Recreating Persistent Homology and Force-Directed Graph Layouts

Philip Warton, and Shreyes Joshi

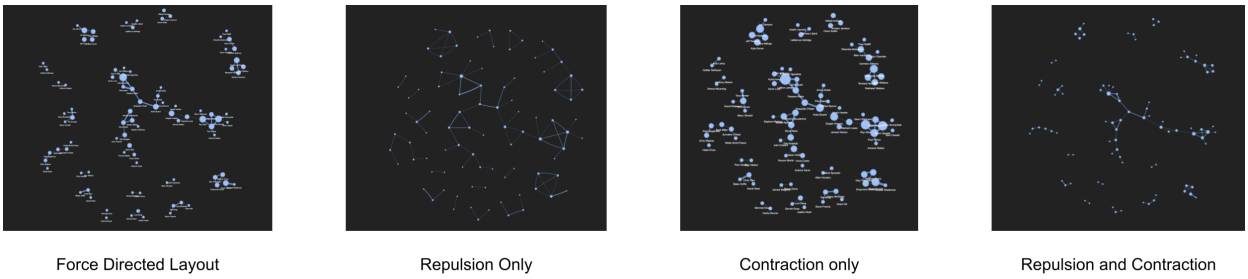

Fig. 1: NBA All-Star teammates. Forces in the graph layout manipulated as labeled

Abstract—Force-directed graphs are a common visualization technique that makes graphs simple and readable. A force-directed graph treats nodes as repelling objects but treats edges as contracting forces, pulling nodes together. Additionally, there is a small central force pulling everything to the middle. The balance of these creates a graph that is spread out but held together by connecting edges. The authors of the paper Persistent Homology Guided Force-Directed Graph Layouts [10] built on this concept. They introduced a barcode display of the persistent homology that controlled the forces in the visualizations. Barcodes could toggle the repulsion of nodes, and a selector controlled the strength of contraction. This paper is an attempt to implement the Persistent Homology Guided Force-Directed Graph Layouts. A free copy of this paper and all supplemental materials are available at https://github.com/joshishreyes/CS553Final.

Index Terms—Graph drawing, force-directed layout, Topological Data Analysis, persistent homology

1 Introduction

In the works of Suh, et. al [10] force-directed graphs are further built upon, interactively. Attempting to push the limits of the visual usefulness of graph visualizations, the authors looked to topology for answers. Their paper allows for the selection and ommision of persistence diagram barcodes, which correspond to extra forces being applied in the context of force-directed graph layouts. This allows the user to experiment with the strength of important connections, making certain clustering and connectivity properties easier to discover. While force-directed layouts, on their own, successfully and with stable results are able to neatly show graphs, but when combined with interactive features can make graph exploration both pheasible and insightful.

Inspired by their paper, we were able to utilize existing interactive graph libraries to explore new network datasets, coming to our own conclusions. And moreover, we implemented the algorithm as described in the paper so that the 0-th persistent homology data from the graph can be extracted. Though we did not implement the added work of contraction and repulsion techniques, through our trials we learned in great detail the implementation of such algorithms, and achieved a much better and broader understanding of network visualizations and analysis.

- *Philip Warton is a student at Oregon State University. E-mail: wartonp@oregonstate.edu.*
- *Shreyes Joshi is a student at Oregon State University E-mail: joshish@oregonstate.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

2 Previous Work

2.1 Graph Theory

There is a long and rich history of the study of graph theory and its applications tracing all the way back to Leonard Euler in the 1700s. [1] Many famous graph algorithms like max flow/min cut were developed throughout the 1900's, and even to this day many graph algorithms and applications of graph theory are being developed, from social networks, to circuits, to the infrastructure of our roads and internet. [7]

2.2 Graph Visualization

We of course derive our work from the paper that we attempt to implement, [10] but it too builds upon previous works to achieve its interactive visualization. The main line of work that has led to their contributions is that of force-directed graph layouts. This is a technique for drawing node-link diagrams, which involves using repulsion forces between nodes and contraction forces along the edges within a physical simulation to spread the graph out in a neat and readable way as in Fruchterman-Reingold [3], and previously Kamada-Kawaii [6]. Interactive such layouts have been explored as well, ranging from basic transformations to click-and-drag-able nodes and even more advanced techniques [7].

2.3 Persistent Homology

Persistent homology, as a field of study, is relatively new [2]. Much of the theoretical understanding is based on the study of simplicial complexes and algebraic topology, which are deep and theoeretical fields in their own right [5], but when applied in the context of data and graph analysis, take on a whole new life.

3 Background

3.1 Graphs

Graphs are a common object of study in computer science, with many practical applications. We define them as follows:

Definition (Graph): An edge-weighted graph $G = (V, E, w)$ is comprised of a set of nodes V and a set of edges where we say $(u, v) \in E$ when the nodes u and v are connected. The weight function $w : E \to R$ assigns real values to each edge. [4]

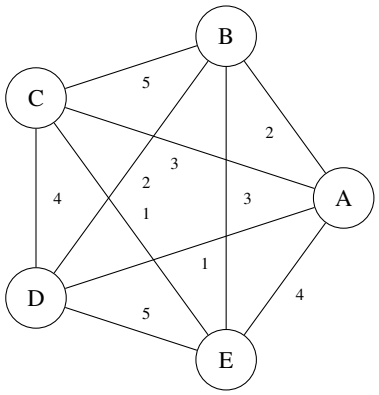

Fig. 2: The complete graph K_5 with edge weights

A graph layout l assigns a position to each node $l : V \to R^2$ which dictates how the graph is drawn to the screen, and how it is visualized. Many times, a layout is defined as a general set of rules that can be applied to any graph (or sometimes and specific class of graph) so that the placement of each vertex is determined based on the structure of the graph.

There are many features and questions that can be observed on graphs. For example, connectedness, which is defined as follows:

Definition (Connected): Two pairs of vertices are connected if there is a path in the graph from one to the other. The graph itself is said to be connected when every pair of vertices within the graph is connected.

A graph can be divided into sections called connected components. These are portions of the graph that are connected by edges. For instance, assuming that edges are all two ways, if there is no path from one node to another, you know they must lie in different connected components. That is there are at least two parts of this graph that are not connected to each other by any edge.

Definition (Connected Component): A connected component of a graph G is any maximal connected subgraph of G.

One key area of mathematics that gives insight into graph theory is that of algebraic topology. It is a way of describing shape more abstractly than geometry, in that objects are topologically equivalent under continuous deformations (no breaks, bends, or folds). It is essentially the study of seperations, holes, voids, etc. as you increase dimension $(0, 1, 2, \ldots$ dimensions, respectively). For instance, the 0-th Betty number (denote β_0) of a graph corresponds to number of connected components of the graph [4].

3.2 Persistent Homology

In topological data analysis, a technique called persistent homology is used to analyse point cloud data, and this can also be applied to graphs. It involves a filtration done on the graph followed by an analysis of how the topology changed over the duration of that filtration.

Definition (Filtration): A filtration on a graph G is sequence of graphs

$$G_1 \hookrightarrow G_2 \hookrightarrow \cdots \hookrightarrow G_i \hookrightarrow \cdots$$

such that $G_i \subseteq G_j$ for any $i \leq j$ and the indices represent the time parameter of some process.

In particular in our work, the filtration consists of building the minimum spanning tree of G, and counting the number of connected components at each step. So we have a sequence of subgraphs of G, each with fewer and fewer connected components.

To capture the topological changes over the filtration we use barcodes. When two connected components get merged into one via the addition of an edge, we have gone down in number of connected components. So, we can think of this as one of the connected components having 'died', while the other 'persists'. In choosing which connected component died, we often judge based off of which emerged first in our filtration. This doesn't work in the case of 0-th homology on graphs, so one can choose this arbitrarily. Barcodes are tuples that give two times, the 'birth' and 'death' of the given homological feature.

Definition (Barcode): We define the p-th barcode of a filtration as the description via tuples of appearance and disappearance of topological features as the time parameter is incremented. That is, if the tuple (t_{birth}, t_{death}) appears in the p-th barcode, then a p dimensional topological feature emerged at time $t = t_{birth}$ and ceased to exist at time $t = t_{death}$.

While these techniques are most often used in topological data analysis [2], they too can be used in the application of graph visualization. In particular, the paper we seek to implement ourselves uses persistent homology to change the forces in the force-directed layout of the graph.

4 Results

We were able to successfully analyze graph data via interactive force-directed layouts. We implemented an algorithm that allowed us to modify the forces in the layout, manipulating the view. The graphs shown below in Figure 3 and Figure 4 are examples of the layouts generated by both contraction and repulsion.

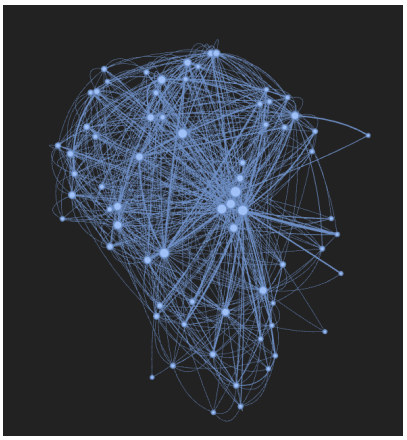

Fig. 3: Delta flights graph with increased repulsion and contraction

Fig. 4: Game of Thrones graph with increased repulsion and contraction

The major takeaway from our testing was the applicability of our methods. We found that the more data points in a dataset, the harder it was to visualize with our methods. While this is inherently true for most graphs, we found it was especially true for force manipulation. Graphs with fewer edges responded far better to contraction, repulsion, and the combination of both. In Figure 1 we saw how the NBA data responded quite well to the different formats. Specifically, the combination of forces created the expected effect, with many strands of compact nodes. That is contrasted to Figure 3, which shows the airport data under the combination of forces. The overlapping of lines makes the graph

unreadable and removes a large portion of its value. It is important to note that the graph in Figure 1 has 92 edges, while the graph in Figure 3 has 772. It is also worth noting that the algorithm performs better with more "sprawling" data. Looking at the Game of Thrones graph in Figure 4, we see significantly better performance, despite only 685 edges. This is because the graph has real patterns that can be seen. Unlike the airport data, not many nodes are connected.

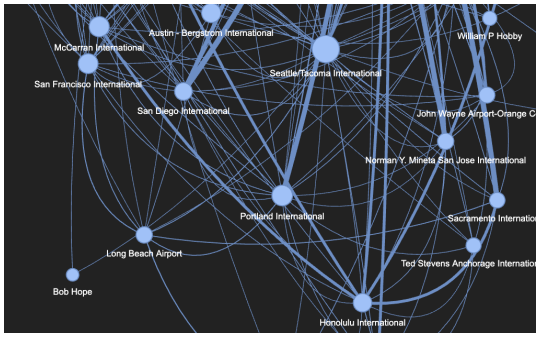

Fig. 5: West Coast airports clustered

That being said, while some graphs were more readable, all graphs had some data to be shared. With the Delta flight data, you could tell the region by the cluster of nodes. For example, the large cluster in the middle includes major central airports like Hartsfield-Jackson Atlanta International Airport, and Dallas-Fort Worth International Airport. In Figure 5 is a group of west coast airports including Portland International Airport, Honolulu Airport, and Seattle Tacoma National Airport. As expected, airports were clustered together by region because flights were more common. Using our algorithms for contraction and repulsion helped us isolate these clusters, and take more away from the data.

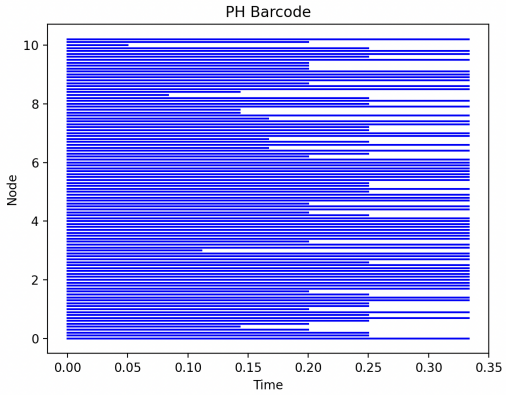

Fig. 6: Game of Thrones Persistent Homology

As can be seen in Sec. 4, the extraction algorithm for persistent homology was successful. Although this isn't the most typical layout for barcodes, it still sufficiently conveys what the data shows. As we can see by the frequency of $(0, 1)$ barcodes, many of the edges with a weight of one are important homological features. It can be interpreted as "much of the connectivity of the Game of Thrones can be attributed to pairs of characters that interact only once". However, there are instances where the opposite is true. Towards the top of the diagram we see a very short bar, and since we are looking at the inverse of the edge weights, this is a pair of characters that interacted approximately 15 - 25 times. Many of these bars land somewhere in the middle, which indicate that the characters have interacted maybe a couple of times, which is to be expected.

Moreover we were able to implement the modified Kruskal's algorithm to generate barcodes from the 0-th homology data on our graphs. However, we were unsuccessful in implementing the homology data into our interactive graph layout as per the paper. Nonetheless, we use the techniques to look at three different datasets.

5 Evaluation

Our initial evaluation strategy was based on the line "user adds and removes contracting and repulsing forces generated by the persistent homology features, eventually selecting the set of persistent homology features that most improve the layout." We wished to evaluate our paper on the same procedure. That is to say, we wanted a user to be able to select sets of persistent homology features, modifying the contraction and repulsion of their graph, until they came up with a satisfactory visual. Unfortunately, we failed to implement part of this goal.

While section 4 outlined how we allowed users to modify the forces in their graph and generate a bar code for the homology features, we failed to integrate these into one component. Our initial goal was to have the barcode control the graph, but it currently functions as an independent display. Repulsion and contraction are controlled by the graph component itself. Because of this, any modification to forces takes effect on the whole graph, rather than groups of nodes.

Because of these shortcomings, we pivoted our evaluation strategy to get a grasp on what we had accomplished. We would consider our project a partial success if we were able to use the datasets to create two accurate components. First, a graph, which represented the node from the dataset connected by the specified edges. It should be possible to pan and zoom around the graph, a user can view individual nodes as well as the whole visualization. The user should also have control over the strength of the contraction between edges, and the strength of repulsion between nodes. Secondly, we needed a persistent homology barcode, which showed the birth and death of topological features through the chosen filtration.

To evaluate our work we used three main datasets:

The Game of Thrones Book 1: [8] In this dataset, each node was a character. An edge represents an interaction between two characters. The heavier the weight, the more interactions the characters had.

Delta Flights: [11] This dataset housed a week of domestic flights on Delta Airlines. Each node is a airport, and each edge is a flight between two airports

NBA All Stars 2010-2016: [9] The final dataset was a list of NBA All-Stars from 2010 to 2016. Each node is a player, and each edge represents players who played together outside of the All-Star game. So any teammates who made the all star game in the same year get an edge drawn between them.

All the datasets were scraped for nodes and edges, and fitted into 3 columns. Source, target, and weight. Since the graphs were undirected source and target were interchangeable for any given edge. With this formatting, we were able to successfully input our data, and create a graph. As desired, a user could zoom or pan around the graph. They could also modify the forces controlling the graph using the sliders on the page. The project also successfully outputted a barcode representing the persistent homology features.

6 Division of Tasks

The tasks were divided into four parts

1. **Data Formatting:** Convert the raw data to a usable set of nodes and edges as a csv file
2. **Graph Creation:** Using the processed csv file make graphs that could be viewed, zoomed, and panned.
3. **Persistent Homology Features:** Implement the persistent homology features in the form of a barcode.
4. **Repulsion and Contraction:** Allow the users to manipulate the repulsion and contraction between nodes.

7 Conclusion

Overall, this project accomplished quite a few of its goals, and was a great introduction to the topic of force-directed graphs and persistent homology features. We were able to implement key parts of [10], and test it on relevant datasets.

That being said, the project left the door open for future improvements. The obvious next step is to implement the remaining features from the original paper. The persistent homology barcode should serve as a user interface, where users can control the forces in the graph. Moreover, when the users control the graph, it should apply only to the groups of nodes represented in the persistent homology barcode.

Following that, the next steps would be similar to the ones discussed in the original paper. Implement the same techniques to visualizations with higher PH.

REFERENCES

[1] S. C. Carlson. graph theory. *Encyclopedia Britannica*, 2023. 1

[2] T. K. Dey and Y. Wang. *Computational Topology for Data Analysis*. Cambridge University Press, 2022. doi: 10.1017/9781009099950 1, 2

[3] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991. doi: 10.1002/spe.4380211102 1

[4] C. D. Godsil and G. F. Royle. *Algebraic graph theory*. Springer, 2010. 2

[5] A. Hatcher. *Algebraic topology*. Cambridge Univ. Press, Cambridge, 2000. 1

[6] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989. doi: 10.1016/0020-0190(89)90102-6 1

[7] A. Majeed and I. Rauf. Graph theory: A comprehensive survey about graph theory applications in computer science and social networks. *Inventions*, 5(1), 2020. doi: 10.3390/inventions5010010 1

[8] M. Marchetti. Game of thrones dataset. https://www.kaggle.com/datasets/mmmarchetti/game-of-thrones-dataset/, 2019. 3

[9] F. Mejia. Nba all-star game 2000-2016 dataset. https://www.kaggle.com/datasets/fmejia21/nba-all-star-game-20002016, 2021. 3

[10] A. Suh, M. Hajij, B. Wang, C. Scheidegger, and P. Rosen. Persistent homology guided force-directed graph layouts. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):697–707, 2020. doi: 10.1109/TVCG.2019.2934802 1, 3

[11] T. X. Flights and airports data. https://www.kaggle.com/datasets/tylerx/flights-and-airports-data/, Unknown. 3