Extremum Graph Layout and Visualization

A PROJECT REPORT
SUBMITTED IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
Master of Technology
IN
Faculty of Engineering

BY

Nandini Gour

Computer Science and Automation
Indian Institute of Science
Bangalore – 560 012 (INDIA)

June, 2022

Declaration of Originality

I, **Nandini Gour**, with SR No. **17929** hereby declare that the material presented in the thesis titled

Extremum Graph Layout and Visualization

represents original work carried out by me in the **Department of Computer Science and Automation** at **Indian Institute of Science** during the years **2020-2022**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: 29 June 2022 Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Vijay Natarajan Advisor Signature

DEDICATED TO

My parents

Acknowledgements

I would like to acknowledge and thank my advisor, Prof. Vijay Natarajan, of the Visualization and Graphics Lab (VGL), under the Department of Computer Science and Automation, IISc Bangalore. His guidance and advice throughout the journey made this project possible. His regular discussion about my project progress and questioning every detail of my work helped to complete this wonderful project.

I would like to appreciate the support of my lab-mate Abhijath Ande for helping me to start the project. I would also like to give my gratitude to Raghavendra G S. His supportive nature throughout the journey, and his guidance, whenever I got stuck in my project, helped me to keep pace and complete the project.

Finally, I would like to thank my mom, dad, and all my beloved friends for being with me when I needed them and for encouraging me to complete this work.

Abstract

There are many tools in the field of topological analysis which provide better and more robust data analysis. They are used for many tasks such as feature extraction, comparison of datasets, etc. An extremum graph is also one of the tools used for topological analysis. Extremum graph allows us to analyze the high dimensional scalar fields while preserving the local geometric structure. The topology of scalar fields provides many essential properties, such as the number of connected components of an iso-surface or their critical points. So here, we use topological spines to visualize the extremum graph and also the time-varying extremum graph. The topological spines are also used for topological analysis. It is a visual representation that preserves the topology and locality of the extrema and hierarchy of contours.

Contents

List of Figures

Chapter 1

Introduction

This chapter describe the goal and motivation for the project.

1.1 Motivation

The topological analysis became an important part of scientific visualization because the topology of scalar fields provides many important properties like the number of connected components of iso-surfaces. Many topological structures are used to visualize some complex datasets some of them are contour trees- It is a topological structure used to analyze and visualize the data having real-valued measurement. They give information about how level sets merge and split; Reeb graph- for a better understanding of data we can use the Reeb graph because they help us to track the topological changes in the level sets; Morse-Smale Complex and extremum graph are also useful tool for feature tracking and extraction.

Extremum graphs also gain much more importance in the field of topological analysis. This tool is used for feature extraction where most of the features are related to the extrema. It is also a widely used tool for topological analysis because it provides a better knowledge of data using scalar fields while preserving extrema connectivity. It can also use for feature extraction of the time-varying dataset where the extremum graph is computed for each time step where it provides various insights within the dataset over time. Work done by Correa et. al.[1] provide a very first algorithm to compute extremum graph.

The topological spines have also become a widely used tool for topological analysis. It is the extended version of the extremum graph. An extremum graph can be augmented with some geometric information like contour nesting information and volume under the descending

manifold, this augmented graph is called the topological spines. These are visual representations that preserve both topological and geometric features of scalar fields. The topological spine uses the canonical visual representation to link the chain of extrema together; by doing this, it preserves the topology, locality of extrema, and nesting structure of contours.Work done by Correa et. al.[1] also provide a very first algorithm to compute topological spine using extremum graph.

1.1.1 Application of topological spines

- The topological spine can be used for visualizing high-dimensional data and complex datasets. Work by Correa et. al. [1] describes how to use the Morse-Smale complex to compute extremum graphs which are later used for computing topological spines and this structure better represents complex structures such as curves and cycles occurring in topology. They also preserve the topological and geometric structure of scalar fields.

1.2 Project Goal

This project aims to provide a layout for the extremum graph and visualize the extremum graph also the time-varying extremum graph with the help of the topological spine. The goal of the project is to implement the layout of the extremum graph while the algorithm is already given and also develop the algorithm to visualize time-varying data with the help of topological spines. We first implement the already given algorithm to compute the topological spine for this we take a 3-dimensional extremum graph and then convert it into a 2-dimensional graph. For this, we use a force-directed layout to calculate the location of vertices in a 2-D plane. To ensure that distance is somewhat preserved between two critical points, we use the euclidean distance between them while using the spring layout. Then we draw the topological spines using the link radius calculation of every maxima saddle maxima pair. Now we extend the topological spine computation further for time-varying data.

For the time-varying data, the steps for computing the topological spine for 1st-time step are the same as described above but for the next time steps, we use previously found positions of the vertex for finding the location of topological spines of this time step.

To ensure that distance is somewhat preserved between two critical points, we use the reciprocal of euclidean distance between them as spring constant while using the spring layout. Then we draw the topological spines using link radius calculation of every maxima saddle maxima pair.

Chapter 2

Defination

This chapter describes some important terms which we use later in our project.

2.1 Force-directed Graph

This graph is based on a 'force-directed graph drawing algorithm'.In this algorithm, nodes act as a magnet that repels each other. An edge between nodes has some spring force that attracts or repels the nodes. This algorithm runs iteratively and determine the force on edges until the position of nodes becomes stable.

2.2 spring_layout

This algorithm simulates a force-directed graph drawing algorithm. It treats nodes as a magnet and the edge between nodes acts as a spring that has some repelling force. The algorithm continues until the position of nodes is in equilibrium. Spring_layout is the function present in the NetworkX python package.
It has following parameters:

- **G**: Contain list of nodes or graph.
- **k**: Optimal distance between nodes.
- **pos**: Initial position for nodes.
- **fixed**: List of nodes which have to keep fixed at initial position.
- **iteration**: Maximum number of iterations taken by algorithm.

- **threshold**: threshold for relative error in node position changes.
- **weight**: It is an edge attribute which has numerical value termed as edge weight. Larger edge weight means stronger attractive force.
- **scale**: Scale factor for positions.
- **dim**: dimension of layout.

This function return a dictionary of nodes where the keys are the nodes.

2.3 Principal Component Analysis

PCA is an unsupervised machine learning algorithm. It is a dimensionality reduction technique that is used to reduce the dimensionality of large datasets. It reduces the large dataset into a small dataset in such a way that the new dataset still contains most of the information from the large dataset.

Chapter 3

Background

This chapter covers some of the past work related to extremum graph and topological spines.

3.1 Extremum Graph

The concept of extremum graph was first introduced by Correa et al. [1]. They use extremum graphs for visualizing high-dimensional data, and since they also preserve the topology and geometric properties of original data. They also introduce many algorithms used to simplify the extremum graph. These extremum graphs are also augmented with geometric information like volume under the descending manifold of its adjacent extrema and contour nesting information. When these extremum graphs augmented then they termed as topological spines.

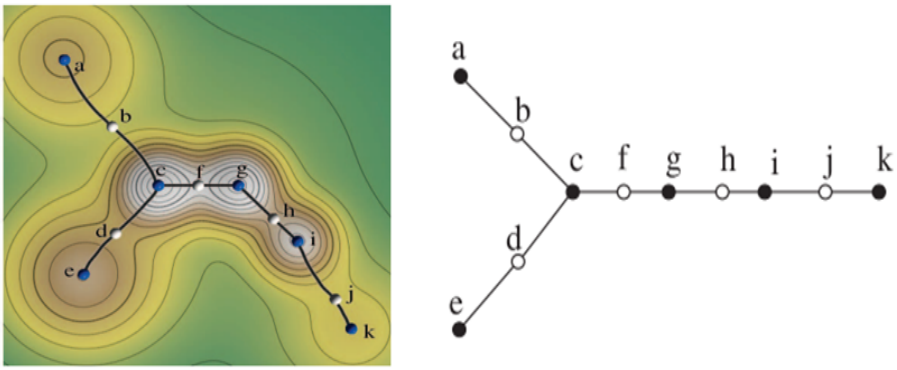

Figure 3.1: Extremum graph (figure taken from [1])

In figure 3.1 left side show an extremum graph of simple 2D terrain connects critical points along steepest ascending(or descending) lines, which join adjacent extrema and therefore better preserve locality. Right side show an abstract representation of the extremum graph.

3.2 Morse-Smale Complex

The morse-Smale complex has also been used for topological analysis. It uses a topological structure to visualize complex datasets. It decomposes the function in terms of the region of uniform gradient flow. Work by Correa et. al. [1] defines an algorithm for computing Morse-Smale complex, their algorithm computes gradient flow graph to identify critical points and gradient paths. As we know many useful structures in a scalar field are associated with critical points(maxima and minima) because of this displaying only the part of the Morse-Smale complex which contains maxima connectivity provides a better visual understanding without losing information.

3.3 Topological spines

The notion of topological spines was also first introduced by Correa et al.[1]. Topological spines use canonical visual representation to link the chain of critical points while preserving the topology, locality of maxima, and nesting structure of surrounding contours. Topological spines maintain much important geometric information. It is used where many existing features are associated with extrema and also if we want to know how neighborhood critical points are present in a data. They first compute extremum graph and then later use it for visualize toplogical spines.

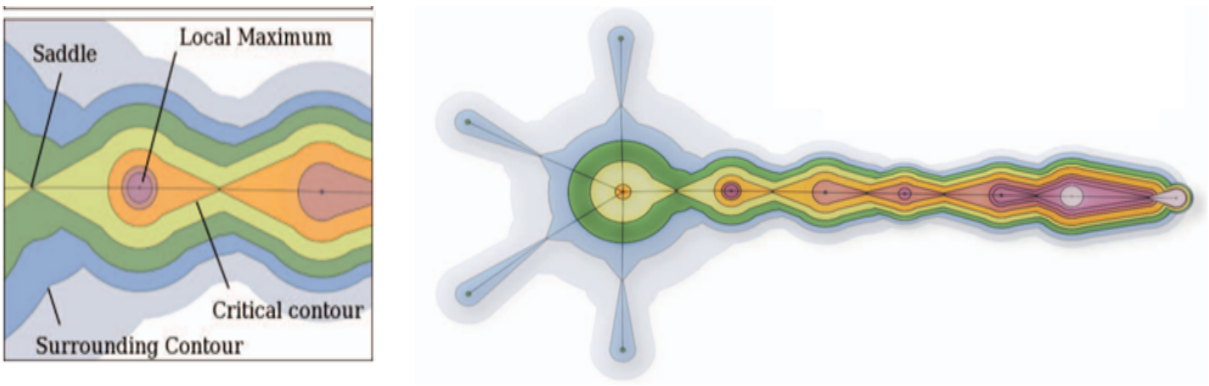

Figure 3.2: Topological spines (figure taken from [1])

In figure 3.2 Right side show topological spine of fuel injection in a combustion chamber. Left side show a zoomed-in view of structural elements of topological spine.

3.4 Feature Tracking

There is much work done in the field of feature tracking and extraction whether it is feature tracking of time-dependent scalar fields or exploring the evolution of features in large-scale

time-varying datasets. Work by Saikia et. al. [2] describes how to track regions in time-dependent scalar fields. In their approach regions are defined using a merge tree and for finding similarities between two consecutive time steps they use their volumetric overlap and histogram differences. They use a directed acyclic graph for recording the similarities information such as nodes representing regions of all time steps, edges representing possible short feature tracks between consecutive time steps and similarity of the connected region is given by edge weight.

Also, the work by Widanagamaachchi et. al. [3] represents an approach to tracking features of time-varying data using dynamic tracking graphs. Their framework provides a natural and interactive exploration of tracking graphs and combined it with a 3D- view of features.

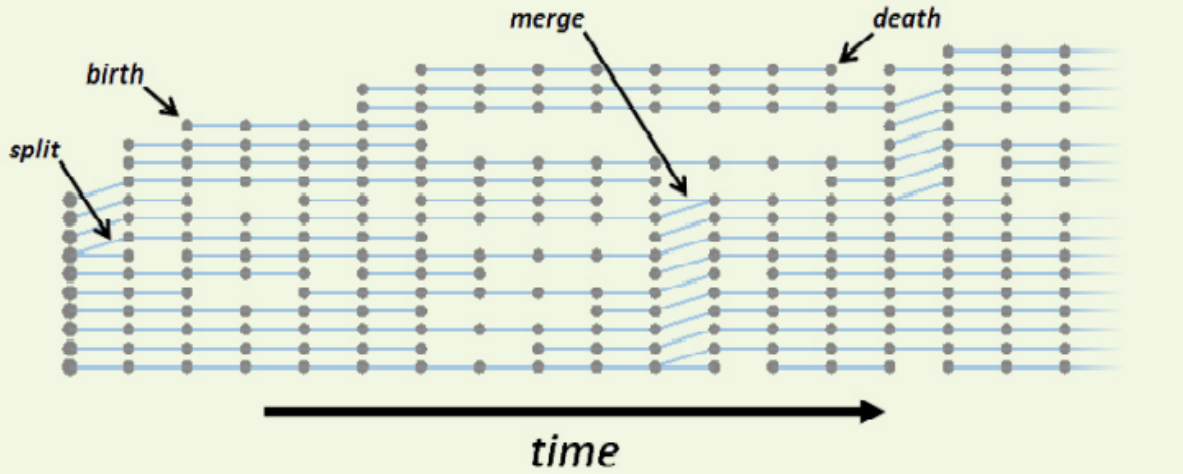

Figure 3.3: Feature tracking (figure taken from [4])

The fig 3.3 is an example of tracking graph showing the evolution of features across time. Within the graph each set of nodes in a vertical column represents features in one time step and shows the tracks of each feature as it evolves: splitting, merging, or disappearing.

3.5 Merge tree

There are many approaches to feature tracking that uses merge tree. Work done by Widanagamaachchi et. al. [4], use the merge concept for feature tracking. In one of their case studies, scientists need to explore both positive and negative pressure perturbation events. They also want to compare feature correlation with respect to metrics based on distance proximity and spatial overlap for this they computed both merge tree and split tree for every time step which allows them to explore both positive and negative pressure perturbation events.

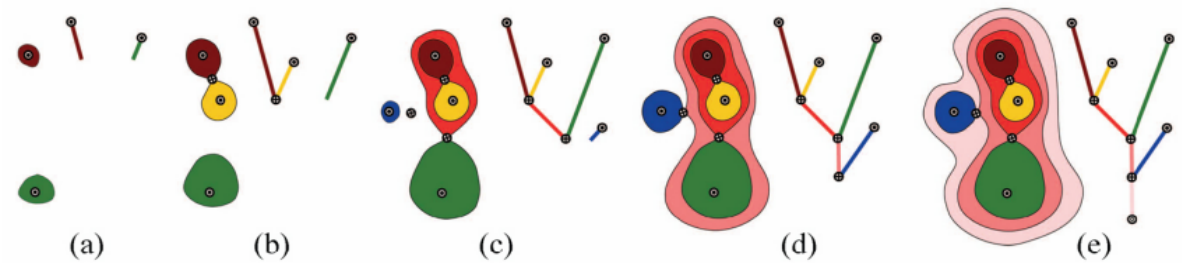

Figure 3.4: Merge tree (figure taken from [3])

Figure 3.4 from (a)-(e) show merge tree construction. A leaf appears in the merge tree each time a new contour appears. That is, each time the issovalue passes through a maximum a new leaf appears in the merge tree and joining of branches in the tree indicates contour merging.

Chapter 4

Problem

In this chapter we describe the algorithm to compute topological spines of time varying scalar fields.

4.1 Problem Statement

To effectively compute the layout and visualize extremum graph and also time-varying extremum graph with the help of topological spines. Here we extend the idea of topological spines to time-varying topological spines. As for each time step topological spines are drawn which uncovers various new insights of data over time.

The key challenge here is to preserve the location of vertices while calculating the topological spine. The location of vertices is somewhat preserved while converting the 3-D graph to a 2-D graph so that the extremum graph doesn't lose its original structure. Another challenge is to prevent edges to come close to each other when new points appear in different time steps in the time-varying dataset.

4.2 Methods Used

4.2.1 Computation of Topological spines

For computing topological spines our very first step is to convert the 3D extremum graph to a 2D extremum graph for this we need a graph representation that gives us a 2D layout and also preserve the original structure so for this, we use the spring_layout function (present in NetworkX package of python). This layout gives us a 2D graph and also provides locations of vertices of topological spines. After finding the positions of vertices we move further to calculating the link radius and then draw circles whose union is a contour across maxima-saddle.

For calculating the topological spine, we use an extremum graph as input.
Extremum graph contains:

- The 3-D coordinates of each vertex.
- The functional value of each point.
- The edges incident on each vertex.
- Also, the Morse index of the critical point, i.e whether the critical point is a saddle or a maximum.

We use a spring layout to calculate the 2-D coordinates of each vertex and the euclidean distance so that distance is preserved between two points.

We also give initial positions of each vertex to the spring layout so that the structure of the extremum graph is preserved. We apply PCA on the 3-D coordinates to calculate the initial position and reduce the dimension to a 2-D.

Then we take an extrema-saddle-extrema pair one by one and calculate the link radius used in topological spine computation.

Link Radius:

- We use functional values to calculate the link radius. Steps to calculate link radius:
- First, take a maxima-saddle pair and the functional value of this maxima and saddle.
- Calculate the distance between maxima and saddle.
- Take a random point on a maxima-saddle arc calculate the distance of this point through maxima and saddle.
- Take the ratio of that distance.
- Then multiply that ratio with the difference in maxima and saddle's functional value, so we get a value, and this is the link radius of that random point on the maxima saddle arc.
- The link radius at the saddle is 0.

- The link radius at the maxima is the difference between the functional value at maxima and saddle.

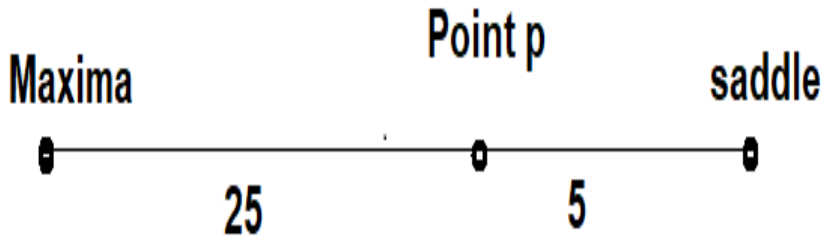

Figure 4.1: Maxima-Saddle pair

Function value of maxima = 100
Function value of saddle = 50
Distance from p to saddle is denoted as 'ps'
Distance from maxim to s is denoted as 'ms'

$$Ratio = \frac{ps}{ms} = \frac{5}{30} = \frac{1}{6}$$

$$r = \frac{1}{6} * (100 - 50) = 8.33$$

- Now we draw circle of radius r or a value proportional to r(link radius need not be the exact value which calculated using above formula) from point p.
- As shown above we calculate the link radius of various points on the arc maxima-saddle and draw circles.
- Then take union of circles and draw a contour(see fig 4.2).

Repeat each step shown above for each extrema- saddle-extrema pair.

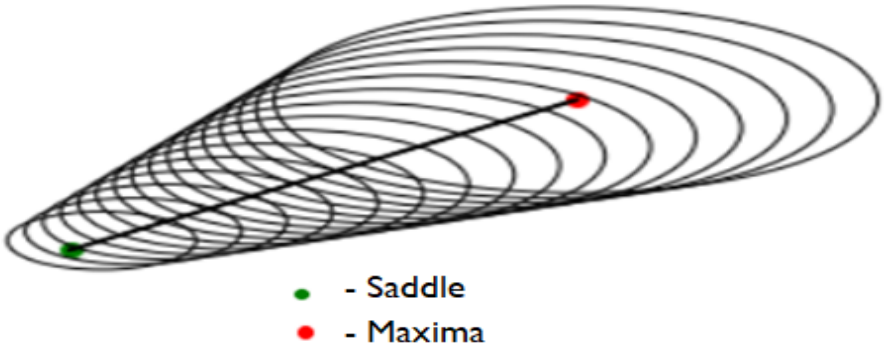

Figure 4.2: Showing contours around maxima saddle as union of circles

4.2.2 Computation of time-varying Topological spines

For computing the time-varying topological spine, first compute topological spine of previous time step then for computing topological spine of next time step we give following things as input:

- The 2-D coordinates of each vertex of previous graph.
- The functional value of each point.
- The edges incident on each vertex.
- Also, the Morse index of the critical point, i.e whether the critical point is a saddle or a maximum.

If there are some new maxima-saddle pairs in next time step points then we follow below steps:

- The initial 2-D coordinates of new points are same as their neighbour so that the correspondence between the previous and new graph should be maintain.
- The functional value of each new point is already given.
- The edges incident on each new vertex is already given.
- Also, the Morse index of the critical point, i.e whether the critical point is a saddle or a maximum is also given.

Now we use a spring layout to calculate the location of vertices of topological spines by fixing the position of previous points and only allow new points to move freely. spring_layout return the location of new vertices.

Now we again give all the vertices to spring_layout this time we do not fix any position so that all the points move freely.

spring_layout return the final location of vertices of topological spine.

Function value of maxima = m
Function value of saddle = s
Distance from maxima to saddle is denoted as 'ms'

$$edge\_weight = \frac{1}{(m-s) * ms}$$

In spring_layout we pass edge_weight as the value of parameter weight

Now we take an extrema-saddle-extrema pair one by one and calculate the link radius used in topological spine computation.

As shown in 4.2.1 we calculate the link radius of various points on the arc maxima-saddle and draw circles. Then take union of circles and draw a contour(see fig 4.2).

Repeat each step shown above for each extrema- saddle-extrema pair.

4.3 Algorithm for computing time-Varying topological spine in brief

---

Input: The 3-D extremum graph or a graph found in previous time step, functional value at each point, Morse-index value at each point

Output: Topological spines.

1: **if** Input graph is of 1st time step **then**
2: Compute its 2-D coordinate as shown in section 4.2.1
3: Compute link radius of various point on maxima-saddle arc as shown in section 4.2.1
4: Draw circles using link radius on various points.
5: Color that circles using colormap and also color them according to the functional value at each point.
6: **else if** Input graph is not of 1st time step **then**
7: Compute its 2-D coordinate as shown in section 4.2.2 with the help of previous graph.
8: Compute link radius of various point on maxima-saddle arc as shown in section 4.2.1
9: Draw circles using link radius on various points.
10: Color that circles using colormap and also color them according to the functional value at each point.
11: **end if**
12: **return** Topological Spines

Chapter 5

Rendering

In this chapter we describe how we render the topological spines.

5.1 Rendering of topological spines

We draw circles of radius proportional to the link radius (inner circles) and take their union which forms a contour passing through the saddle also draw circles of radius slightly larger than the inner circles. We use different color-map for both inner and outer circles around the arc maxima-saddle whose union form contour around maxima and saddle, also use different opacity. The inner circle colored according to their functional value as we can see in fig 5.1 the color is red around saddle and changes to yellow as we moved towards maxima.

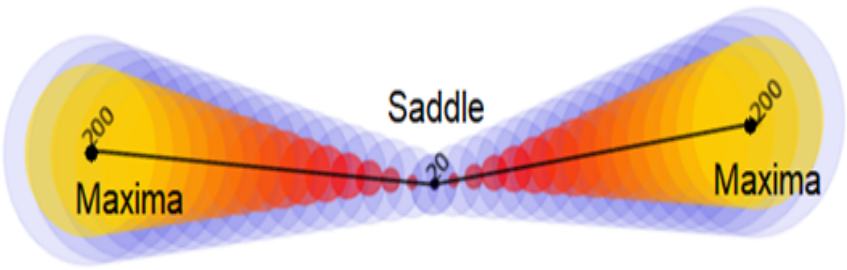

Figure 5.1: Topological spines

Chapter 6

Experiments

We generate two data-sets which is a 3D extremum graph see fig 6.1 which is the first data-set and fig 6.2 which is the second data-set. In both the figures red points are maxima and green points are saddle.

- First data-set

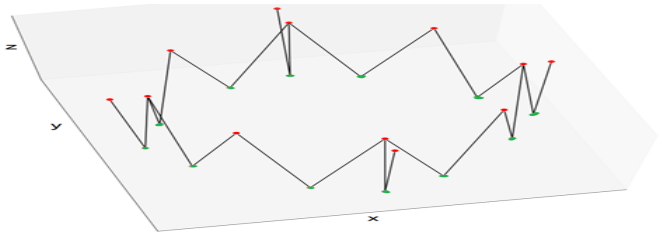

Figure 6.1: 3D Extremum graph of first data-set

- Second data-set

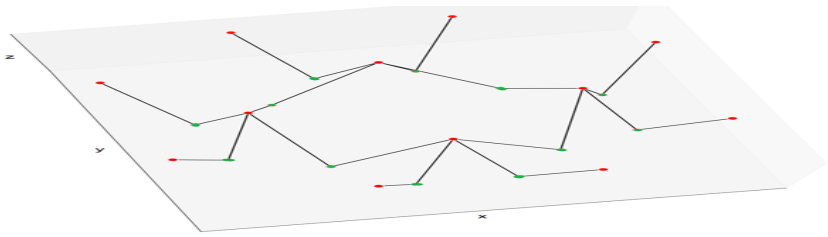

Figure 6.2: 3D Extremum graph of second data-set

6.1 Experiments

- We first use the reciprocal of euclidean distance between two nodes(It is computed using 3D coordinates) as the spring constant between them, because of this new edges come closer to each other, and the location of vertices is also not preserved. To deal with this we first multiply the euclidean distance between nodes by the difference in their functional value and then take its reciprocal as spring constant. The motivation behind doing this is just that if the difference in functional value of maxima and saddle point is large means they are farther from each other and if it is small means they are close to each other in this way we can preserve both distances between the nodes and layout of the graph see fig 6.3. In 6.3 we can see that the highlighted maxima which close in left image is now far away in right image.

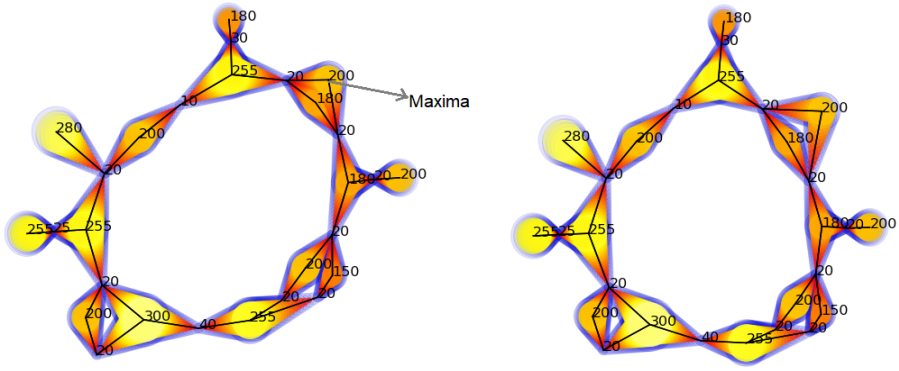

Figure 6.3: Experiments on topological spines

In fig 6.3 left: Computing topological spine using the reciprocal of euclidean distance between two nodes(it is computed using 3D coordinates) as the spring constant between them. Right: computing topological spine using the multiplication of euclidean distance between nodes by the difference in their functional value and then take its reciprocal as spring constant.

Chapter 7

Results

7.1 Topological spines of first data-set

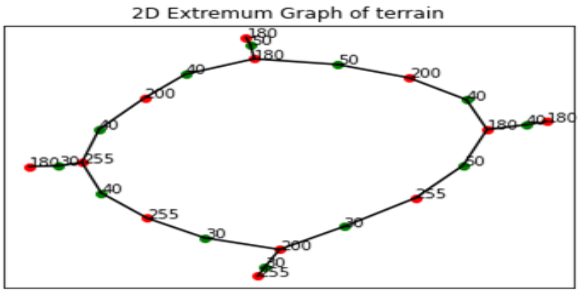

Figure 7.1: 2D extremum graph of first data-set

The fig 7.1 is 2D extremum graph of first data-set where red points are maxima and green points are saddle.

Figure 7.2: Topological spines of 1st and 2nd time step

In 2nd time step of fig 7.2 new maxima-saddle pair appears of functional values are 200,50 respectively and also a maxima appear whose functional value is 255.

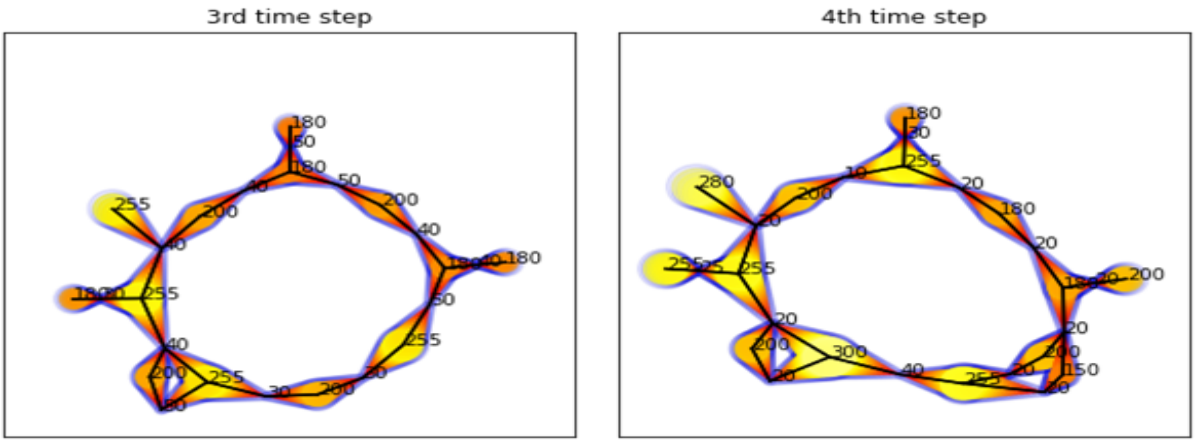

Figure 7.3: Topological spines of 3rd and 4th time step

In 3rd time step of fig 7.3 a maxima-saddle pair disappear whose functional values are 250,30 respectively. In 4th time step of fig 7.3 a new maxima-saddle pair appear whose functional value are 150,20 respectively.

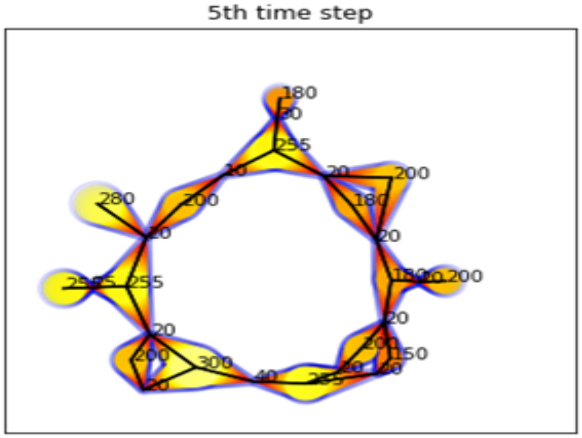

Figure 7.4: Topological spines of 5th time step

In 5th time step of fig 7.4 new maxima appear whose functional value is 280.

7.2 Topological spines of second data-set

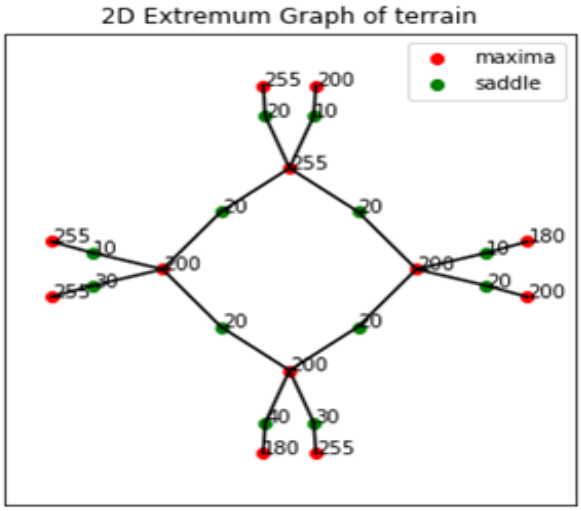

Figure 7.5: 2D extremum graph of second data-set

The fig 7.5 is 2D extremum graph of second data-set where red points are maxima and green points are saddle.

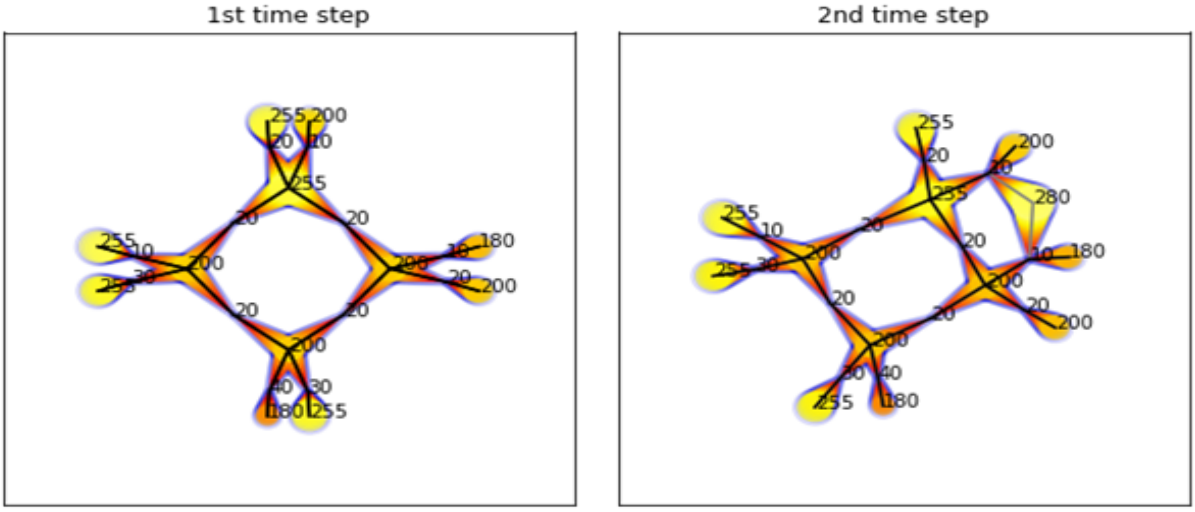

Figure 7.6: Topological spines of 1st and 2nd time step

In 2nd time step of fig 7.6 new maxima appears whose functional value is 280.

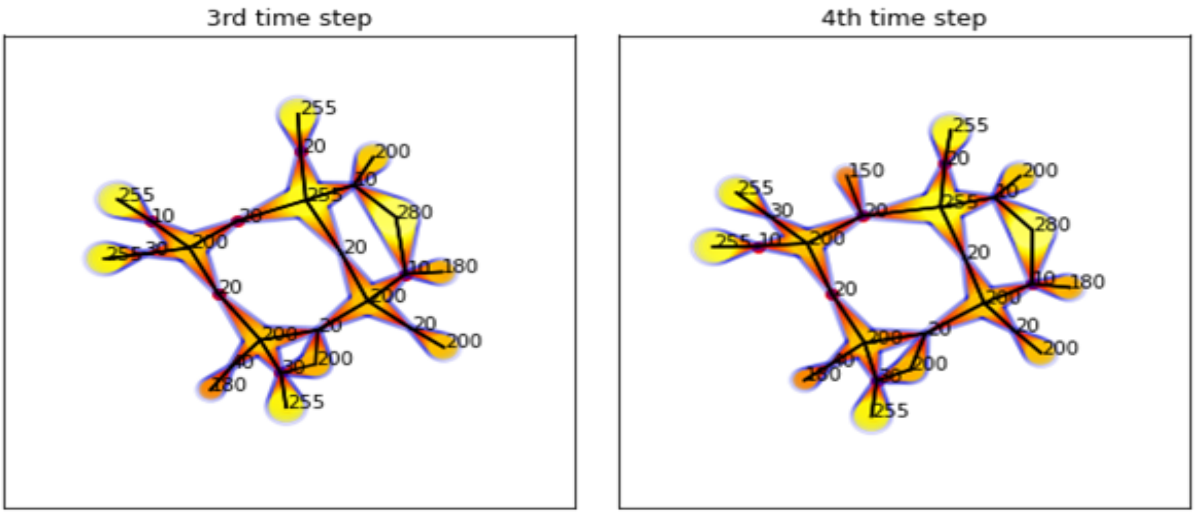

Figure 7.7: Topological spines of 3rd and 4th time step

In 3rd time step of fig 7.7 new maxima appears whose functional value is 200. In 4th time step of fig 7.7 new maxima appears whose functional value is 150.

Chapter 8

Conclusion and future work

8.1 Conclusions

The algorithm for computing topological spines of time-varying data is successfully created. Its main features are:

- This algorithm is successfully preserve the structure of original data set while computing topological spines.
- This algorithm work for both time-varying data and non time-varying data.

8.2 Future Work

Currently, our algorithm does not work for large datasets, so it can be extended for computing topological spines for larger datasets.

Bibliography

[1] Carlos Correa, Peter Lindstrom, and Peer-Timo Bremer. Topological spines: A structure-preserving visual representation of scalar fields. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1842–1851, 2011. v, 1, 2, 5, 6

[2] Himangshu Saikia and Tino Weinkauf. Global feature tracking and similarity estimation in time-dependent scalar fields. In *Computer Graphics Forum*, volume 36, pages 1–11. Wiley Online Library, 2017. 7

[3] Wathsala Widanagamaachchi, Cameron Christensen, Valerio Pascucci, and Peer-Timo Bremer. Interactive exploration of large-scale time-varying data using dynamic tracking graphs. In *IEEE symposium on large data analysis and visualization (LDAV)*, pages 9–17. IEEE, 2012. v, 7, 8

[4] Wathsala Widanagamaachchi, Alexander Jacques, Bei Wang, Erik Crosman, Peer-Timo Bremer, Valerio Pascucci, and John Horel. Exploring the evolution of pressure-perturbations to understand atmospheric phenomena. In *2017 IEEE Pacific Visualization Symposium (PacificVis)*, pages 101–110. IEEE, 2017. v, 7, 8