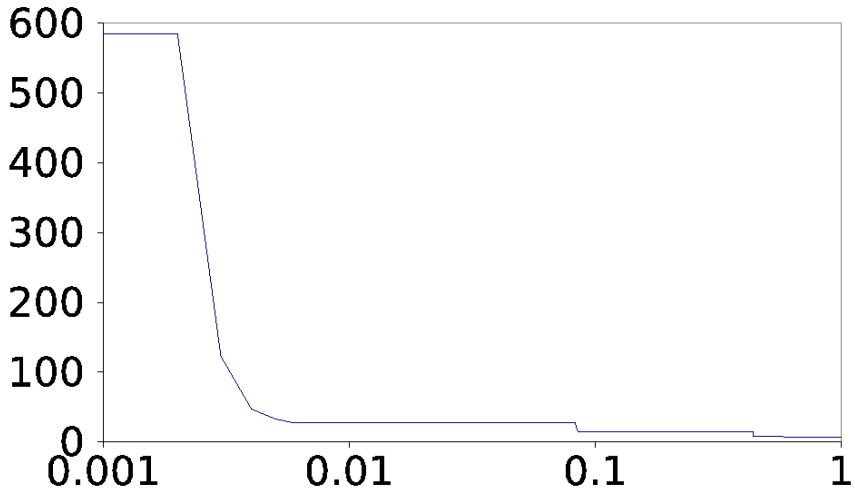

Figure 3.8: Persistence diagram of the energy function of a CH_4 guest molecule in porous material. The large drop in number of minima (y-axis) hints the optimal threshold value for simplification, in this particular case around 0.005.

tions of DFA. The noise of the dataset is high, thus it required application of the described simplification scheme to reduce the final graph, based on the persistence diagram in Figure 3.8. Figure 3.9 shows free energy and the corresponding graph representation for the CH_4 molecule inside LTA zeolite. This graph represents only the large cage of LTA zeolite, which is the only fragment of void space in LTA accessible to CH_4. The graph representation of the free energy reveals important information about the material. There are 14 important energy minima per periodic unit cell of LTA corresponding to favorable locations of the adsorbing CH_4 molecule (adsorption sites). Six of them correspond to lower energy (ca. 3.7 k_BT), and are localized near windows connecting two periodic cells (near faces of the unit cell). The remaining eight minima are localized further away from the windows, on the surface of the large cage of LTA. The further analysis of connections between nodes/minima in these representations reveals that all 14 minima localized within the big care are separated by low barriers, and therefore hops of CH_4 between the adsorption sites are feasible. However, connections between large cages in the extended material lead through high barriers. These high barriers along diffusion paths in every direction are reflected in slower diffusion rates.

Discussion of diffusion of guest molecules inside zeolites extends beyond the scope of this

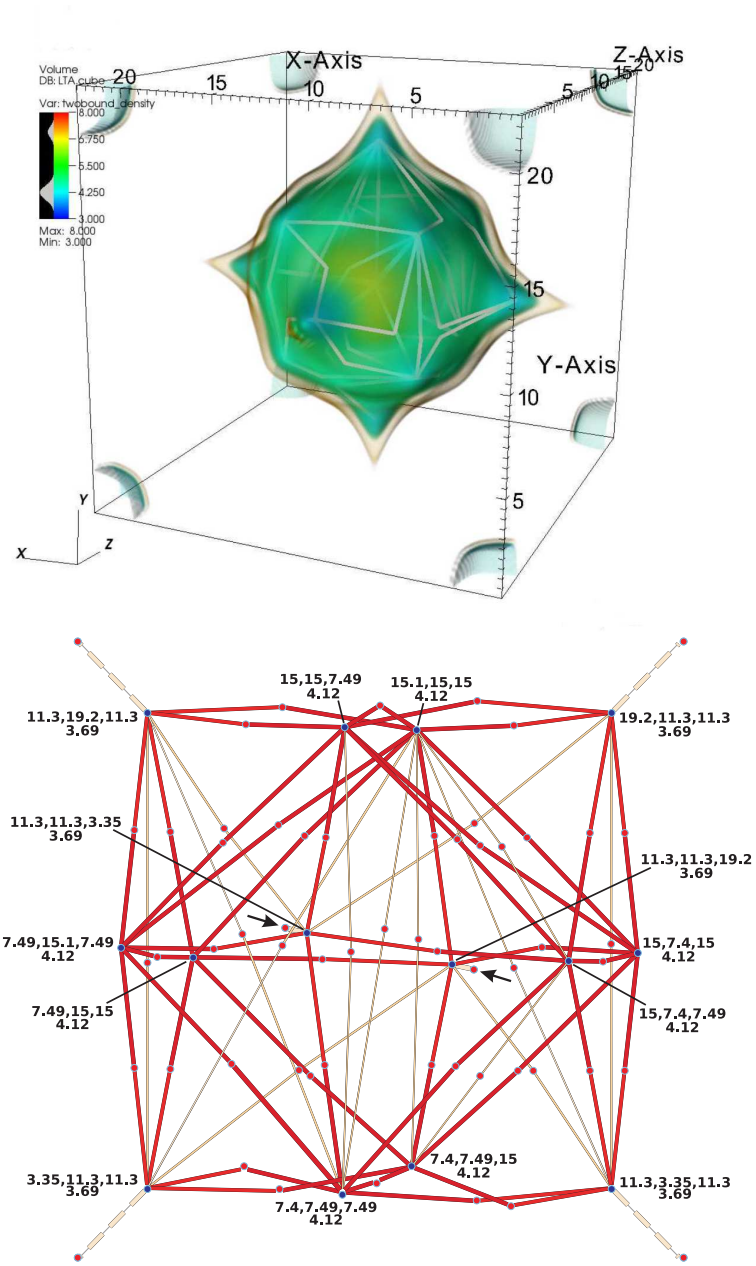

Figure 3.9: Top: Volume rendering of the energy function of a CH_4 molecule in an LTA zeolite and lowest energy paths connecting neighboring minima. Bottom: Corresponding graph showing lowest energy paths (an edge going through a face of the periodic box marked with arrows).

chapter. However, it should be pointed out that proposed graph representations of free energy of different guest molecules inside a material allow for fast analysis and comparison of a material's properties with respect to these guest molecules: these representations reveal differences in numbers and significance of adsorption sites as well as highlight the differences in barrier heights along diffusion paths.

Chapter 4

Comparative Geometry–preserving Topological Landscapes

This chapter presents a novel topology-based technique for the comparative analysis of scientific data. In particular, this technique embeds the geometric proximity information of critical points into a visualization metaphor called *topological landscape*, which, in turn is based on a topological structure called the contour tree. Again, the selection of approximation methods is supported by the study conducted in Chapter 6.

4.1 Introduction

In light of major advances in simulation capability and growth of computational power, visual exploration of scientific data is becoming an increasingly important task. An integral part of visual exploration is the presentation of helpful two- and three-dimensional abstractions that provide an insight into the structure of the scientific data.

One approach to creating such abstractions is based on the use of topological information. Important insights are known to be obtained using topological analysis in a number of fields [85,

15]. In particular, the contour tree was extensively used in scalar data analysis and visualization [5, 80, 21, 59]. However, interpreting the contour tree usually requires at least an intermediate level understanding of Morse theory and related concepts. As a result, topological landscapes [83] have been developed to convey the same information in a more intuitive manner, namely as a two-dimensional terrain with a same level-set topology as the original data set.

A major feature that is missing in the original technique by Weber et al. [83] and its modifications [47, 61, 62] is a correlation between the geometrical placement of the topological features in the landscape and the original domain space. This property simplifies the process of understanding the landscapes by domain scientists by correlating the proximity of the topological features. It also adds a new capability of comparing functions. However, given the potential complexity and higher dimensionality of the data, it is not simple to develop it.

A new technique for building geometry-preserving topological landscape from a contour tree addresses the problem of correlating the function and its topological landscape. The proposed technique also makes it possible to compare several complex scalar functions via their topological landscapes, something previous versions of the topological landscapes lack.

The technique consists of three major stages: first, it computes a contour tree of the input function; second, it uses the dimension reduction to project the vertices of the contour tree onto the plane and graph drawing algorithm to connect them by non-intersecting edges; finally, it uses a contour construction algorithm, which creates contours of different height and produces the depiction of the terrain similar to the contour map. Finally, obtained terrain is triangulated and its surface is rendered, resulting in the geometry-preserving topological landscape. The technique is demonstrated for case of performance data analysis.

4.2 Contributions

The contributions of this chapter are a three-stage technique for creation of geometry-preserving topological landscapes and the capability to conduct comparative analysis of several functions using their topological landscapes.

4.3 Related Work

4.3.1 Contour Tree Visualization

A number of visual models for the contour tree exists. For example, planar and volume graph representations of the contour tree are widely used [40, 50]. In fact, this technique produces the planar graph representation of the contour tree as an intermediary result (detailed discussion is in Section 4.3.3).

While a graph representation is a good starting point, it is often hard to visually derive the required information from it, especially in case of large contour trees. To address this issue, more intuitive visualizations were proposed. The contour nest [59, 58] focuses on the nesting properties of isosurfaces and represents the contour tree as a set of nested rectangles, where rectangle size corresponds to feature size. Another metaphor that provides the requested insight is called *topological landscape*, proposed by Weber et al. [83]. It proved to be useful in several application fields [47, 61, 62, 63], and it is selected as a basis for proposed technique.

In the original work [83], a topological landscape was constructed from the persistence-based branch decomposition [67] of the contour tree [21], such that each branch corresponds to some box element of the landscape. Box elements were arranged using a spiral layout. Later, several modified versions of the original topological landscapes were proposed [61, 62], including the one that uses a tree map layout scheme for box elements [47]. However, both layout schemes (spiral and tree map) ignore feature proximity in the domain. Therefore, the technique proposes a new

layout scheme that consists of three steps: projecting vertices of the contour tree onto the plane; projecting edges of the contour tree onto the plane; drawing contours around the projected contour tree. These steps produce the terrain representation similar to the topographic map, which after additional triangulation step results in the desired landscape.

4.3.2 Dimension Reduction

In the first stage of the proposed layout method, vertices of the contour tree are projected onto the plane. This projection can be achieved by number of dimension reduction techniques (refer to the work by Fodor [36] for an overview). In particular, classical multidimensional scaling [79] is chosen. It is a well-accepted method [42, 34] that preserves relative distances between points, as they are projected from the original domain onto the plane.

4.3.3 Graph Drawing

Once the vertices of the contour tree are fixed, the edges are drawn on the plane. A number of algorithms for drawing a tree on the plane exist, see the book by Tamassia [77] for an overview. Level-based layouts arrange the tree in a hierarchical fashion, starting from the root vertex. Radial layouts draw the vertices of the tree on concentric circles with different radii. However, these methods assume that vertex locations are flexible, which is not true in the given case. Pach and Wenger [65] address the problem of drawing a tree with fixed vertex locations. However, the authors mainly discuss theoretical issues and limitations, such as an upper bound on number of bends per edge. While they describe a high-level scheme for drawing an actual tree, it appears to be theoretical and mostly for illustration purposes. In particular, they allow the spacing between the edges to be infinitely small, leading to potentially cluttered layouts.

Alternatively, it is possible to draw the edges iteratively, using edge routing algorithms. Inspired by the fields of robotics and electronic circuit design, these methods are based on notion

of finding the route between two fixed locations with given obstacles. Usually, these obstacles are the vertex locations and previously drawn edges, given that layout should not contain self-intersections. Drawing an edge as a single straight line is not always possible due to potential obstacles. Instead, an edge can be drawn via polyline, curve, etc. The majority of the edge routing algorithms tries to optimize certain predefined requirements for the route [29], such as minimizing a number of bends or maximizing the smallest bending angle. However, the graph layout is also used as an input for constructing contours. This imposes a requirement that edges should be spread out. Currently, there is no method satisfying this requirement. Therefore, the technique uses a novel edge routing algorithm, which for each edge uses the Voronoi diagram of already drawn parts of the contour tree, hence satisfying the stated requirement.

Once the layout of the contour tree on the plane is computed, it is passed on to the contour construction algorithm, described in Section 4.4.3, followed by constrained Delaunay triangulation and rendering of the surface, to produce the resulting landscape.

4.4 Algorithm

This section presents an algorithm for building the geometry-preserving landscape. The general outline of the algorithm is as follows: First, the contour tree is computed from the data. Then, its vertices are projected onto the plane using multidimensional scaling, and its edges are drawn on the plane using edge routing algorithm. Once the contour tree layout is finalized, the contours extracted from the contour tree are constructed on top of the layout. The resulting contour map is then triangulated and rendered to produce the required landscape.

4.4.1 Contour Tree Computation

Consider a scalar function $f : R^d \rightarrow R$. The contour tree of the function f is computed using the algorithm by Carr et al. [21], which is then branch decomposed based on persistence [67]. Additionally, the spatial locations of contour tree vertices are recorded in the original domain space.

4.4.2 Contour Tree Layout

Let a point set $P_d = \{p_1, p_2, ..., p_N\}$ represent the spatial locations of the contour tree vertices $V = \{v_1, v_2, .., v_N\}$ in d-dimensional space. This point set is then projected to two dimensions using classical multidimensional scaling method $P_d \xrightarrow{MDS} P_2$.

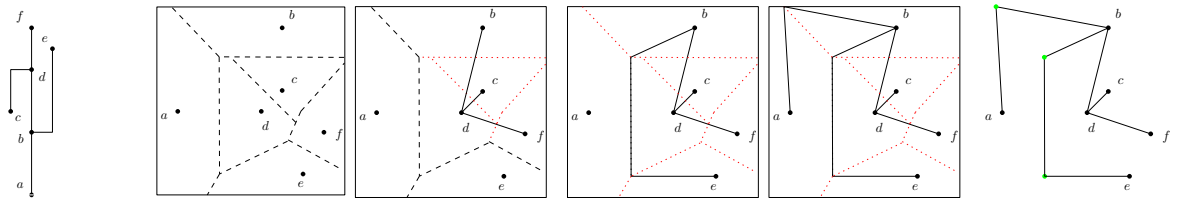

Figure 4.1: **The contour tree layout process.** First the vertices are projected as a point set. Then, the edges are drawn iteratively by connecting the corresponding points. The initial three edges $(f,d),(c,d)$, and (d,b), are drawn as a straight line. Then, the edge (e,b) is routed via Voronoi diagram. Finally, the edge (b,a) is drawn through intersections of Voronoi diagram and bounding box, since no path through Voronoi diagram itself is available.

Once the locations of the vertices are calculated, the edges of the contour tree are drawn iteratively by the following algorithm: For a given edge $e = (v_i, v_j)$, the algorithm constructs a polyline that connects corresponding points p_i, p_j on the plane. For this purpose, the algorithm computes the Voronoi diagram for all existing elements (already drawn vertices and edges), finds an intersection-free path along the edges of the diagram from p_i to p_j, and adds it as a polyline to the layout. If more than one path exists, Dijkstra's algorithm can be optionally used to find the shortest (in terms of line segments or the distance travelled along those line segments) path. To

guarantee the existence of at least one path between any two points, the edge routing is allowed to be done along the bounding box of drawn parts of the contour tree, slightly extended in all directions.

The ordering of the edges by their increasing L_2-norm length on the plane (i.e., direct Euclidean distance between the end points of an edge) was found to lead to fewer and shorter polylines in the final layout. Also, often a few swaps in the ordering can lead to the significant improvements in the layout, thus it is recommended to try several slightly different orderings as well. One example of the swap criterion can be swapping edges with close L_2-norm lengths.

4.4.3 Landscape Construction

At this point, an intersection-free contour tree layout on the plane is computed, given as the set of points P and polylines on the plane. Now, this section describes the algorithm of constructing a landscape from the contour tree layout.

The main idea is to take a branch decomposition of the contour tree, and for each branch (which is an extremum–saddle pair) construct a contour that goes through its saddle and encloses its extremum. If the branch has no children, i.e., it is a leaf branch, and a *triangle contour* is constructed, see Figure 4.3. Otherwise, i.e., if it is a parent branch, its children are sorted in an ascending or descending order, depending on the type of extremum of the parent branch. Then, for each child branch two operations are performed. First, inner contours are constructed as needed (recursively, if it is also a parent branch). Second, an *offset contour* is constructed for the processed part of the parent, see Figure 4.4. One exception is the first child, for which the previously processed part of the parent is an edge (e.g., processed part (f,d) of the parent branch (f,a) in Figure 4.2), hence the triangle contour is used.

The sequence of processing the branches is based on the hierarchical traversal of the given branch decomposition. The processing starts with a root branch. If it is a simple branch, the

Algorithm 4.1 Pseudocode of the function DRAWCONTOURS.

```
b_cur(e_cur, s_cur)
b_cur ← root branch
function DRAWCONTOURS(b_cur)
    S ← all child saddles
    if S = ∅ then
        TRICONTOUR(b_cur)
        return
    if val(e_cur) > val(s_cur) then
        sortDecreasing(S)
    else
        sortIncreasing(S)
    for s ∈ S do
        if s is first then
            TRICONTOUR((e_cur, s))
        else
            OFFSETCONTOUR((e_cur, s))
        DRAWCONTOURS(b_s)
```

triangle contour is constructed and returned. Otherwise, the parent branch handling (described above) is used. This procedure is used recursively for all child branches.

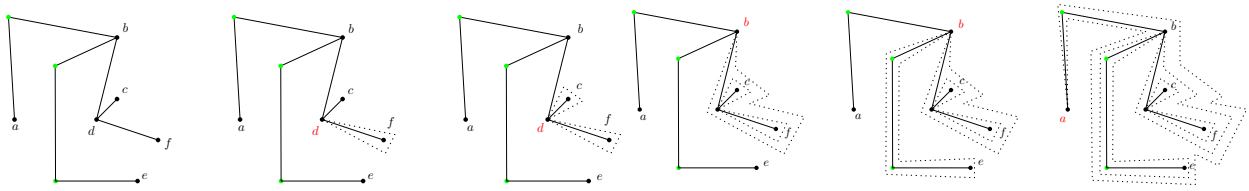

Figure 4.2: **Contour map construction process.** The process starts from a root branch (f,a). It has children, so they are sorted in descending manner, and the process continues with the child branch (c,d), as it corresponds to the highest saddle d. The upper (processed) section of this branch is an edge (f,d), so the triangle contour is used. Then, its child branch (c,d) is a leaf, so the triangle contour is used again. However, the upper section for the next saddle b is complex, so the offset contour is used. Next, the corresponding child branch (e,b) is a leaf, so the triangle contour is used. Finally, the offset contour is applied to the branch (f,a), and that concludes the contour map construction process.

The geometric details of contours and the process of their construction are following: First, each contour is represented as a simply-connected polygon, and each saddle maintains the list of

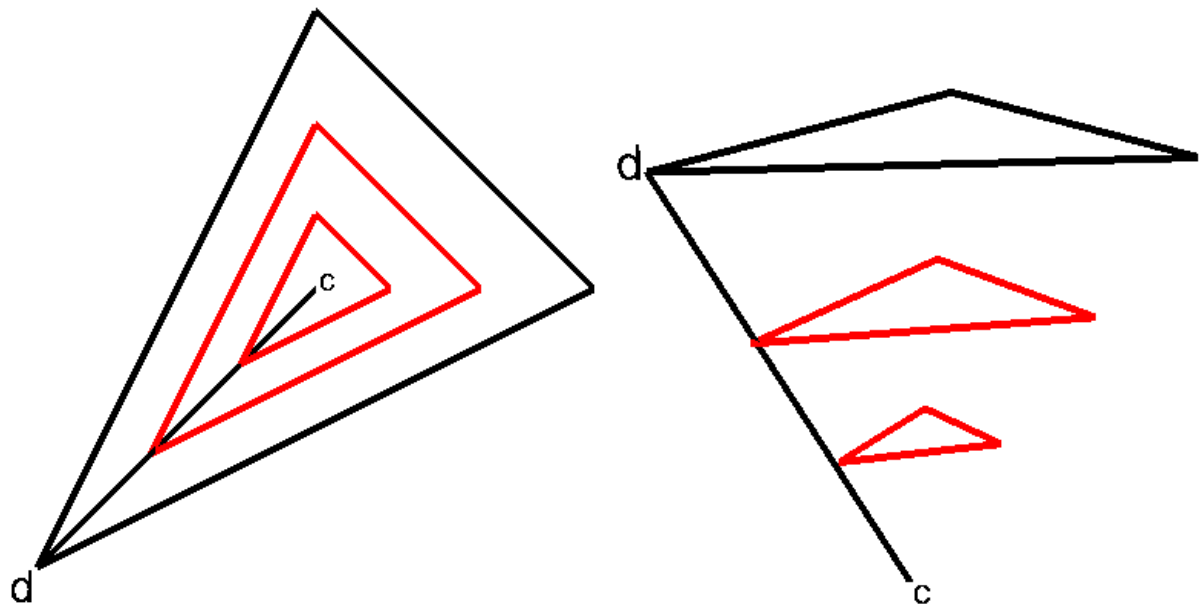

Figure 4.3: **Triangle contour.** Constructing a triangle contour (black) for the branch (c,d) creates a valley with inner contours (red) corresponding to the given branch.

all polygons, associated with it.

The function TRICONTOUR draws a simple triangle around the given edge, see Figure 4.3. If the edge is polyline, a single triangle is still computed for an edge of the same length as the given polyline edge. Then, the triangle coordinates are recomputed at the bends of the polyline edge (see Figure 4.2, contour of the edge (e,b)).

The function OFFSETCONTOUR uses as input an upper/lower (already processed) section of some saddle s. It consists of contours that belongs to the previous saddle s_{prev} and an edge to it from the current saddle, see Figure 4.4. The aim is to enclose these into a single contour. Since each contour of saddle s_{prev} has a corresponding polygon, all these polygons are combined into a single, non-convex, strictly-simple polygon $P_{s_{prev}}$, where strictly-simple means that the area, bounded by the polygon, is simply-connected. Then, a 2D polygon offsetting method, provided by *Computational Geometry Algorithms Library* (CGAL) [18], is applied to polygon $P_{s_{prev}}$. This offsetting produces new, non-convex, strictly-simple polygon $P^{offset}_{s_{prev}}$. Next, the triangle contour is

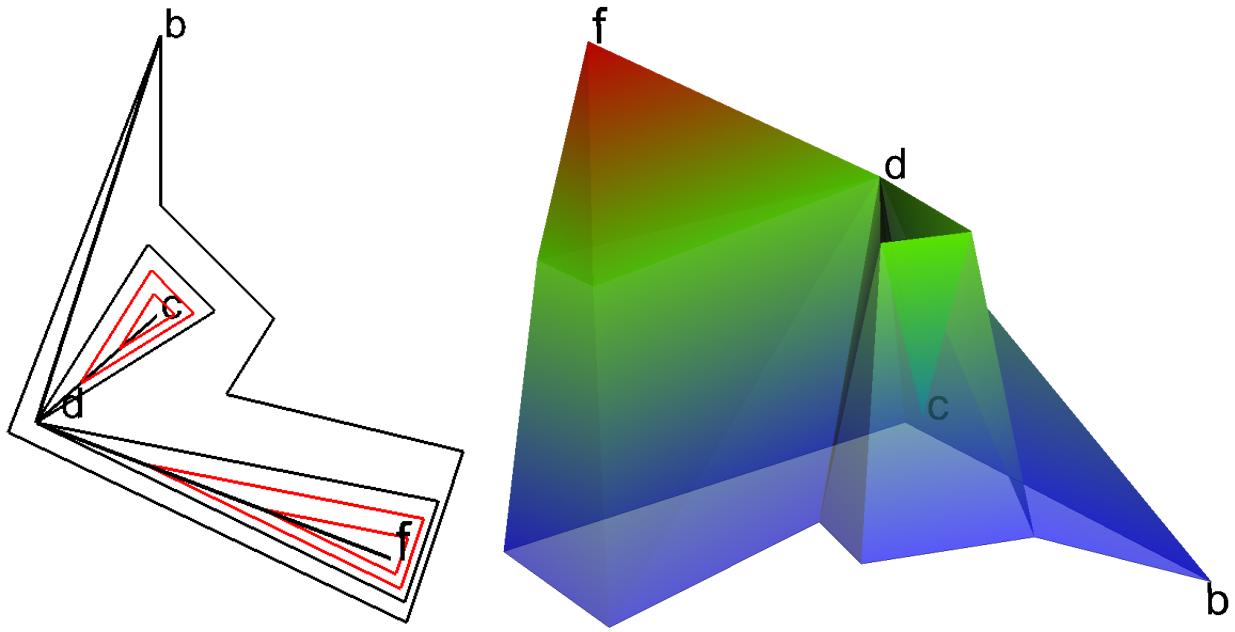

Figure 4.4: **Offset contour.** Constructing an offset contour (black) around the upper section of the saddle b creates a continuation of the peak f with the valley c on it.

constructed for the edge (s, s_{prev}), as this edge is simple. Finally, a 2D polygon join operation (also provided by CGAL) is applied to the computed offset polygon and the triangle together, which produces the output polygon $P_s = P_{s_{prev}}^{offset} \cup T_{(s,s_{prev})}$. This polygon is saved to the list of contours for the current saddle s.

Once all the contours are drawn, the constrained Delaunay algorithm (provided in CGAL) is applied to create a triangulated surface, resulting in a landscape with the given contour tree.

4.4.4 Comparable Landscapes

To produce comparable landscapes for several contour trees $T_1, ..., T_k$, their vertices are combined into one set $V = \{V_1, \ldots, V_k\}$ and the corresponding complete point set is then projected onto the plane $P_d \xrightarrow{MDS} P_2$. After the projection, the algorithm continues with vertices of each contour tree separately. The resulting landscapes are then comparable in terms of the locations of corresponding elements.

4.5 Results

The technique was applied to the problem of tuning a ray casting algorithm on a multicore shared-memory system, based on the study by Bethel and Howison [11] that explores wide range of potential tuning parameters for this algorithm. For the purposes of demonstrating the technique, the following parameters were selected from the study: the work block width $\{1,\ldots,512\}$ and height $\{1,\ldots,512\}$, and levels of concurrency $\{1,2,4,8\}$. One additional parameter was considered, namely the ray sampling method option, which can be either *nearest-neighbor* or *trilinear*. This parameter produces two data sets (one for each option) that are used to demonstrate the comparative capability of the geometry-preserving landscapes.

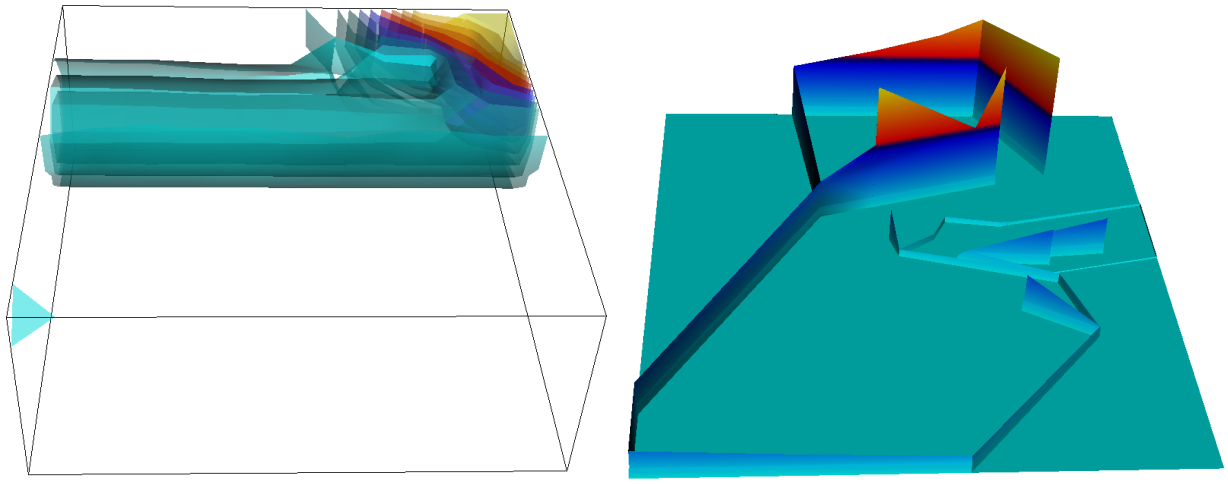

Figure 4.5: **Trilinear ray sampling option data set.** Volume rendering (top) and the corresponding geometry-preserving topological landscape (bottom).

Each data set is obtained by running a ray casting algorithm based on a selected ray sampling strategy and all combinations of other input parameters, and recording the resulting normalized running time, given in milliseconds. High running time corresponds to poor performance, depicted in the landscapes as peaks. Figure 4.5 shows the volume rendering of the performance data set (produced by selecting trilinear ray sampling option), together with corresponding geometry-preserving topological landscape.

It is visually easier to see the correlation between features in the landscape than a direct visualization. For example, from the landscape one can derive the closeness of the peaks, the fact that needs some effort to check otherwise (e.g., by sweeping different isovalues and constructing corresponding isosurfaces, see Figure 4.6). This is an important observation, which suggests to the domain scientist that there exists a subspace enclosing poor configurations to be avoided.

Further, comparable landscapes are produced by combining the projection step for two data sets with trilinear and nearest-neighbors ray sampling options (as discussed in Section 4.4.2). A correlation between two data sets (see Figure 4.6) can be observed, where very high peaks, i.e., particularly poor performing configurations, appear to be close in both data sets.

An interesting observation can be made about moderate peaks (see Figure 4.6). In the case of the trilinear ray sampling option, they are distributed close to high peaks (circle A in Figure 4.6(a)). However, in the case of the nearest-neighbors ray sampling option, they are distant from the main cluster (distinct circles A and B in Figure 4.6(b)). This observation provides an important insight into the underlying behavior of the algorithm and the ray sampling option selection. It can be suggested that, should the parameter ranges be limited, the trilinear ray sampling option is preferable, since unlike nearest-neighbors, in the lower parameter ranges it has no moderate peaks, i.e., moderately poor configurations. This fact can be missed easily in the direct visual exploration or other versions of topological landscapes.

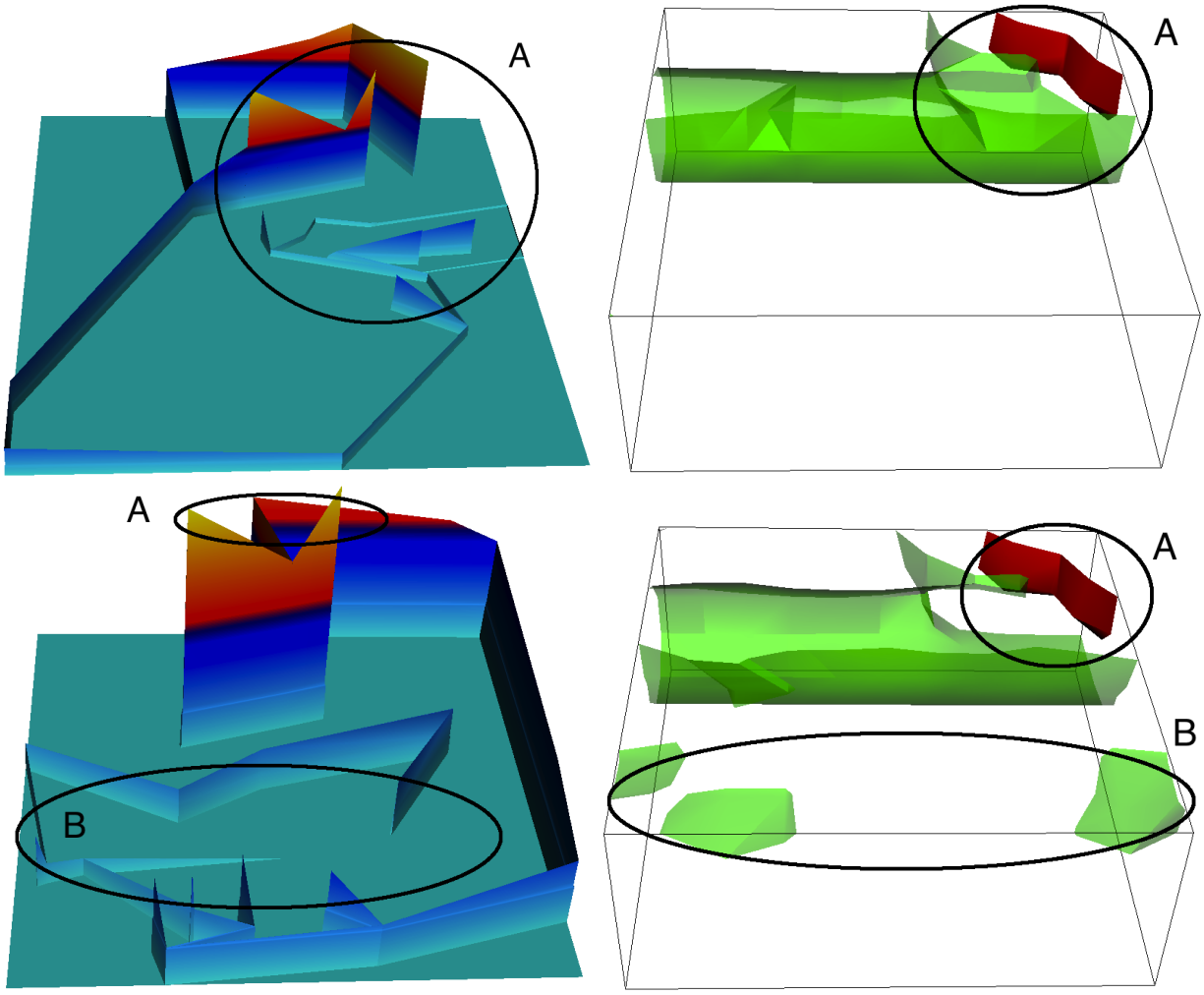

Figure 4.6: **Geometry-preserving landscapes.** Top: Trilinear ray sampling option data set, for which poor performing configurations are closely located (circle A). Bottom: Nearest-neighbor ray sampling option data set, for which poor performing configurations are spread around (circles A, B). From landscapes, one can easily see how close/far the poor performing configurations are from each other. To derive the same observation from the direct view, one has to search for specific, often different, isosurfaces that would encompass such configurations. Furthermore, this search might be impossible when the dimensionality of the function exceeds three.

Chapter 5

Measuring Distance between Merge Trees

The problem of quantifying the error for proposed techniques is addressed in this and the following chapters. This chapter tackles a more general problem of quantifying the difference in topology for any two functions, resulting in a novel distance definition for topological structures called merge trees. In place of two arbitrary functions, the next chapter considers the real and approximated functions, thus allowing the distance to serve as an error measure. This error measure is then used to evaluate various approximation methods, resulting in recommendations, including those that pertain to previously proposed techniques.

5.1 Introduction

This chapter addresses the generic problem of comparing the topology of two scalar functions. Figure 5.1 demonstrates this problem. It presents slightly shifted versions of the same function, colored red and blue. Commonly used analytical distances (e.g., norms of the difference) between these functions would result in a non-zero value, failing to highlight the fact that they have the same sub-level set topology.

One well-established distance that expresses the topological similarity in the above example is

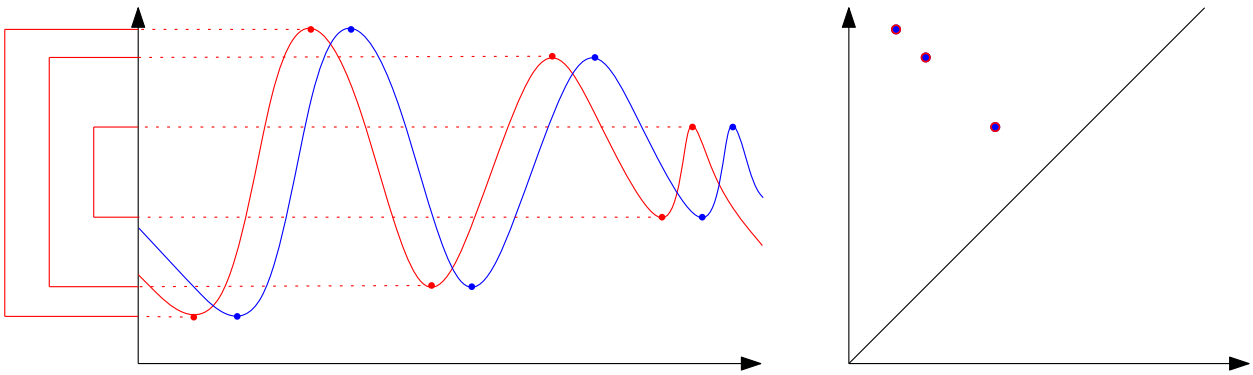

Figure 5.1: Consider two scalar functions (left), one of which is a slightly shifted version of the other. Comparing them directly, e.g., via L_∞ norm, results in a large difference. On the other hand, their persistence diagrams are the same (right), thus capturing the topological similarity of these functions.

the bottleneck distance between persistence diagrams, introduced by Cohen-Steiner, Edelsbrunner, and Harer [24]. Indeed, computing the bottleneck distance for the example in Figure 5.1 gives zero. Originally motivated by the shape matching problem, where the goal is to find how similar shapes are based on similarity of their topology, the bottleneck distance also has an important property — robustness to noise, see Figure 5.2.

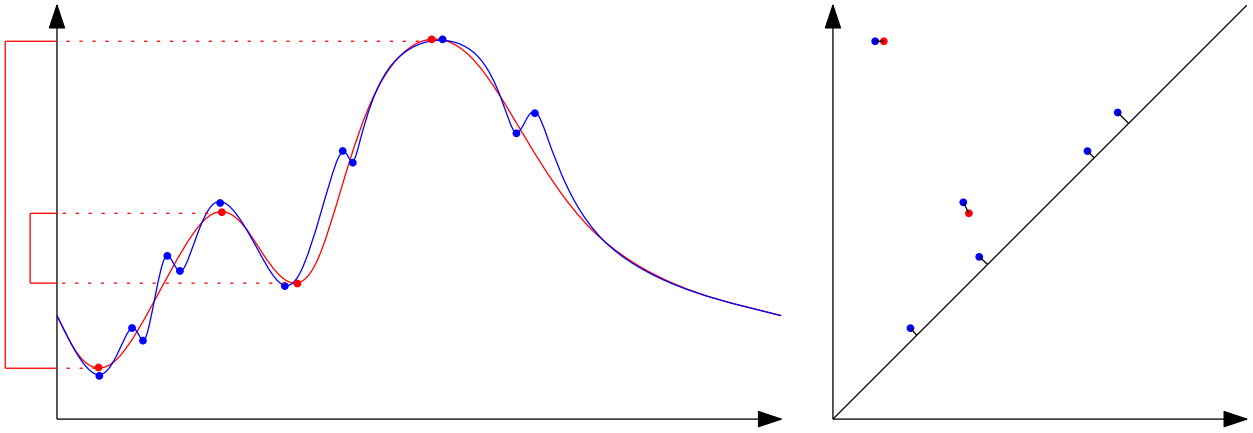

Figure 5.2: Consider two "close" scalar functions, shown on the left, where one contains additional noise. If to construct their persistence diagrams and find the bottleneck distance (which corresponds to the longest black line segment between paired points on the right), the result is small, correctly reflecting the closeness of the functions. In fact, the difference is the same as the level of the noise, which in this example is small.

However, the bottleneck distance does not incorporate sub-level set nesting information, often necessary for analysis. Figure 5.3 shows two functions that differ by the nesting of the maximum m. The bottleneck distance between corresponding persistence diagrams again returns zero. On the other hand, the corresponding merge trees cannot be matched exactly, hinting at a positive difference.

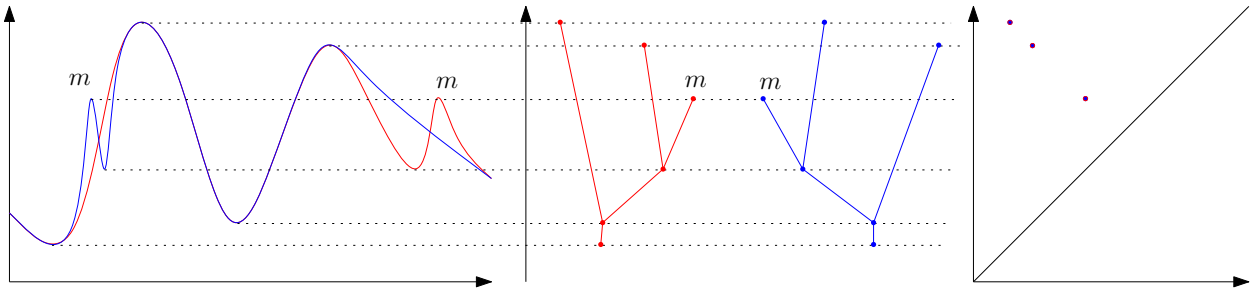

Figure 5.3: Consider two scalar functions on the left. The bottleneck distance between persistence diagrams on the right equals zero, as points of two diagrams overlap. However, comparing corresponding merge trees in the middle reveals a difference, since they cannot be matched exactly. This difference highlights an existence of additional nesting information in merge trees. Quantifying it is the main goal of this chapter.

To resolve this problem, a novel definition of the distance between merge trees is proposed. This distance resembles the bottleneck distance between the persistence diagrams of sub-level sets of the function, but it also respects the nesting relationship between sub-level sets. Furthermore, the proposed distance implicitly distinguishes the noise in the data, similar to the bottleneck distance, resulting in robust measurements resilient to perturbations of the input. This property is crucial when working with scientific data, where noise is a serious obstacle to any analysis.

It is worth noting that the definition of the distance between Reeb graphs was recently introduced in the abstract by Bauer et al. [7], however the discussion on its relationship to the proposed distance is omitted, as the abstract appeared after the completion of this dissertation.

5.2 Contributions

The main contributions of this chapter are: a definition and an algorithm for computing the distance between merge trees; computation of the number of branch decompositions of the merge tree; an experimental comparison between the proposed distance, the bottleneck distance, as well as the L_∞ norm, as applied to analytical and real-world data sets.

5.3 Related Work

5.3.1 Persistence Diagrams

Persistence diagrams reflect the importance of topological features of the function: the larger the difference $d-b$ of any point, the more change to the function is required to eliminate the underlying feature. Thus, persistence diagrams distinguishes between real features in the data and the noise.

Cohen-Steiner et al. [24] proved the stability of persistence diagrams with respect to the bottleneck distance, $d_B(D(f),D(g))$. This distance is defined as the infimum over all bijections, $\gamma : D(f) \to D(g)$, of the largest distance between the corresponding points,

$$d_B(D(f),D(g)) = \inf_{\gamma} \sup_{u \in D(f)} ||u - \gamma(u)||_\infty.$$

Their result guarantees that the bottleneck distance is bounded by the infinity norm between functions:

$$d_B(D(f),D(g)) \leq \|f-g\|_\infty.$$

The bottleneck distance between persistence diagrams is to be used as a comparison baseline for the distance between merge trees.

5.3.2 Distance between Graphs

Graph theory offers a large body of work dedicated to comparing graphs and defining a notion of a distance between them. A common approach to measuring a distance between graphs is based on an edit distance. It is computed as a number of edit operations (add, delete, and swap, if a graph is labeled) required to match two graphs [16], or, in a special case, trees [14]. The edit distance focuses on finding an isomorphism between graphs/subgraphs, while for merge trees one can have two isomorphic trees with a positive distance (see the example in Figure 5.3).

Alternatively, in a specific case of rooted trees, one can consider generalized tree alignment distance [53], which in addition to the edit distance considers minimization of the sum of distances between labeled end-points of any edge in trees. However, it is not clear how to adapt this distance definition for purposes of this evaluation.

5.3.3 Using Topology of Real Functions for Shape Matching

The field of shape matching offers several methods related to this work. Generally, these methods focus on developing topological descriptors by treating a shape as a manifold, defining some real function on that manifold, and computing topological properties of the function. The selection of the particular function usually depends on which specific topological and shape properties that are sought after (refer to an overview by Biasotti et al. [12]).

While majority of mentioned descriptors are not directly related to this work, two topological descriptors use similar approaches in defining similarity measure. First one is called a multiresultion Reeb graph, proposed by Hilaga et al. [51], which encodes a nesting information into nodes of a Reeb graph for different hierarchy resolutions. Here, hierarchy is defined by the simplification of Reeb graph. Second descriptor is based on an extended Reeb graph (ERG), proposed by Biasotti et al. [13]. It starts by computing the ERG of the underlying shape, which is basically a Reeb graph with encoded quotient spaces in its vertices. It couples various geometric attributes with

the ERG, resulting in an informative topological descriptor. In both cases, similarity of shapes is measured by applying a specialized graph matching (based on embedded/coupled information) to descriptors. However, the focus of this chapter is only on the sub-level set topology information, so the straightforward matching algorithm is designed, tailored specifically for this case.

In the work by Thomas and Natarajan [78], the authors focus on discovery of symmetry in a scalar function based on its contour tree. The authors develop a similarity measure between subtrees of the contour tree, which in many regards is similar to the proposed measure. However, their method considers a single pre-processed stable branch decomposition, and focus on discovering symmetry in a sole function.

5.4 Distance between Merge Trees

This section provides a formal definition of the distance between merge trees and give an algorithm (with optimizations) for computing it. In short, to compute the distance between two merge trees, all branch decompositions of both trees are considered, to try to find a pair that minimizes the matching cost between them. Additionally, the details of computing the number of branch decompositions of a merge tree, used in complexity analysis of the algorithm, are given.

5.4.1 Definition

Let K be a simplicial complex; let $f : K \to \mathbb{R}$ be a continuous piecewise-linear function, defined on the vertices and interpolated in the interiors of the simplices. Furthermore, assume all the vertices have unique function values; in practice, this can be simulated by breaking ties lexicographically.

Let T_f be a merge tree of the function f; every vertex of K is mapped to a vertex in the merge tree. Accordingly, every vertex of the merge tree has a degree of either one, two, or more, cor-

responding to a minimum, a regular point, or a merge saddle. Note that given definition works for higher dimensional saddles (degenerate critical points) as well, and they need explicit consideration only in the complexity analysis of the algorithm (Section 5.7). A merge tree obtained by removing all regular vertices is called *reduced*.

A branch decomposition B of a reduced merge tree T is a pairing of all minima and saddles such that for each pair there exists at least one descending path from the saddle to the minimum [67]. Consider a rooted tree representation R of the branch decomposition B, such that the rooted tree representation R is obtained by translating each branch $b = (m,s) \in B$ into a vertex $v \in R$, where m and s are minimum and saddle that form the branch b. The edges of the rooted tree representation describe parent–child relationships between branches, see Figure 5.4.

Given two merge trees, T_f and T_g, consider all their possible branch decompositions, $B_{T_f} = \{R_1^f, ..., R_k^f\}$ and $B_{T_g} = \{R_1^g, ..., R_k^g\}$, respectively; see Figure 5.4.Two auxiliary definitions are described below, to be used for the main definition of the matching of rooted branch decompositions.

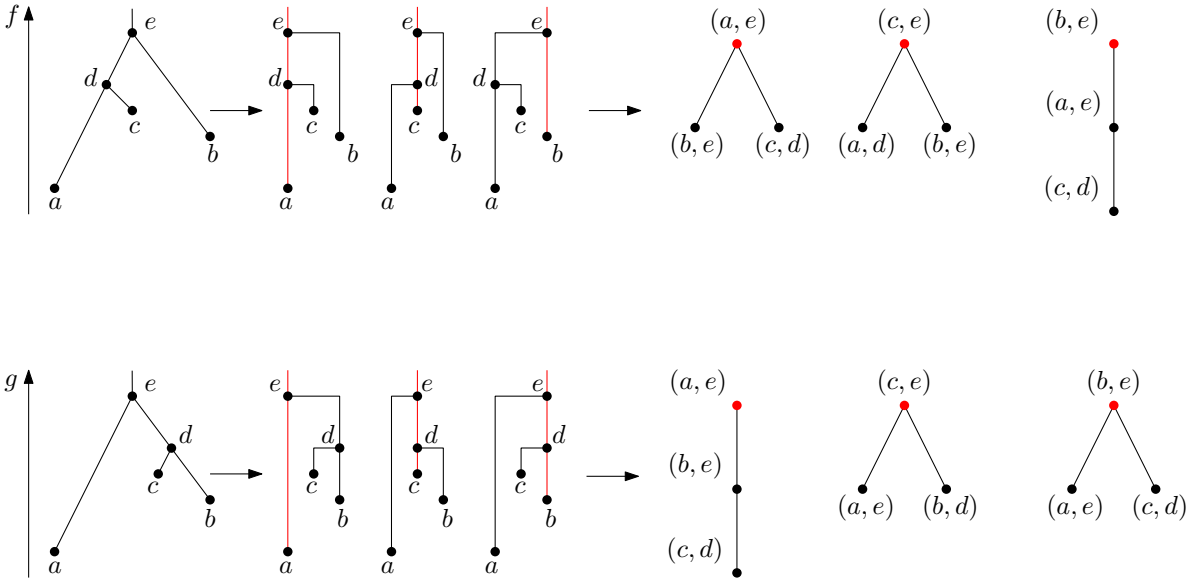

Figure 5.4: Merge trees T_f (top) and T_g (bottom), all their possible branch decompositions, and corresponding rooted tree representations. Root branches are colored in red, demonstrating the mapping of branches to vertices.

Definition 1 (Matching cost)**.** *The cost of matching two vertices* $u = (m_u, s_u) \in R_i^f$ *and* $v = (m_v, s_v) \in R_j^g$ *is the maximum of absolute function value difference of their corresponding elements,*

$$mc(u,v) = max(|m_u - m_v|, |s_u - s_v|).$$

Definition 2 (Removal cost)**.** *The cost of removing a vertex* $u = (m_u, s_u) \in R_i^f, R_j^g$ *is*

$$rc(u) = |m_u - s_u|/2.$$

A partition (M^f, E^f) of the vertices of a rooted branch decomposition R^f is deemed *valid*, if the subgraph induced by the vertices M^f is a tree. Here, the vertices M^f are mapped vertices, while the vertices E^f are reduced vertices. An isomorphism of two rooted trees is said to preserve order, if it maps children of a vertex in one tree to the children of its image in the other tree.

Definition 3 (ε-Similarity)**.** *Two rooted branch decompositions* R^f, R^g *are* ε*-similar, if one can find two valid decompositions* (M^f, E^f) *and* (M^g, E^g) *of their vertices, together with an order-preserving isomorphism* γ *between the trees induced by the vertices* M^f *and* M^g*, such that the distance between each matched pair of vertices and the maximum cost for reduced vertices does not exceed* ε*, i.e.,*

$$\max_{u \in M^f} mc(u, \gamma(u)) \leq \varepsilon \tag{5.1}$$

$$\max_{u \in E^f \cup E^g} rc(u) \leq \varepsilon \tag{5.2}$$

A minimum epsilon, for which the above two inequalities hold, is denoted as $\varepsilon_{min}(R_i^f, R_j^g)$.

Definition 4 (Distance between merge trees)**.** *The distance between two merge trees* T_f, T_g *is*

$$d_M(T_f, T_g) = \min_{R_i^f \in B_{T_f}, R_j^g \in B_{T_g}} (\varepsilon_{min}(R_i^f, R_j^g)).$$

5.5 Naïve Algorithm

The distance d_M is computed by an algorithm that follows Definition 4. In particular, the algorithm constructs all possible pairs of branch decompositions, computes ε_{min} for each pair, and selects the minimum among them.

The algorithm uses a recursive construction of the branch decompositions of a merge tree. The main operation is to pair a given saddle, one by one, with each minimum in its subtree. The algorithm start by pairing the highest saddle s_r with all minima in a tree. Each pair acts as a root branch (s_r, m_i) in the recursive operation. Then, for each child saddle s_j on the root branch, the pairing is recursively repeated until all the saddle–minimum pairs are fixed, producing a unique branch decomposition b_i.

The computation of $\varepsilon_{min}(R_i^f, R_j^g)$ first uses a function ISEPSSIMILAR$(\varepsilon, R_i^f, R_j^g)$, which for a predefined ε, determines whether two branch decompositions match. It starts by setting ε to a high value — for example, the maximum of the amplitudes of the two functions — and performs a binary search to determine ε_{min}.

The function ISEPSSIMILAR is the core of the algorithm. It works by matching the vertices and the edges at each level of the tree. Recall that each vertex $u = (m_u, s_u) \in r_i, v = (m_v, s_v) \in r_j$ is a minimum-saddle pair. There are only two vertices at the root levels of R_i^f and R_j^g, so they are compared directly by checking whether their endpoints can be matched, i.e., whether $\max(|m_u - m_v|, |s_u - s_v|) \leq \varepsilon$. If they cannot, the function ISEPSSIMILAR returns false. Otherwise, all the child vertices are considered (see Figure 5.4). Since now there are several potential matches, a bipartite graph between the child vertices is constructed, such that the edge between a pair of children $u \in R_i^f, v \in R_j^g$ exists if and only if they can be matched within given ε, and ISEPSSIMILAR returns true for their subtrees. Also, a ghost vertex is added for each vertex in the rooted branch decomposition, if it can be reduced within ε. Then, if there is a perfect matching in the bipartite graph, the function returns true; otherwise, it returns false. Note that if one or both of the current

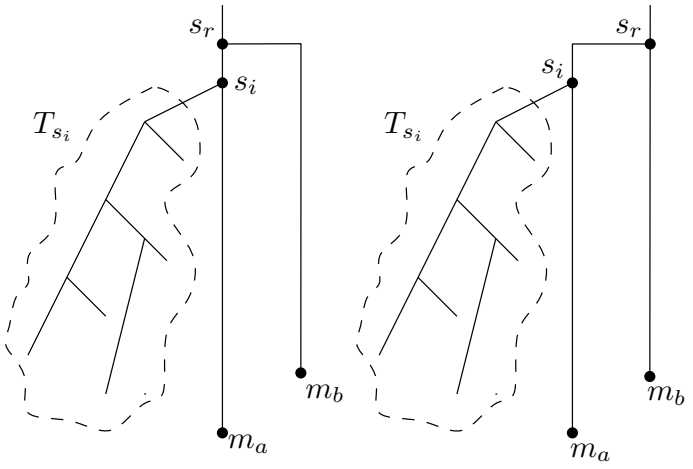

Figure 5.5: As long as s_r is paired with a minimum that is outside of the subtree T_{s_i}, the set of branch decompositions of that subtree remains the same.

pair of children has children of their own, the function ISEPSSIMILAR is called recursively. (The matching is perfect when there exists an edge cover such that its edges are incident to all the non-ghost vertices and do not share any of them.)

5.6 Optimized Algorithm with Memoization

The naive algorithm above has exponential complexity. Indeed, constructing all branch decompositions of a tree takes $O(2^{N-1})$, where N is a number of extrema (details are given in Section 5.7). Therefore, comparing all branch decompositions of two trees would require $O(2^{N+M-2})$ operations, where N, M are the number of extrema in each tree. This cost is unacceptable even for small trees.

Therefore, an optimized algorithm is devised, which combines both construction of all branch decompositions and the recursive comparison of vertices of rooted tree representations of the branch decompositions.

Note that for a given saddle s_i, the set of branch decompositions of its subtree T_{s_i} does not change as long as no minimum from this subtree is paired with a higher-valued saddle (see Fig-

ure 5.5). Thus, if to record the results of comparison for pairs of saddles on each level of branch decomposition, it is possible to reuse them throughout the computation. Since the number of saddles in a tree equals the number of minima minus one, at most $N \times M$ previous comparisons are recorded and used as necessary. More precisely, for any saddle pair $s_i \in T_f$ and $s_j \in T_g$ the following is recorded:

$$match[s_i][s_j] = \begin{cases} 0, & \text{if not compared before,} \\ 1, & \text{if compared as false before, or} \\ 2, & \text{if compared as true before.} \end{cases}$$

Then, the function IsEpsSimilar takes as input the highest saddles s_f, s_g and the current ε. It loops over all minima reachable from the current saddle and sets the minimum–saddle pair as the current root branch. It collects all child saddles and constructs a bipartite graph such that there is an edge between a pair of saddles, when they can be matched within a given ε. The latter condition is determined recursively by calling IsEpsSimilar function on the new pair of saddles. Once the recursion reaches the leaf branches, it compares them directly. As the algorithm progresses, more recorded comparison results become available, decreasing the number of required recursive calls.

Finally, the algorithm uses the function IsEpsSimilar to find the ε_{min}, which is the target distance d_M. This is done by using the binary search algorithm for a range between zero and selected high value, e.g., the absolute difference between the global maximum and minimum.

Note that between iterations of the binary search, parts of the array *match* stay unchanged. Specifically, if two subtrees are compared and recorded as unmatchable for the current ε, and in the next iteration the value of ε decreases, the two subtrees remain unmatchable. Similarly, if two subtrees match for some ε, they remain matched, if ε increases in the next iteration.

Proposed optimization reduces the run time complexity from exponential to polynomial. Indeed, the function IsEpsSimilar performs $N \cdot M$ outer operations, multiplied by the sum of pro-

Algorithm 5.1 Optimized algorithm uses the function ISEPSSIMILAR, described below, for a binary search of the smallest ε.

```
s_f, s_g ← highest saddles of T_f, T_g

function ISEPSSIMILAR(s_f, s_g, ε)
    if match[s_f][s_g] ∈ {1,2} then
        return match[s_f][s_g]
    for all minima m_f ∈ T_f do
        for all minima m_g ∈ T_g do
            if max(|m_f − m_g|, |s_f − s_g|) > ε then
                match[s_f][s_g] ← 1
                return false
            for all c_f ∈ {child saddles of s_f} do
                for all c_g ∈ {child saddles of s_g} do
                    if ISEPSSIMILAR(c_f, c_g, ε) then
                        bipartite[c_f][c_g] ← true
                    else
                        bipartite[c_f][c_g] ← false
            found ← FINDMATCH(bipartite)
            if found then
                match[s_f][s_g] ← 2
                return true
    match[s_f][s_g] ← 1
    return false
```

cessing saddles $n_{c_f} \cdot n_{c_g}$ and the complexity of a maximal matching algorithm $(n_{c_f} + n_{c_g}) \cdot n_{c_f} \cdot n_{c_g}$. Note that the latter complexity dominates the former term. Hence, assuming that lookup of previous results is done in a constant time via memoization, the resulting run time complexity of the function ISEPSSIMILAR is $O(N^2M^2(N+M))$. This complexity gets multiplied by the number of iterations of the binary search algorithm, which can be considered logarithmic, given a reasonable selection of the search range and the precision. The worst-case memory complexity of the optimized algorithm is $O(N*M)$, less prohibitive than its run time complexity.

5.7 Number of Branch Decompositions of a Merge Tree

This section provides the details of computing the number of branch decompositions of the merge tree, used for the naive algorithm complexity analysis in the beginning of Section 5.6. The number of branch decompositions $P(N)$ for a merge tree with N minima is computed in two steps. First, the number $P(N)$ is computed for the case when the merge tree is binary, in which case the tree has maximum possible number of saddles. Second, it is shown that for fewer saddles the number $P(N)$ only decreases, leading to the worst case $P(N) = 2^{N-1}$ branch decompositions for any merge tree.

Theorem 1. *The number of branch decompositions of the binary merge tree with N minima equals* $P(N) = 2^{N-1}$.

Proof. For any saddle s, the number of branch decompositions in its subtree is $P_s = 2 * P_{c_1} * P_{c_2}$, where c_1 and c_2 are the children of the saddle s. Indeed, if the saddle s is paired with a minimum in a subtree of child c_1, then for each such pairing, the all the possible branch decompositions of a subtree of child c_2 gives $P_{c_1} * P_{c_2}$ possibilities. Symmetrically, for the child c_2, there are $P_{c_2} * P_{c_1}$ possibilities.

Now, using this fact, the following proof can be constructed by induction:

- For the base case of $N = 1$, the number of branch decompositions is one. On the other hand, $P(1) = 2^{1-1} = 1$. Hence, the formula holds.
- Assume that for all $N = 1, \ldots, k$ the formula $P(N) = 2^{N-1}$ remains true.
- Now consider the case with $N = k+1$, for which it should be proven that $P(k+1) = 2^k$. For the root saddle r of the tree with $k+1$ minima, remember that $P_r = 2 * P_{c_1} * P_{c_2}$. If to denote the number of minima in the subtree of the child c_1 as $i \in [1,k]$, with $k-i+1$ denoting the number of minima in the subtree of the child c_2, the term $P(k+1)$ can be expanded as

$P(k+1) = 2 * P(i) * P(k-i+1)$. Since both i and $k-i+1$ are not greater than k, $P(i)$ and $P(k-i+1)$ can be substituted in accordance with assumptions for $N = 1, \ldots, k$:

$$\begin{aligned} P(k+1) &= 2 * P(i) * P(k-i+1) \\ &= 2 * 2^{i-1} * 2^{k-i+1-1} \\ &= 2 * 2^{i-1+k-i+1-1} = 2^k. \end{aligned}$$

□

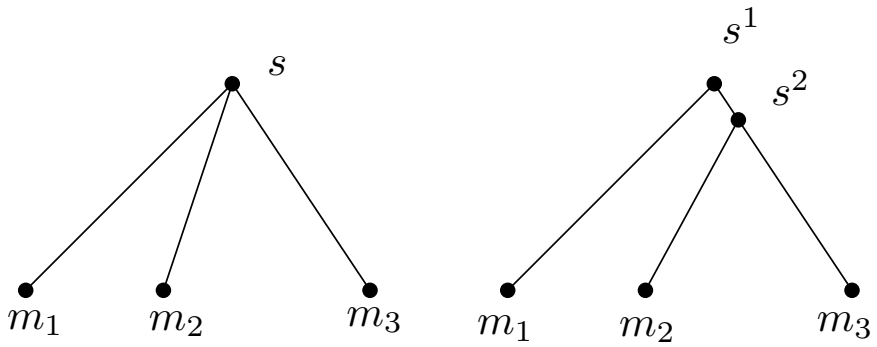

Figure 5.6: Splitting the higher degree (> 2) saddle always creates more branch decompositions. For example, the saddle s induces three branch decompositions, however after being split into two saddles s^1 and s^2, now there are four branch decompositions.

Now, consider the case with a number of saddles less than $N-1$, i.e., the merge tree has saddles with degree higher than two. The number of minima N remains the same, while some of the saddles have more than two children. Then any saddle of degree $d > 2$, can be split into $d-1$ saddles of degree two, such that the structure of the tree changes only around the selected saddle, see Figure 5.6. Such split leads to 2^{d-1} possible branch decompositions instead of the d for the selected saddle. Since $d > 2$, the inequality $2^{d-1} > d$ holds true, which means having degree two saddles always leads to more branch decompositions. At the extreme, if all saddles become degree two, the merge tree becomes a binary merge tree, for which the number of branch decompositions was already computed as 2^{N-1}.

Chapter 6

Measuring Error in Approximating Sublevel Set Topology

This chapter presents a new approach to evaluating the error for approximated functions. The approach adapts two distances (the distance between merge trees from the previous chapter, and the bottleneck distance between persistence diagrams) as error measures to evaluate approximation methods. In particular, the focus is on methods, used to approximate a function, and the performance of these methods in keeping the distance between the approximated function and the ground truth function small, i.e., keeping the error measures low.

6.1 Introduction

In practice, scalar functions are often approximated from discrete samples, acquired from simulations or experiments. Sampling introduces an approximation error, which causes a loss of topological information, often important for the end analysis.

The following scenario illustrates the problem. Consider a potential energy function of a molecule, previously presented in Chapter 3. In particular, choose the function, obtained by the

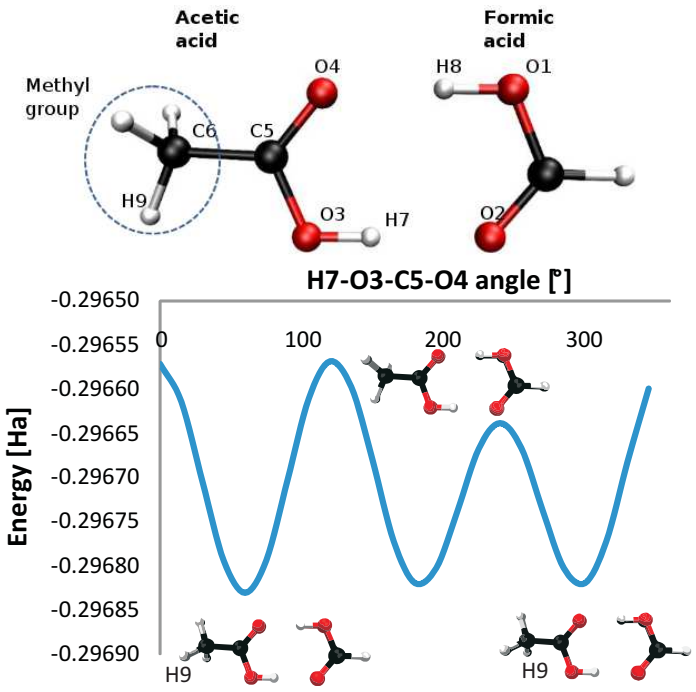

Figure 6.1: (Left) Molecule of dimer of formic and acetic acid. (Right) Rotation of its methyl group produces the potential energy function.

rotation of the methyl group in the dimer of formic and acetic acid (DFA), see Figure 6.1.

The minima of this function correspond to the stable states of the DFA molecule. Hence, they can be used to better understand the structure of the molecule and its evolution during chemical reactions. However, one is only able to measure or to simulate a discrete set of samples of this function. Figure 6.2 shows two sets of samples and their approximated functions; the algorithm tries to extract information about the original minima.

If one is interested only in the correct number of minima (i.e., stable states), one can quantify the error as the ratio of the missing to the correct minima. The real function has three minima. Thus for the first approximated function the error would be measured as $0/3 = 0$. However, the second approximated function misses two minima, leading to the error of $2/3 = 0.67$. The latter non-zero error reflects the fact that it missed some of the topological information of interest, in this particular case, the number of minima. The error is only considered in terms of the number of

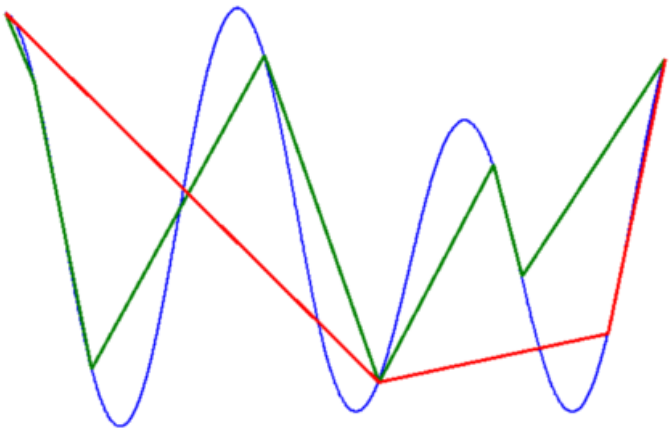

Figure 6.2: Although the first approximated function (green) deviates from the real function (blue), it still preserves the correct number of minima — three — thus bearing no error on the count of minima. However, the second approximated function (red) contains only one minimum, leading to an error of $2/3$.

minima, not positions or function values.

The sources of error in this example, i.e., the density and the distribution of samples, have separate effects on the error. Indeed, even for a large number of samples, a bad distribution can cause a large error. On the other hand, a small number of points would not sample the domain well enough, regardless of their distribution. To limit the scope, only uniformly random distributions of samples are considered, thus shifting the focus mainly on the *sampling density error*.

Another factor that causes the error is the approach to connecting the samples and approximating the function in between. The previous one-dimensional example intuitively connected samples along the only axis and used linear interpolation in between the samples. However, with functions on multi-dimensional domains, the strategy of how to connect samples becomes less clear. Consider a two-dimensional function in Figure 6.3. Given a set of samples, they can be connected into different meshes, each of which might lead to a different error. For the purpose of the evaluation, the interpolation scheme is fixed to be linear (1D interpolation along the edges of the mesh for any dimensionality), and the focus is on the *mesh error*.

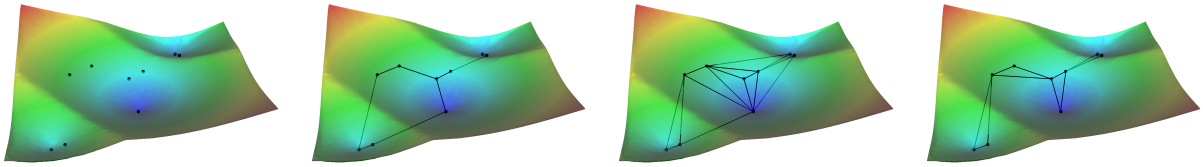

Figure 6.3: (Left) Given a set of nine samples of a two-dimensional function, one can construct various meshes with potentially different approximation error. For example, following meshes are generated by (middle left) Gabriel graph, (middle right) Delaunay triangulation, (right) K-Nearest-Neighbors, with $K = 2$, the lowest value for which the mesh becomes connected.

So far, a topological feature of interest was minimum. Minima, or, more generally, extrema, were the focus in an evaluation study by Correa and Lindstrom [26], where the authors developed an error measure called F-measure. This investigation goes beyond the extrema and focuses on topological features related to *sub-level sets* of the function, trying to quantify the error in their approximation. The investigation considers more comprehensive error measures, and uses them for the evaluation of approximation methods, which is the main goal of the investigation.

Given a compact subset X of the Euclidean space R^d and a threshold c, the sub-level set of a function $f : X \to R$ is the set of all points in the domain of the function whose value does not exceed c; formally, it is the set $f^{-1}(-\infty, c] = \{(x_1, ..., x_d) \in X \mid f(x_1, ..., x_d) \leq c\}$. Consider two topological approaches to describing the evolution of sub-level sets. The first is based on a *merge tree*, which tracks the evolution of components of sub-level sets by recording their birth and merge events as the threshold value c increases. The second is a *persistence diagram*, which records the lifespan of components of sub-level sets. In particular, it records all pairs (b, d) such that a component born in $f^{-1}(-\infty, b]$ dies in $f^{-1}(-\infty, d]$ by merging with an older component.

To quantify the error for these topological structures, one need a notion of distance between a measure computed from the real function and the one from the approximation. In case of persistence diagrams, a natural choice is the *bottleneck distance*, introduced by Cohen-Steiner, Edels-

brunner, and Harer [24]. This distance is used as a basis for the first error measure. In the case of merge trees, the *distance between merge trees* [10], presented in Chapter 5, is used as the second error measure. Further, these distances are denoted as d_B, d_M, and corresponding normalized error measures are denoted as $\varepsilon_B, \varepsilon_M$.

The proposed error measures require the real function to be known, thus they cannot be used if only a set of samples are available. However, they provide a powerful evaluation capability, given both the real function and its sampling, to compare common approximation methods. Presented work evaluates the approximation quality of different types of meshes, as several parameters vary, e.g., the density of the sampling. Such evaluation reveals common types of problems and leads to recommendations of preferred types of meshes under various conditions.

The key advantage of this approach, when compared to F-measure of Correa and Lindstrom [26], is that error measures implicitly distinguish the noise in the data, resulting in robust measurements resilient to perturbations of the input. This property is crucial when working with scientific data, where noise is a serious obstacle to any analysis.

6.2 Contributions

The contributions of this chapter are:

- Evaluation of common mesh construction methods using error measures ε_B and ε_M that have implicit ability to distinguish the noise.
- Targeted evaluation of scalar functions, defined on curved domains (manifolds), embedded in higher dimensional space.
- Recommendations for each type of mesh based on the evaluation results.
- Application of obtained recommendations to proposed techniques from Chapters 3, 4.

6.3 Related Work

6.3.1 Sampling and Mesh Types

Well-known sampling strategies, *regular* and *random*, are often used to generate real-world data sets. Most of the random sampling methods were developed in statistics (see [57, 17]) as a response to the need for rigorous representative selection of the subsets from the full set. In case of scalar functions, this translates into the selection of discrete points in the domain of the function. Regular sampling is a selection of points in the domain of the function with a fixed step in each dimension. Random sampling is a probabilistic selection of points within the domain, usually with requirements like the uniform density of the sample or variability. This approach uses randomly sampled scalar data.

Since the sampled scalar data can lie in higher dimensions, approximation schemes that try to interpolate (using polynomials or splines) inside the whole domain by decomposing it into cells are costly, so a simple approximation scheme that connects the samples via a mesh and linearly interpolates the function along the edges of the mesh is considered. In some cases, cell decomposition methods are used for the mesh generation purposes.

There are a number of methods that work with non-regular sampling. One example is the Delaunay triangulation. It connects all the sample points into simplices of the same dimensionality as the underlying domain and, for each, guarantees that there are no sample points inside its circumsphere. It is frequently used because it produces average sized simplices. The evaluation of a mesh, generated by this method, is presented among others.

Computing triangulated meshes can be expensive. So it is common to use sparser meshes. One example are meshes generated by empty region graphs [19]. As the name suggests, these meshes are constructed by connecting any two sample points whenever their "region" is empty, i.e., free of other sample points. Depending on the definition of "region," they assume different names; for

Figure 6.4: The edge between any two points p and q exists, if their "region" is empty, i.e., free of other points. Left: For a Gabriel graph, the "region" is a circle with an edge (p,q) as a diameter. Right: For a relative neighborhood graph, the "region" is an intersection of two half-circles, with centers at p and q, and a radius (p,q).

example, *Gabriel graph* or *relative neighborhood graph* meshes, see Figure 6.4. The former was used in the work by Oesterling et al. [62]. The latter was suggested as a primary choice for data with prohibitive dimensionality or size by Correa and Lindstrom [26]. These two types of meshes are included into the evaluation.

Finally, the evaluation considers meshes that are generated by parameterized neighborhood graphs. First one is a *k–nearest neighbors* graph that constructs the mesh by connecting each point with its k nearest points. This type of mesh is used in number of studies in scalar field topology [76, 47]. Since there is no fixed way of selecting the value of k, so by default the mesh generated by the minimum value of k such that the mesh is connected is used. Second one is a mesh generated by Vietoris–Rips complex [81], as a comparison alternative for k–nearest neighbors mesh. This complex is constructed by placing a simplex for every set of points within pairwise distance r from each other. When the underlying data lies on some sub-manifold of the ambient space, a Vietoris–Rips complex serves as an approximation of this manifold. Similar to k–nearest neighbors mesh, as a default the minimum value of r that makes its mesh connected is used.

6.3.2 Evaluation of Mesh Types

Although different combinations of sampling strategies and meshes are often used, the question of how far the approximated topology diverges from the original is rarely addressed. Usually,

it is just assumed that the selected combination of sampling and mesh construction sufficiently approximates the original function.

An explicit attempt to address this question appears in the work of Correa and Lindstrom [26], who evaluated different mesh types in terms of the correct extrema discovery. They quantified the number of false positive and true negative cases of extrema classification and computed normalized harmonic mean, which they called the F-measure. They also proposed improved *relaxed* mesh types, given the results of evaluation based on the F-measure and additional observations. Such mesh types are obtained by relaxing the containment requirement for different empty region graphs [26]. So the evaluation also considers relaxed versions of the relative neighborhood graph and the Gabriel graph.

Small perturbations of the data can generate an arbitrarily large number of false extrema. On the other hand, it is easy to recognize most of them as noise precisely because they result from a small perturbation, and, therefore, their persistence is low. Moreover, they can be explicitly eliminated with a small change of the function [33]. By focusing only on the number of extrema, F-measures overlook this crucial distinction: not all false extrema are created equal. In contrast, this chapter considers the persistence of the extrema, and automatically de-emphasizes the noise in the evaluation, thus effectively measuring how well an approximation preserves important topological features of the function.

6.4 General Evaluation

This section provides an evaluation of mesh types using the approximation error measures ε_B and ε_M. In particular, first it describes a class of functions serving as a ground truth in evaluation. Second, it provides details of normalizing the distances d_B, d_M (defined in Section 6.3) for each concrete function, which makes the error measures scale-independent and thus comparable. Finally, it evaluates each mesh type and summarizes recommendations.

6.4.1 Ground Truth Function

A class of functions that serve as the ground truth is designed based on the following considerations: First, it must be possible to compute a merge tree and a persistence diagram of a ground truth function, as a part of the error measures calculation. Also, to normalize the error measures, function values of the global maximum and global minimum must be known. Finally, it is crucial to be able to change the number of maxima and the dimensionality of the function for more comprehensive evaluation. A definition of one simple class of functions that satisfies the listed considerations is provided below.

Definition A parametric function is given as an upper-envelope of cones, with a fixed slope coefficient k, and apexes at a set of maxima M with values $val(m \in M)$:

$$f_{gt}(p) = \sup_{m \in M} (val(m) - k * d(m, p)),$$

where $d(m, p)$ is the Euclidean distance between two points, see Figure 6.5.

The computation of the merge tree (the first consideration) of such function can be done in the following manner: First, the set M of all the maxima with locations and function values is chosen (possibly randomly). Since each maximum $m_i \in M$ induces an uniformly decreasing region R_i around it, for any pair of regions R_i, R_j their merge saddle can be determined (if one exists). This is done by finding the highest point of the shared boundary. Since all slopes are fixed, this point lies on the line that connects two maxima m_i, m_j. To compute the coordinates of that point, i.e., the merge saddle, it is enough to use simple geometry, knowing that the distances from each maximum to the merge saddle, are proportional to function values of two maxima with the fixed slope. The function value of the merge saddle can be computed in a similar way. Once all maxima and the merge saddles are determined, a graph $G = \{M, E\}$ is constructed, such that each maximum is connected to the neighboring maxima through the corresponding merge saddles. Computing the

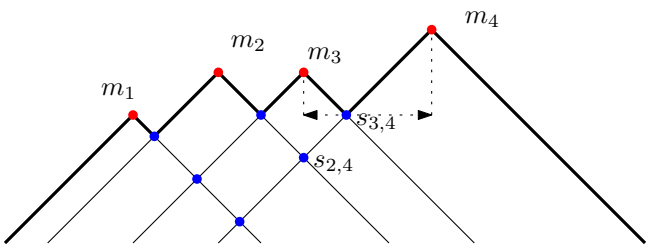

Figure 6.5: The ground truth function is indicated by the bold black line, a supremum of maxima generators. Red nodes are maxima and blue nodes are possible saddles. Note that saddle $s_{3,4}$ belongs to the given function, while $s_{2,4}$ do not. A function value and coordinates of any saddle can be explicitly computed from the information about the maxima (dotted lines).

merge tree of the graph G produces the required result.

For the second consideration, note that all maxima are defined, hence the global maximum can be determined by direct comparison. The global minimum is computed by considering the boundaries between regions. Indeed, the lowest point of any boundary segment is at the boundary ends. Then, by comparing all the boundary intersection points (ends), including those with the domain boundaries, one can find the global minimum. Note that the number of maxima and the dimensionality are parameters of the parametric function, thus they can be directly controlled.

A concrete set of functions that would serve as the ground truth is selected in the following manner. First, the number of maxima generators are set to $\{3, 20, 50\}$, and the required number of maxima are created within the domain. Resulting functions constitute core set of ground truth functions. Whenever necessary, the number of dimensions is varied. For consistency purposes, two additional random sets of maxima per each number of maxima are generated. Unless explicitly stated, the default number of maxima is 20, the default number of dimensions being two.

6.4.2 Measure Normalization

By default, the distances d_B and d_M are computed in function value units and, thus, can vary greatly for different functions. To eliminate this dependency, the distances are normalized based on their value range.

For the distance between merge trees, the smallest possible value is zero. To find the largest possible value, consider the difference between global extremum m and root saddle r of a merge tree T. Then, considered difference equals $d_{T_g} = |m_g - r_g|$ for the merge tree T_g of the ground truth function g, and $d_{T_a} = |m_a - r_a|$ for the merge tree T_a of the approximated function a. The inequality $d_{T_g} \geq d_{T_a}$ always holds, since the function a is only a sampling of the function g. On the other hand, the distance between merge trees is computed based only on function value differences between vertices of merge trees, thus it cannot exceed d_{T_g}. Hence this difference is used to normalize proposed error measures.

6.4.3 Evaluation and Analysis

The evaluation is conducted for each ground truth function, defined earlier, by generating a set of samples, and selecting its subsets of increasing size. Each subset is approximated as a function using the selected type of mesh, and the error measure. The main premise of the evaluation is that a better mesh produces lower error.

As discussed in the Section 6.3, the evaluation considers five types of meshes in the evaluation: Delaunay mesh (DEL); relative neighborhood graph mesh (RNG); Gabriel graph mesh (GAB); k-nearest-neighbors mesh (KNN); and Vietoris-Rips mesh (VR). These acronyms to be used throughout the dissertation.

Evaluation starts off by computing the error measures for DEL, RNG, and GAB meshes as the number of samples increase, see Figure 6.6. For all meshes, as the number of samples increases, error measures decrease. Once the number of samples gets really high, the error measures start to flatten. This behavior is expected, as at the beginning samples start to discover maxima regions, thus keeping the error high and spiky. Afterwards, samples get dense enough to approximate each maximum region, thus error start to decrease smoothly. However, since the probability that samples would hit all the critical points exactly is very low, the error flattens.

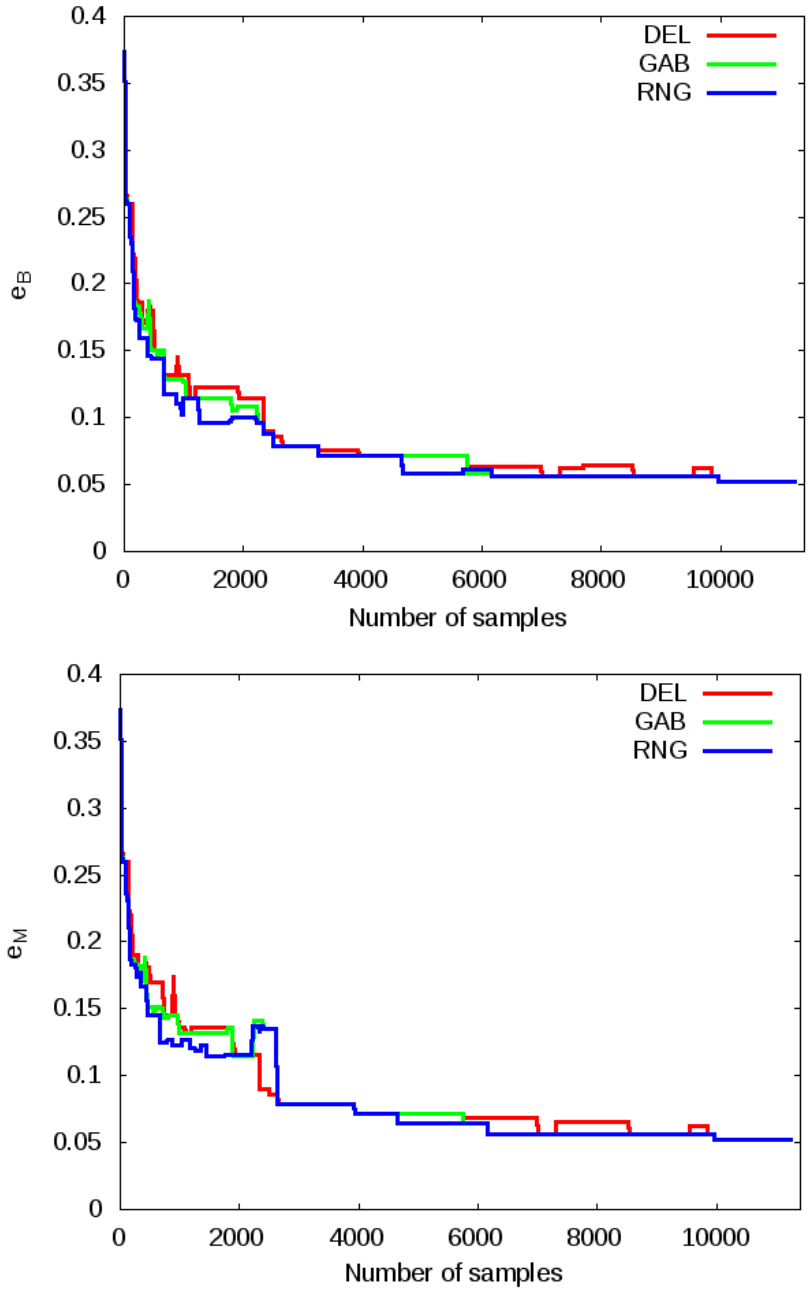

Figure 6.6: The increase of the number of samples leads to the decrease of error values of $\varepsilon_B, \varepsilon_M$.

Delaunay Mesh. Overall, small error fluctuations are expected due to the random nature of the sampling. However, note an irregular increase of the error for the DEL mesh in the Figure 6.7, between 2100 and 6900 samples, while the error for the GAB and RNG meshes stay consistently

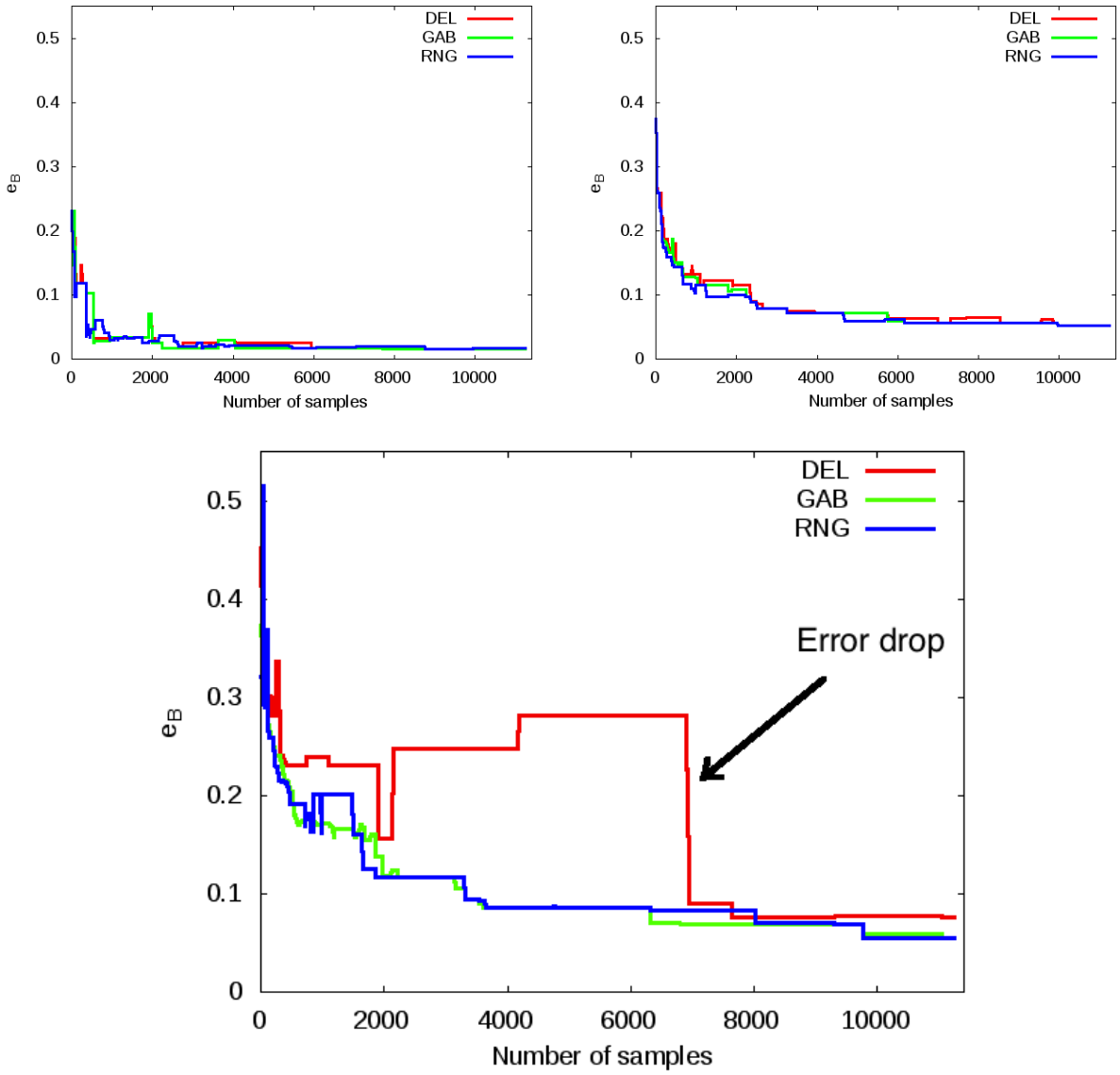

Figure 6.7: The error measure ε_B for functions with varying number of maxima. As expected, more maxima slow the error decrease rate, as it takes more samples to discover all maxima. Indeed, for max3 function (left), the error drops below 0.1 around 400 samples, for max20 function (middle) around 1000 samples, for max50 function (right) around 3000 samples. Note an irregular increase of the error for the DEL mesh, between 2100 and 6900 samples.

low. Furthermore, the DEL mesh performs comparably worse in higher dimensions as well, see Figure 6.8. Detailed investigation of the meshes, shown in Figure 6.9, reveals long edges along the boundary of the domain for the DEL mesh, a potential source of the irregular error increase.

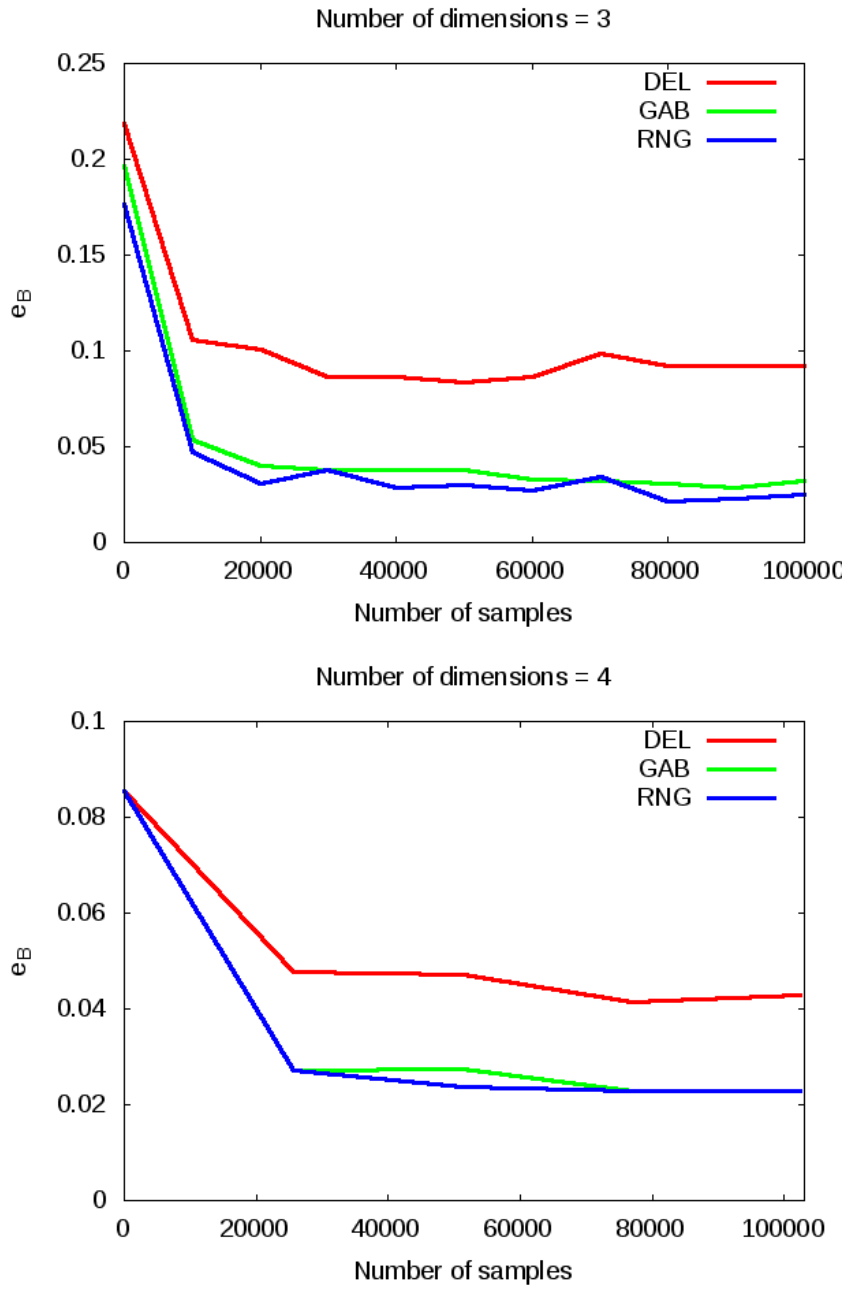

Figure 6.8: Increase in dimensionality does not affect the expected decreasing error trend. Again, the performance of the DEL mesh is relatively bad.

Indeed, these edges over-connect maxima, thus decreasing the persistence of some of them, leading to an error increase. One case of the error drop, shown in Figure 6.9, reveals that the problematic

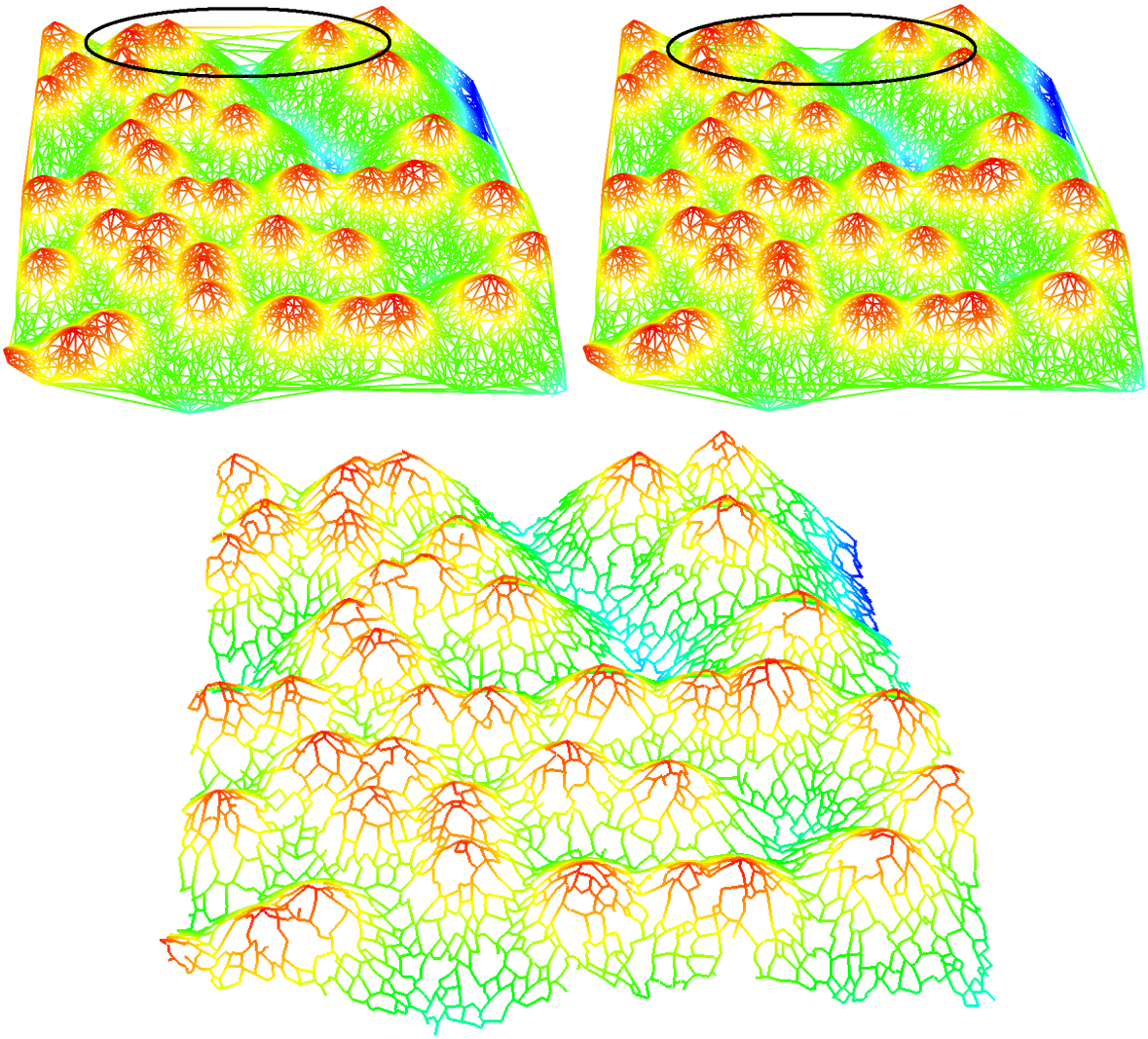

Figure 6.9: Detailed investigation of an irregular increase of the error in Figure 6.4.3. The DEL mesh before (left) and after (middle) the error drop around 6900 samples. The DEL mesh before the error drop has many, potentially over-connecting, long edges along the boundaries of the domain. The error drop happens, as some of them are reduced due to the newly inserted samples. For comparison, the RNG mesh before the error drop is provided (right), showing that it is not affected by the problem of over-connecting boundary edges.

edges disappear with insertion of more samples, thus leading to error decrease. The GAB and RNG meshes are less affected by this problem, due to shorter edges on average, e.g., see Figure 6.9.

The problem of badly shaped boundary triangles in Delaunay triangulation is well-known; it is reported in context of scattered data interpolation by Lasser and Stüttgen [56]. In the same

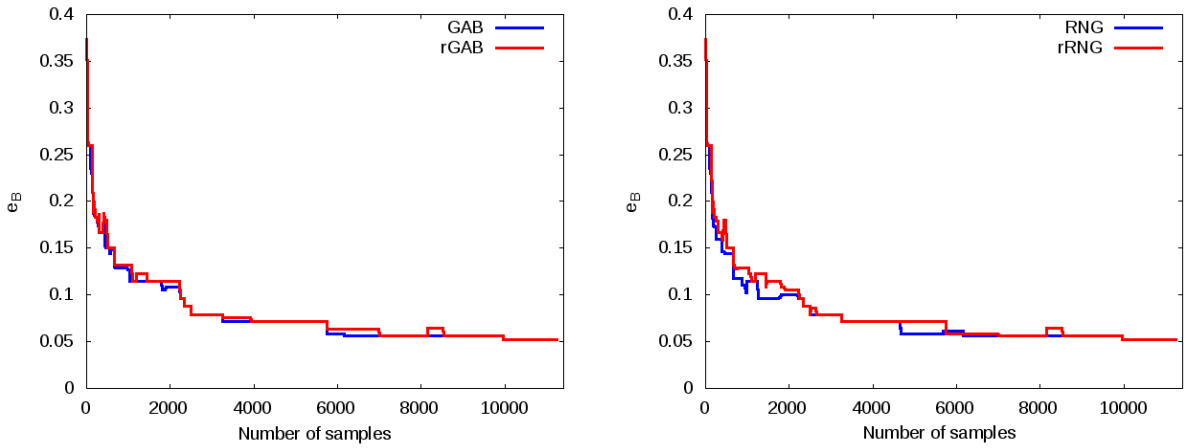

Figure 6.10: Evaluation of the relaxed versions of the GAB and RNG meshes shows no significant improvement in terms of the error.

publication, the authors offer several solutions based on inserting additional samples. However, further discussion is out of scope of this evaluation. Just note that this problem does not appear, if the domain boundary is evenly sampled, e.g., in the case of regular sampling.

The use of a Delaunay mesh is usually limited to lower dimensions, due to both computational and memory complexity. These limitations, in conjunction with an unstable performance, suggest the use of the RNG and GAB meshes instead of DEL mesh, except the case when it is known that the domain boundary is evenly sampled.

Gabriel graph, relative neighborhood graph meshes. The GAB and RNG meshes show consistently good performance for all functions, thus their use is suggested for the setup and functions similar to ones used in this evaluation. Note that similar recommendation was made in the evaluation study using the F-measure, conducted by Correa and Lindstrom [26].

However, the relaxed versions of both RNG and GAB meshes, suggested as an improvement in the same study, didn't show any significant performance improvement, see Figure 6.10, in the evaluation. In fact, the high number of false extrema was handled as noise by proposed error measures, the key difference between these error measures and the F-measure.

Furthermore, note that in the majority of provided experiments, for lower number of samples,

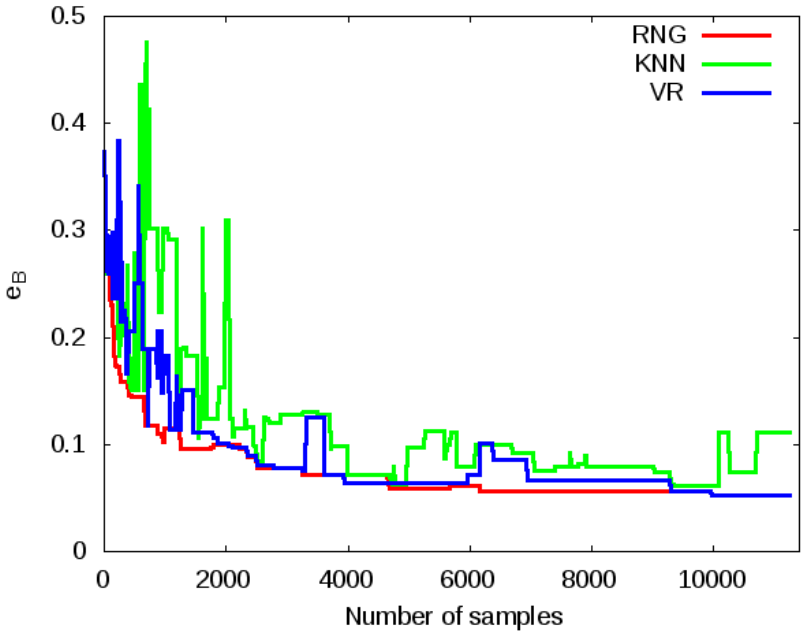

Figure 6.11: The minimally connected KNN and VR meshes compared to the RNG mesh. The minimally connected VR mesh performs better, as it approximates the function better.

the RNG mesh performs slightly better than the GAB mesh.

K-Nearest-Neighbors, Vietoris-Rips meshes. Here the performance of the KNN mesh is evaluated. Since no optimal strategy is known for selecting the parameter k, a minimally connected version of the KNN mesh is evaluated. This version is based on computing the lowest value of the parameter k, for which the mesh is connected. The value of k differs for every set of samples. Comparison of the minimally connected KNN mesh to the RNG mesh is provided in Figure 6.11. The performance of the minimally connected KNN mesh is highly unstable, mainly due to the small number of considered neighbors for each sample.

In comparison, the average number of neighbors for the minimally connected VR mesh is significantly higher. Hence, as expected, a similar evaluation of the minimally connected VR mesh shows more stable behavior and lower error, see Figure 6.11. Indeed, fixing the value of k for the KNN mesh to the average number of neighbors of the minimally connected VR mesh, the performance improves significantly, see Figure 6.12.

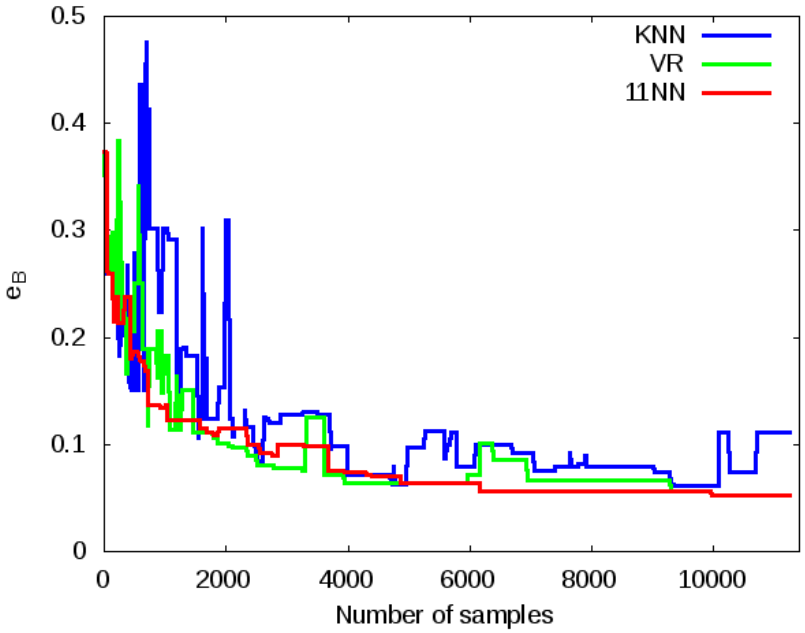

Figure 6.12: The average number of neighbors for the minimally connected VR mesh in Figure 6.11 equals 11. If to set the number of neighbors for the KNN mesh to 11, and compare it to the minimally connected KNN and VR meshes, the performance is significantly more stable.

To evaluate the KNN mesh further, an additional sweep of the parameter k is performed. The resulting error function is given in Figure 6.13 as a surface, where the valley corresponds to the higher value(s) of the parameter k, for which the KNN mesh performs better. While these results do not provide definitive means to calculate the optimal value of the parameter k, choosing moderately high values of the parameter k reduces the error. The latter observation agrees with the similar results of the evaluation using F-measure, mentioned earlier.

6.4.4 Nonlinear Manifolds

So far, the domain of the ground truth function was assumed to be a linear manifold, embedded into Euclidean space of the same dimensionality — a square in 2D space, a cube in 3D space, etc. But this is not always the case. For example, for shape matching [12], it is common to consider the shape as the underlying manifold, to define a scalar function on it (e.g., the geodesic distance to a

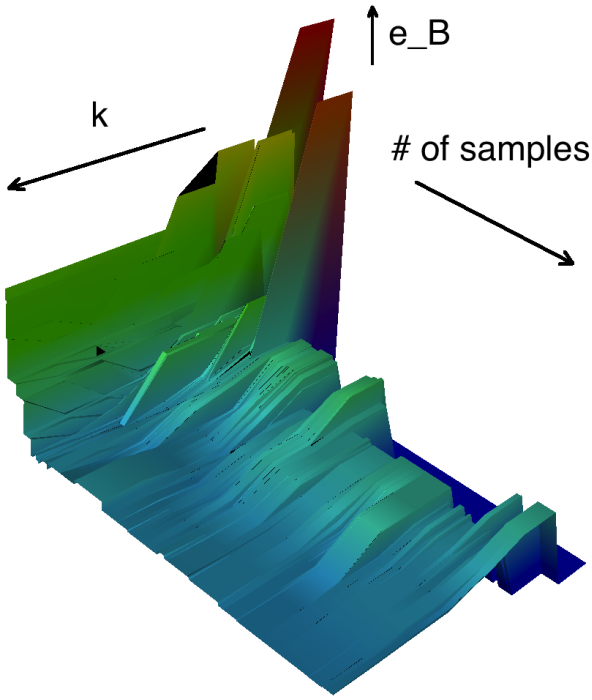

Figure 6.13: The sweep of the parameter k versus the number of samples. The deep blue area on the right side corresponds to the values of k, for which mesh is disconnected, thus no results are available. Almost in all cases, the error decreases and then increases, with growing k value.

point), and then to use topological properties of that function to match shapes. Another context is computing approximated topological structures. Takahashi et al. [76] devised a method that, given data samples in high-dimensional space, find a low-dimensional manifold such that the projections of sample points to the manifold reflect their proximity in the original space. By reducing the dimensionality, their method improves the runtime efficiency at the cost of decreasing precision. Although neither context is in the scope of this work, they raise the question of how different types of meshes behave when ground truth functions are defined on embedded manifolds.

To answer this question, first consider linear manifolds, embedded in Euclidean space with higher dimension. Since the distances between samples do not change in this case, the evaluation results also stay the same. A more interesting case arises when manifolds are nonlinear, i.e., curved. To investigate this case, additional experiments are proposed, where all the previous parameters are

used, except the underlying manifold, which is curved and embedded into Euclidean space of one dimension higher, see Figure 6.14.

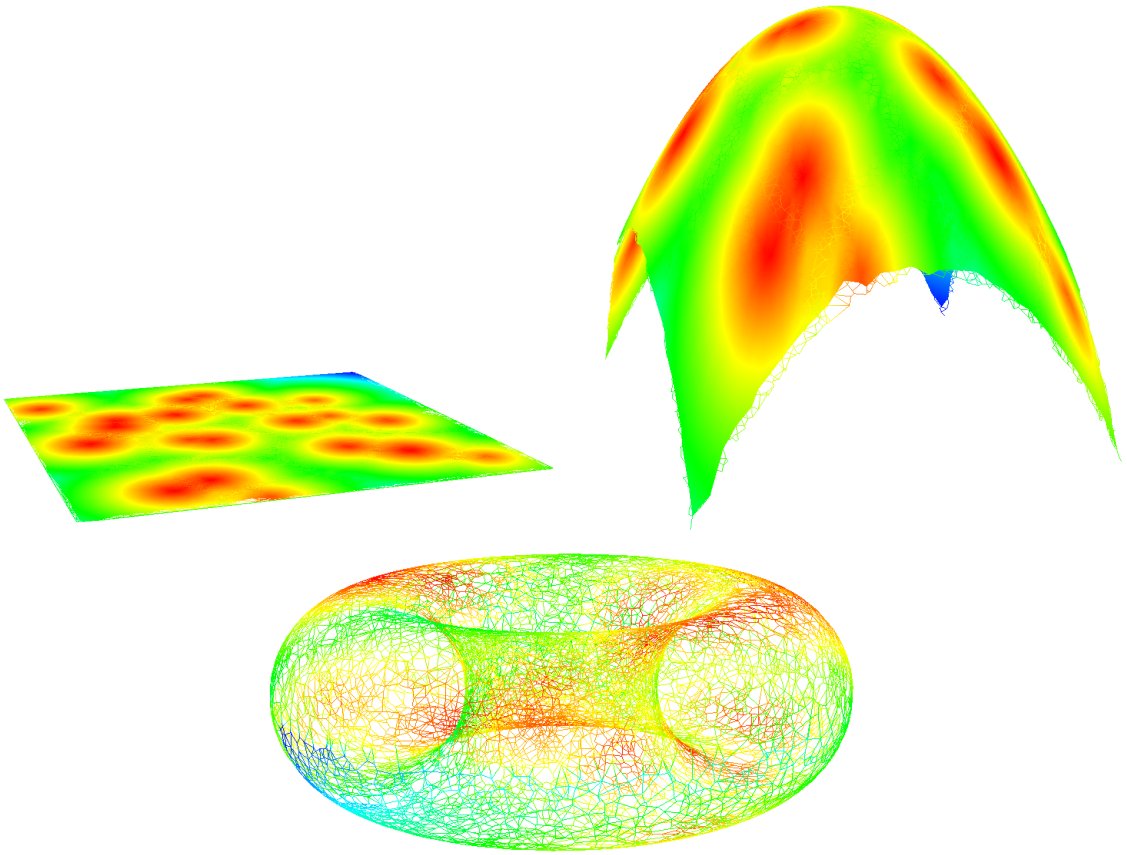

Figure 6.14: The evaluation with the underlying manifold changed from a square $(x, y, z = 0)$ in 2D Euclidean space (top left) to the paraboloid $(x, y, z = -x^2 - y^2)$ embedded in 3D space (top right); and to the torus $(x, y, z = \pm\sqrt{r^2 - (R - \sqrt{x^2 + y^2})^2})$ embedded in 3D space (bottom). In all cases, the following parameter values $\{x = y = [0, 150], z = [0, 75], r = 25, R = 50\}$ are used. The ground truth function, shown in colors (red indicating high function values, blue indicating low function values), stays the same in all cases.

The experimental results suggest that in the case of tested curved manifolds, all mesh types perform similar to the case of linear manifolds, see Figures 6.15, 6.16, with the exception of the worse performance in the case of the DEL mesh. Such behavior is expected since the Delaunay triangulation has higher dimensionality than sampled manifolds, connecting its distant points; in contrast, the other meshes approximate the embedded manifolds well. In addition, obtained results

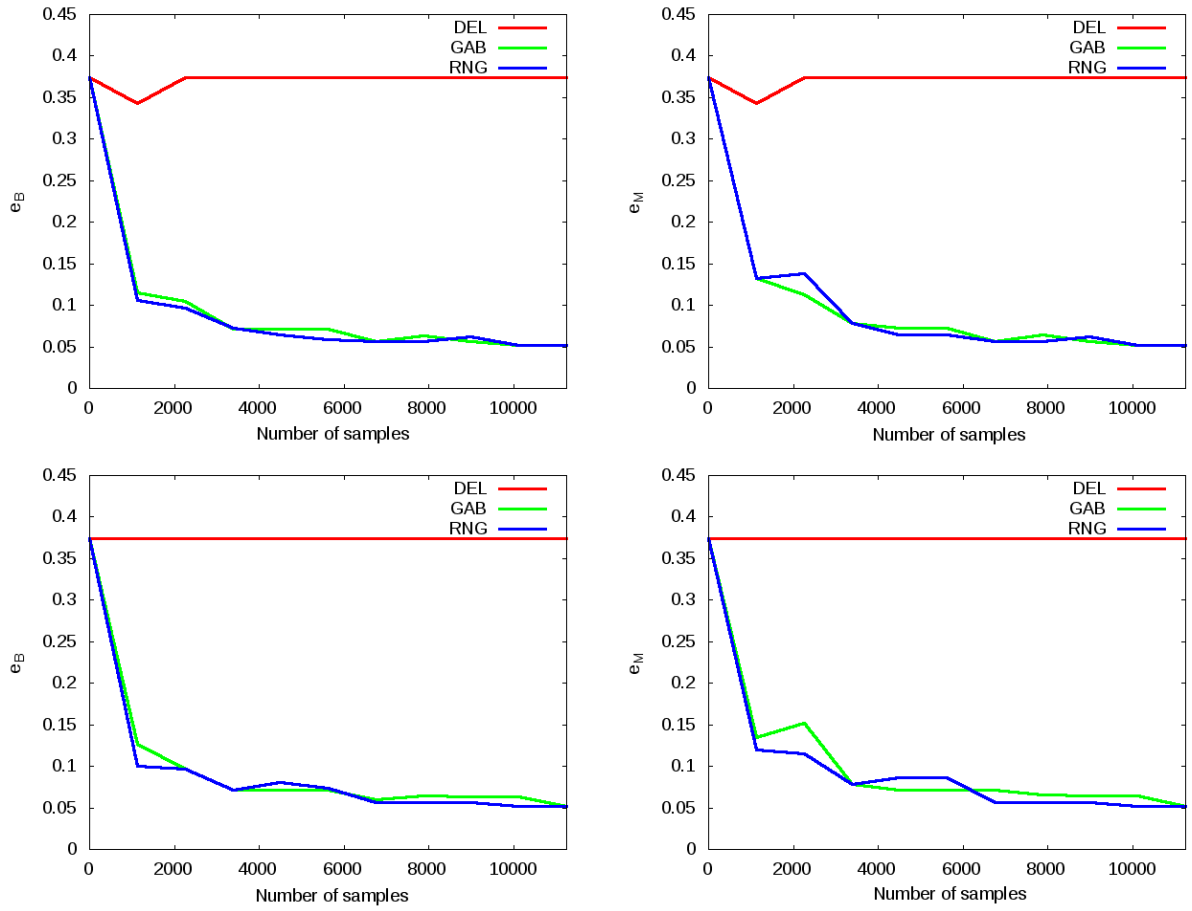

Figure 6.15: Performance of the DEL, GAB, and RNG meshes on curved square manifolds, namely on the paraboloid (top two), and on the torus (bottom two) manifolds. The ground truth function is the same as the one used in Figure 6.6.

are slightly better for the paraboloid than for the torus, which, again, is expected due to the better approximation of the embedded manifold.

In conclusion, note that the considered curved manifolds are relatively simple, and a separate study with more comprehensive analysis is required.

6.4.5 Evaluation Summary

To summarize the results of the evaluation: (1) the GAB and RNG meshes show stable good performance, with the RNG mesh being slightly better in cases with lower number of samples; (2)

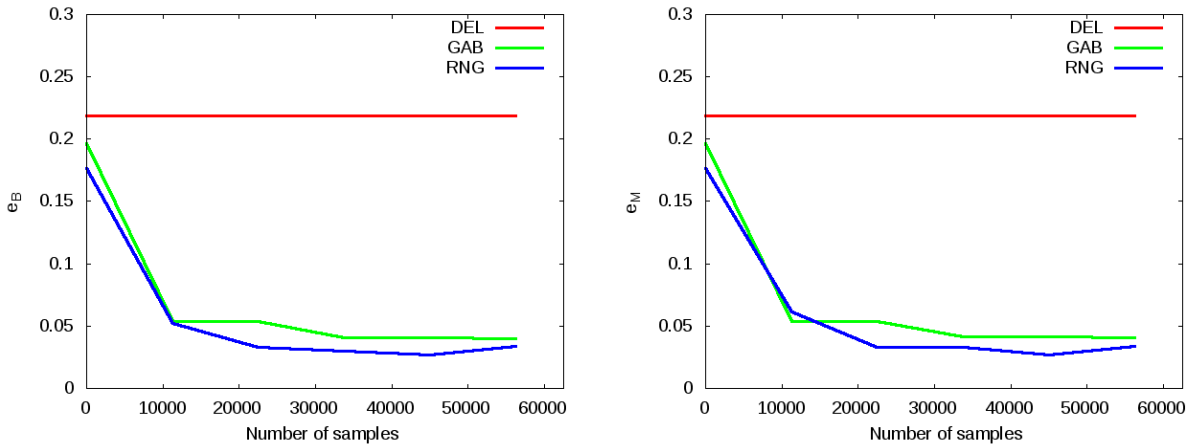

Figure 6.16: Performance of the DEL, GAB, and RNG meshes on curved (paraboloid) cube manifold.

the relaxed RNG and GAB meshes perform similar to the original versions; (3) the performance of minimally connected KNN mesh is highly unstable, while the minimally connected VR mesh performs significantly better; (4) in case of small number of samples, the minimally connected KNN and VR meshes perform similar to the GAB and RNG meshes, thus their use can be justified for such cases; (5) curved domains do not change the previous recommendations much, except for the DEL mesh, performance of which gets significantly worse.

Note that these results are applicable for functions, similar to ones in the set of ground truth functions, i.e., a function must have relatively slow changing slopes (no spikes) and no complex topology (e.g., loops). Also, the sampling strategy should be random.

6.5 Evaluation of Proposed Topology-based Techniques

Conducted evaluation and its recommendations are only partially applicable to techniques from Chapters 3, 4, since their input functions (data sets) differ from those considered in the previous section. So the evaluation was extended to include data sets that were used to demonstrate proposed techniques. Produced specialized recommendations support the selection of approximation

methods for proposed techniques.

The first technique deals with potential and free energy data sets, both of which are uniformly densely sampled. Given that there are no sudden large value changes in the ground truth function, a function, approximated from any of these dense set of samples, is close to the ground truth, i.e., has little error. Subsequently, such approximated functions were considered as new ground truth functions.

To decide which type of mesh to use for the new ground truth, the performance of different types of meshes on the same dense set of samples was tested (for all available sets of samples), including the mesh, constructed using Freudenthal subdivision (used for the first technique). The results showed that all types of mesh performed within normalized 0.0003 or 0.03% error range between each other, suggesting that any one of these types of meshes can be selected to define the new ground truth. For comparison, the lowest obtained error in the general evaluation for similar analytical 3D function was around 0.03 or 3%, see Figure 6.8. Further on, the new ground truth is by default assumed to be computed using the Freudenthal mesh.

Finally, for data sets with dimensions higher than three, the KNN mesh was tested against the new ground truth. The results again show that with increase of dimensionality, as long as sampling density is high, the KNN mesh error stays within normalized 0.0005 or 0.05% of the new ground truth . This suggests that the use of the KNN mesh, chosen due to its faster computation in higher dimensions and less number of edges, is acceptable.

A similar discussion applies to the performance data set, used by the second technique. The error range for different types of mesh on a dense set of samples in this case is 0.2%, and the KNN mesh performs within 0.5% of the new ground truth.

These recommendations, produced by the extended evaluation, support the selection of the Freudenthal mesh for lower dimensions and the KNN for higher dimensions for both techniques.

Chapter 7

Conclusions

This doctoral dissertation presented several research contributions to the advancement of the field of topology-based data analysis and visualization, including two novel analysis techniques with specific goals and capabilities and a new approach to error measurement that provides some confidence in the precision of approximation techniques.

First, an application-inspired problem of analyzing transformation pathways led to the development of a novel analysis technique. This technique combines the Morse complex, dimension reduction, noise and periodicity handling, and graph representation methods to provide a new transformation pathway analysis tool; this tool is an improvement over existing techniques that are either less informative or low-dimensional. Given various limitations of Morse complex computation, noise handling, and graph projection, this technique is best suited to extracting transformation pathways for data sets of moderate dimensionality (up to 13). The visual representation developed in this technique constitutes an addition to the computational chemist's visualization toolbox and provides new capabilities for visualizing complex multidimensional energy functions. The technique emphasizes the most relevant information for chemists: the number and location of energy minima and the heights of connecting barriers. The resulting map allows the investigation of all possible transformation (e.g., reaction) pathways, the identification of the lowest energy paths, etc.

Although the matter has not been discussed in this study, given representations are readily extended to include special transformation paths that utilize tunneling mechanisms rather than ordinary transitions through barriers.

This technique can be further improved by the inclusion of additional information into the plots, as well as by combination with other types of chemical analysis. Specifically, one improvement is achieved through the addition of information on path directions. The current version of the representation does not indicate the relative positions of minima except by directly labeling them with the corresponding coordinates of the minima. One might consider coloring pathways corresponding to transformations along a particular coordinate axis. In the case of periodic systems such as the zeolite example presented here, A variation of this approach could indicate lowest energy pathways along each of principal directions by marking them with distinct colors. Another improvement regards the handling of visual complexity. For more complex cases that involve large high-dimensional energy grids, the resulting graphs are very complex, making visual analysis difficult. Two possible approaches to simplification are: (i) displaying only a selected fragment of a graph. This capability will require additional implementation for redrawing graphs that contain only selected subsets of the entire graph; (ii) simplification of graphs, for example by partial clustering of nodes. This approach may be based on implementation of some chemical information into the graph. For example, when considering energy landscapes inside a porous material, one often finds many close minima that correspond to a location inside a given cage in a porous material. Such minima could be clustered into a single minimum point corresponding to a particular cage, and the resulting graph would present only cages. An additional improvement might be the development of an interactive interface. Since the Morse complex is hierarchical relative to persistence and upper thresholds, its graph representation can be interactively visualized with little overhead. Other features, such as the selection and redrawing of a subgraph might also be useful.

Second, the problem of enabling comparative analysis has been investigated, leading to the development of a novel comparative technique based on geometry-preserving topological landscapes.

In particular, three stages of building such landscapes are described: the contour tree computation, the contour tree layout, and landscape construction. The necessary details of the implementation are provided, and an example application of the technique to real-world data is given.

This technique is limited to moderate size data sets, mainly as a consequence of the potential explosion of the number of points/edges during the iterative edge routing process. This issue can be addressed by improving the edge routing algorithm based on segment Voronoi diagram. Another issue that might be addressed is the calculation of the gap distance during the offsetting. Currently, the algorithm roughly calculates this value as the shortest distance between any two points, divided by number of saddles (i.e., the number of potential contours). A more precise approximation can be achieved, since routing via a Voronoi diagram maintains the half-distance to other elements.

Third was the exploration of the problem of quantifying the error in approximated topology, a common problem in scientific data analysis. In particular, a definition of a distance between merge trees is given, which is then employed to quantify the approximation error. This error measure is used to evaluate common approximation methods, resulting in recommendations, including those that support the selection of methods in the two techniques previously presented. The recommendations, so produced, are applicable to functions similar to the set of functions used in the evaluation study; and they are also valid for chemistry and the performance data sets specifically added into the evaluation. The notable conclusions drawn from the evaluated set of functions are that analytical functions are slow changing and have no complex topology (e.g., holes in manifolds); and chemical and performance data sets are well-sampled. The evaluation of meshes for various other functions is left for future research.

Potential follow-up work on measuring distances between merge trees includes a theoretical investigation of the proposed distance. For example, its relation to the bottleneck distance between persistence diagrams, as well as proving that it is a stable metric under possible small perturbations in merge trees. Another direction to explore is measuring the distance between other topological structures, such as the Reeb graph and Morse-Smale complex.

References

[1] M. S. Apaydin, D. L. Brutlag, C. Guestrin, D. Hsu, and J.-C. Latombe. Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion. In *Proceedings RECOMB'02*, pages 12–21, 2002.

[2] M. Atallah and M. Blanton. *Algorithms and Theory of Computation*. Chapman and Hall/CRC, 2009.

[3] S. Auerbach, K. Carrado, and P. K. Dutta. *Handbook of Zeolite Science and Technology*. Marcel Dekker, 2004.

[4] R. Bachorz, M. Haranczyk, I. Dabkowska, J. Rak, and M. Gutowski. Anion of the formic acid dimer as a model for intermolecular proton transfer induced by a π^* excess electron. *Journal of Chemical Physics*, 122(20):204–304, May 2005.

[5] C. L. Bajaj, V. Pascucci, and D. R. Schikore. Visualization of scalar topology for structural enhancement. In D. Ebert, H. Hagen, and H. Rushmeier, editors, *Proc. IEEE Visualization*, pages 51–58. IEEE, IEEE Computer Society Press, 1998.

[6] G. D. Batista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1998.

[7] U. Bauer, X. Ge, and Y. Wang. Measuring Distance between Reeb Graphs. *ArXiv e-prints*, July 2013.

[8] K. Beketayev, D. Morozov, G. H. Weber, A. Abzhanov, and B. Hamann. Geometry–preserving topological landscapes. In *Workshop at SIGGRAPH Asia*, pages 155–160, 2012.

[9] K. Beketayev, G. Weber, M. Haranczyk, P.-T. Bremer, M. Hlawitschka, and B. Hamann. Topology-based visualization of transformation pathways in complex chemical data. *Computer Graphics Forum*, 30(3):663–672, June 2011.

[10] K. Beketayev, D. Yeliussizov, D. Morozov, G. H. Weber, and B. Hamann. Measuring the distance between merge trees. In *Topological Methods in Data Analysis and Visualization III*, Mathematics and Visualization. Springer-Verlag, 2013. To appear.

[11] E. W. Bethel and M. Howison. Multi-core and many-core shared-memory parallel raycasting volume rendering optimization and tuning. *International Journal of High Performance Computing Applications*, 2012. http://dx.doi.org/10.1177/1094342012440466.

[12] S. Biasotti, L. De Floriani, B. Falcidieno, P. Frosini, D. Giorgi, C. Landi, L. Papaleo, and M. Spagnuolo. Describing shapes by geometrical-topological properties of real functions. *ACM Computing Surveys*, 40:12:1–12:87, Oct. 2008.

[13] S. Biasotti, M. Marini, M. Spagnuolo, and B. Falcidieno. Sub-part correspondence by structural descriptors of 3D shapes. *Computer Aided Design*, 38(9):1002–1019, Sept. 2006.

[14] P. Bille. A survey on tree edit distance and related problems. *Journal of Theoretical Computer Science*, 337:217–239, June 2005.

[15] P.-T. Bremer, G. H. Weber, J. Tierny, V. Pascucci, M. S. Day, and J. B. Bell. A topological framework for the interactive exploration of large scale turbulent combustion. In *Proc. 5th IEEE International Conference on e-Science*, pages 247–254. IEEE, 2009.

[16] H. Bunke and K. Riesen. *Graph Edit Distance: Optimal and Suboptimal Algorithms with Applications*, pages 113–143. Wiley-VCH Verlag GmbH & Co. KGaA, 2009.

[17] D. Bursztyn and D. M. Steinberg. Comparison of design for computer experiments. *Journal of Statistical Planning and Inference*, 136:1103–1119, 2006.

[18] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[19] J. Cardinal, S. Collette, and S. Langerman. Empty region graphs. *Comp. Geom. Theory Appl.*, 42:183–195, Apr. 2009.

[20] H. Carr and J. Snoeyink. Contour tree simplification with local geometric measures. In *14th Annual Fall Workshop on Computational Geometry*, page 51, 2004.

[21] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry – Theory and Applications*, 24(2):75–94, Feb. 2003.

[22] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *IEEE Visualization 2004*, pages 497–504. IEEE, Oct. 2004.

[23] Y.-J. Chiang, T. Lenz, X. Lu, and G. Rote. Simple and output-sensitive construction of contour trees using monotone paths. Technical report, Freie Universtat Berlin, 2002.

[24] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. In *Proceedings of 21st Annual Symposium on Computational Geometry*, pages 263–271. ACM, 2005.

[25] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in reeb graphs of 2-manifolds. *Discrete & Computational Geometry*, 32:231–244, 2004.

[26] C. D. Correa and P. Lindstrom. Towards robust topology of sparsely sampled data. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1852–1861, Dec. 2011.

[27] C. D. Correa, P. Lindstrom, and P.-T. Bremer. Topological spines: A structure-preserving visual representation of scalar fields. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1842–1851, Dec. 2011.

[28] D. Demir, K. Beketayev, G. H. Weber, P.-T. Bremer, V. Pascucci, and B. Hamann. Topology exploration with hierarchical landscapes. In *Workshop at SIGRRAPH Asia*, pages 147–154. ACM Press, 2012.

[29] T. Dwyer and L. Nachmanson. Fast edge-routing for large graphs. In *Proc. of 17th International Conference on Graph Drawing*, pages 147–158, 2009.

[30] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-Smale complexes for piecewise linear 3-manifolds. In *Proceedings of the 19th ACM Symposium on Computational Geometry*, pages 361–370, 2003.

[31] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifold. *Discrete & Computational Geometry*, 30:87–107, 2003.

[32] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete Computational Geometry*, 28:511–533, 2002.

[33] H. Edelsbrunner, D. Morozov, and V. Pascucci. Persistence–sensitive simplification functions on 2-manifolds. In *Proceedings of 22nd Annual Symposium on Computational Geometry*, pages 127–134. ACM, 2006.

[34] D. Engel, R. Rosenbaum, B. Hamann, and H. Hagen. Structural decomposition trees: semantic and practical implications. *Computer Graphics Forum*, 30(3):921–930, 2011.

[35] C. Flamm, I. L. Hofacker, P. F. Stadler, and M. T. Wolfinger. Barrier trees of degenerate landscapes. *J. Physical Chemistry*, 216:155–173, 2002.

[36] I. Fodor. A survey of dimension reduction techniques. Technical report, LLNL, 2002.

[37] R. Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134(1):90–145, 1998.

[38] H. Freudenthal. Simplizialzerlegung von beschränkter Flachheit. *Ann. Math.*, 43:580–582, 1942.

[39] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, J. A. Montgomery, Jr., T. Vreven, K. N. Kudin, J. C. Burant, J. M. Millam, S. S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G. A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J. E. Knox, H. P. Hratchian, J. B. Cross,

V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, P. Y. Ayala, K. Morokuma, G. A. Voth, P. Salvador, J. J. Dannenberg, et al. Gaussian 03, Revision C.02. Gaussian, Inc., Wallingford, CT, 2004.

[40] E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Software—Practice and Experience*, 30:1203–1233, 1999.

[41] E. García-Pérez, J. B. Parra, C. O. Ania, A. García-Sánchez, J. M. van Baten, R. Krishna, D. Dubbeldam, and S. Calero. A computational study of co2, n2, and ch4 adsorption in zeolites. *Adsorption*, 13(5-6):469–476, Sept. 2007.

[42] C. Garth, X. Tricoche, and G. Scheuermann. Tracking of vector field singularities in unstructured 3d time-dependent data sets. In *Proceedings of IEEE Visualization 2004*, pages 329–336, 2004.

[43] S. Gerber, P.-T. Bremer, V. Pascucci, and R. Whitaker. Visual exploration of high dimensional scalar functions. *IEEE Transcations on Visualization and Computer Graphics*, 16(6), Nov. 2010.

[44] A. Gyulassy, P.-T. Bremer, V. Pascucci, and B. Hamann. Efficient computation of Morse-Smale complexes for three-dimensional scalar functions. In *Proceedings of IEEE Visualization*, pages 1440–1447, 2007.

[45] A. Gyulassy, P.-T. Bremer, V. Pascucci, and B. Hamann. A practical approach to Morse Smale complex computation: Scalability and generality. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1619–1626, Nov. 2008.

[46] A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann. Topology-based simplification for feature extraction from 3D scalar fields. In *Proceedings of the IEEE Visualization 2005 (VIS'05)*, pages 535–542. IEEE Computer Society, 2005.

[47] W. Harvey and Y. Wang. Topological landscape ensembles for visualization of scalar-valued functions. *Computer Graphics Forum*, 29(3):993–1002, May 2010.

[48] W. Harvey, Y. Wang, and R. Wenger. A randomized O (m log m) time algorithm for computing Reeb graphs of arbitrary simplicial complexes. In *Proceedings of the 2010 annual symposium on Computational geometry*, pages 267–276. ACM, 2010.

[49] C. Heine, G. Scheuermann, C. Flamm, I. L. Hofacker, and P. F. Stadler. Visualization of barrier tree sequences. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):781–788, Sept./Oct. 2006.

[50] C. Heine, D. Schneider, H. Carr, and G. Scheuermann. Drawing contour trees in the plane. *IEEE Trans. Vis. Comput. Graph.*, 17(11):1599–1611, 2011.

[51] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of ACM SIGGRAPH*, 2001.

[52] T. James, D. J. Wales, and J. H. Rojas. Energy landscapes for water clusters in a uniform electric field. *The Journal of Chemical Physics*, 126:054506, 2007.

[53] T. Jiang, E. Lawler, and L. Wang. Aligning sequences via an evolutionary tree: complexity and approximation. In *Symposium on Theory of Computing*, pages 760–769, 1994.

[54] D. Keffer, V. Gupta, D. Kim, E. Lenz, H. T. Davis, and A. V. McCormick. A compendium of potential energy maps of zeolites and molecular sieves. *Journal of Molecular Graphics and Modelling*, 14:108–116, 1996.

[55] R. Krishna and J. van Baten. Using molecular simulations for screening of zeolites for separation of co2/ch4 mixtures. *Chemical Engineering Journal*, 133:121–131, Sept. 2007.

[56] D. Lasser and T. Stüttgen. Boundary improvement of piecewise linear interpolants defined over Delaunay triangulations. *Journal of Computers & Mathematics with Applications*, 32(10):43–58, Nov. 1996.

[57] S. Lohr. *Sampling: Design and Analysis*. Brooks/Cole, Cengage Learning, 2010.

[58] S. Mizuta, T. Ono, and T. Matsuda. Contour nest: A two-dimensional representation for three-dimensional isosurfaces. In *Proc. Volume Graphics*, pages 67–70, 2006.

[59] S. Mizuta, Y. Suwa, T. Ono, and T. Matsuda. Description of the topological structure of digital images by contour tree and its application. Technical report, Institute of Electronics, Information and Communication Engineers, 2004.

[60] J. R. Munkres. *Elements of Algebraic Topology*. Addison-Wesley, Redwood City, California, 1984.

[61] P. Oesterling, C. Heine, H. Jänicke, and G. Scheuermann. Visual analysis of high dimensional point clouds using topological landscapes. In *Proc. IEEE Pacific Visualization 2010 Symposium*, pages 113–120, 2010.

[62] P. Oesterling, C. Heine, H. Jänicke, G. Scheuermann, and G. Heyer. Visualization of high-dimensional point clouds using their density distribution's topology. *IEEE Transactions on Visualization and Computer Graphics*, 17:1547–1559, 2011.

[63] P. Oesterling, G. Scheuermann, S. Teresniak, G. Heyer, S. Koch, T. Ertl, and G. Weber. Two-stage framework for a topology-based projection and visualization of classified document collections. In *Proc. IEEE Symposium on Visual Analytics Science and Technology*, 2010.

[64] T. Okushima, T. Niiyama, K. S. Ikeda, and Y. Shimizu. Graph-based analysis of kinetics on multidimensional potential-energy surface. *Physical Review E*, 80(3):036112, 2009.

[65] J. Pach and R. Wenger. Embedding planar graphs at fixed vertex locations. *Graphs and Combinatorics*, 17:717–728, 2001.

[66] S. Parsa. A deterministic O(m log m) time algorithm for the Reeb graph. In *Proceedings of the twenty-eighth annual symposium on Computational geometry*, SoCG '12, pages 269–276, New York, NY, USA, 2012. ACM.

[67] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. Multi-resolution computation and presentation of contour trees. Technical report, LLNL, 2005.

[68] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. The toporrery: Computation and presentation of multi-resolution topology. In *Math. Found. of Sci. Vis., Comput. Graph., and Massive Data Explor.*, pages 19–40. Springer-Verlag, 2009.

[69] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. Robust on-line computation of Reeb graphs: Simplicity and speed. *ACM Transactions on Graphics*, 26(3):58.1–58.9, Aug. 2007.

[70] G. Reeb. Sur les points singuliers d'une forme de Pfaff complément integrable ou d'une fonction numerique. *Comptes Rendus Acad. Science Paris*, 222:847–849, 1946.

[71] G. Rote. Piecewise linear Morse theory.

[72] B. Smit and T. L. M. Maesen. Molecular simulations of zeolites: Adsorption, diffusion, and shape selectivity. *Chemical Reviews*, 108:4125–4184, Sept. 2008.

[73] B. Smit and T. L. M. Maesen. Towards a molecular understanding of shape selectivity. *Nature*, 451:671–678, Feb. 2008.

[74] J. J. P. J. Stewart. Optimization of parameters for semiempirical methods i. method. *Journal of Computational Chemistry*, 10:209–220, Mar. 1989.

[75] J. J. P. J. Stewart. Optimization of parameters for semiempirical methods ii. applications. *Journal of Computational Chemistry*, 10:221–264, Mar. 1989.

[76] S. Takahashi, L. Fujishiro, and M. Okada. Applying manifold learning to plotting approximate contour trees. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1185–1192, June 2009.

[77] R. Tamassia. *Handbook of Graph Drawing and Visualization*. Chapman and Hall/CRC, Aug. 2012.

[78] D. M. Thomas and V. Natarajan. Symmetry in scalar field topology. *Transactions on Visualization and Computer Graphics*, 17(12):2035–2044, Dec. 2011.

[79] W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17:401–419, 1952.

[80] M. J. van Kreveld, R. van Oostrum, C. L. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Symposium on Computational Geometry*, pages 212–220, 1997.

[81] L. Vietoris. Über den höheren zusammenhang kompakter räume und eine klasse von zusammenhangstreuen abbildungen. *Mathematische Annalen*, 97:454–472, 1927.

[82] G. H. Weber. *Visualization of Adaptive Mesh Refinement Data and Topology-based Exploration of Volume Data*. PhD thesis, University of Kaiserslautern, 2003.

[83] G. H. Weber, P.-T. Bremer, and V. Pascucci. Topological landscapes: A terrain metaphor for scientific data. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1416–1423, Nov. 2007.

[84] G. H. Weber, P.-T. Bremer, and V. Pascucci. *Topological Cacti: Visualizing Contour-based Statistics*, pages 63–76. Springer-Verlag, Heidelberg, Germany, 2011.

[85] G. H. Weber, S. E. Dillard, H. Carr, V. Pascucci, and B. Hamann. Topology-controlled volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):330–341, Mar. 2007.