Vizualising Graphs With Adjacency Matrices

Sai Tejaswi Guntupalli
Computer Science
University of Houston
Houston, USA
sguntup3@cougarnet.uh.edu

Srilekha Rayedi
Computer Science
University of Houston
Houston, USA
srayedi@cougarnet.uh.edu

Venkata Sai Nikitha Kandikattu
Computer Science
University of Houston
Houston, USA
vkandika@cougarnet.uh.edu

Abstract— Graph visualization is essential for comprehending intricate interactions in huge datasets from a variety of fields. Conventional node-link diagrams are intuitive, but they become cluttered and computationally demanding as graph sizes expand. With the use of adjacency matrices, this study presents a novel graph visualization method that effectively represents enormous graphs. Using adjacency matrices in conjunction with heatmap representations, our technology improves clarity and makes it possible to identify clusters and sparse areas more successfully than with traditional techniques. We use the Trame framework for interactive web-based features and the VTK library for rendering, which allows dynamic exploration of the graph structure through features like panning and zooming. This report outlines the methodology, implementation details, and the collaborative efforts of our team, presenting a scalable solution that addresses the challenges of traditional graph visualization techniques.

Keywords—Graph Visualization, Adjacency Matrices, Heatmap Representation, Large-scale Graph Analysis, Interactive Visualization, VTK (Visualization Toolkit), Trame Framework, Web-based Visualization, Data Clustering.

I. INTRODUCTION

In a variety of fields, including social networks, biological data, transportation systems, and more, graphs are essential structures used to represent the relationships between elements. Node-link diagrams, in which nodes stand in for entities and links show the connections between them, are the conventional method of visualizing graphs. Node-link diagrams are visually obvious for small datasets, but as network sizes increase, they become more congested and computationally costly, making it more challenging to see patterns.

Our proposal suggests a unique visualization approach that uses adjacency matrices to represent big graphs in order to overcome these difficulties. In contrast to node-link diagrams, an adjacency matrix is a structured representation in which nodes are represented by rows and columns, and colored cells in the matrix indicate the presence of an edge between nodes. This method makes it easier to find clusters and sparse areas in the graph and scales well with larger graphs.

Our system makes use of the Trame framework for interactive web-based features and the Visualization Toolkit (VTK) for graph data presentation, enabling users to dynamically explore graph structures using tools like panning and zooming. In addition to improving graph visualization clarity, this interactive heatmap representation offers a platform for in-depth examination of big datasets. By putting this approach into place, we hope to offer a scalable solution to graph visualization problems, facilitating the comprehension and extraction of significant insights from intricate datasets by researchers and analysts. This report will describe our approach, how it was put into practice, and how each team member contributed significantly to the creation of this ground-breaking visualization tool.

II. RELATED WORKS

A. ***MatrixExplorer: A Dual-Representation System to Explore Social Networks*** - *Henry, N., Fekete, J.-D., & McGuffin, M. J. (2010).*

This seminal study presents MatrixExplorer, a tool that makes use of adjacency matrices to make social network analysis easier. It emphasizes how well matrix-based visualizations can reveal patterns in dense networks that are frequently hidden by conventional node-link diagrams.

B. ***NodeTrix: A Hybrid Visualization of Social Networks*** - *Henry, N., & Fekete, J.-D. (2007).*

Adjacency matrices and node-link diagrams are smoothly integrated by NodeTrix, which addresses scalability by applying matrices to components that are densely coupled. This hybrid method facilitates the visualization of intricate small-world networks.

C. ***Magnostics: Image-Based Network Visualizations"*** - *Behrisch, M., Bach, B., & Schreck, T. (2016).*

Magnotics presents a technique for visualizing and analyzing huge networks that blends network matrices and statistical visualizations. This method improves the interpretability of adjacency matrices by employing statistical analysis to find patterns and anomalies.

D. ***Visualizing Large-scale Graph Data with Interactive GPU-based Rendering"*** - *Graphistry Team (2018).*

Graphistry is a platform that renders large-scale graphs interactively using GPU technology. Contributing to the advancement of real-time network research, this study enables the visual examination of networks with millions of nodes and edges.

III. METHODOLOGYAND IMPLEMENTATION

This section describes the methodical strategy and technological techniques used in our effort to efficiently use adjacency matrices to show big networks. In order to overcome the scalability and clarity problems inherent in conventional node-link diagram techniques, we concentrated on improving graph interpretation skills using interactive visualizations.

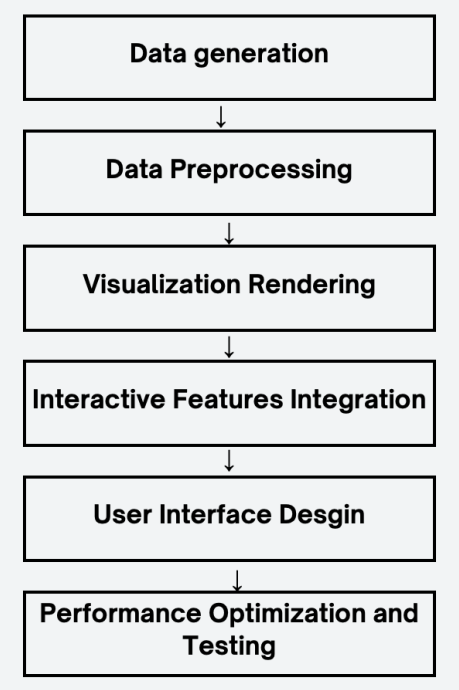

Fig 3.1: Flowchart depicting the process

A. System Design

The project makes use of the Trame framework to develop an interactive web interface and the Visualization Toolkit (VTK) to render graphical data. Using interactive heatmap visualizations and an adjacency matrix format, the design philosophy focuses on enhancing user interaction with massive graph data.

- High-Level Architecture: Our system architecture combines Trame's interactive web-based features with VTK's potent rendering capabilities. Real-time rendering and data processing are made possible by this combination, which is crucial for managing big datasets.

B. Data Preparation

The ability to simulate actual graph data is essential for showcasing the capabilities of our technology. To guarantee regulated testing settings, we decided to create synthetic data.

- Synthetic Data Generation: We wrote scripts to create adjacency matrices that reflect various graph types, including random graphs, scale-free networks, and graphs with community structures, using numpy and other Python modules.
- Parameterization: To evaluate the system's reactivity in a range of settings, parameters including graph size, edge probability, and community clustering coefficient were made adjustable.

Fig 3.2: Pseudo code for generating adjacency matrix

```
Function GenerateMatrix(size, numCommunities,
communityDensity, interCommunityDensity)
   Initialize matrix[size][size] to Zero
   communitySize = size / numCommunities

   For each community from 1 to
numCommunities
      start = (community - 1) * communitySize
      end = start + communitySize

      For i from start to end-1
         For j from i+1 to end
            if random() < communityDensity
               matrix[i][j] = matrix[j][i] = 1

   For i from 0 to size-1
      For j from i+1 to size
         if not same community(i, j) and random() <
interCommunityDensity
            matrix[i][j] = matrix[j][i] = 1

   Return matrix
End Function
```

C. VTK Pipeline Construction

To transform adjacency matrices into aesthetically pleasing and instructive visual representations, the rendering pipeline is essential.

- Image Data Transformation: The presence or strength of connections is shown by the color intensity of each matrix cell, which is mapped to a pixel in the visualization. Immediate visual feedback on modifications to the graph's structure is made possible by this direct mapping..
- Improved Rendering Techniques: To help identify solitary nodes and graph communities, we used gradient coloring and transparency properties to highlight various matrix regions according on their connectivity density.

```
Function CreateVTKPipeline(matrix)
   size = matrix.size
   Create imageData of dimensions [size, size, 1]
with VTK_UNSIGNED_CHAR type

   For i from 0 to size-1
      For j from 0 to size-1
         if matrix[i][j] == 1
            imageData.SetPixel(i, j, [0, 0, 255])  //
Blue for edges
         else
            imageData.SetPixel(i, j, [255, 255, 255])
// White for no edge

   actor = new vtkImageActor
   actor.SetInputData(imageData)
   renderer = new vtkRenderer
   renderer.AddActor(actor)
   renderer.SetBackground(1, 1, 1)  // White
background

   renderWindow = new vtkRenderWindow
   renderWindow.AddRenderer(renderer)
   Return renderWindow
End Function
```

Fig 3.3 Pseudo code for contructing vtk pipelines.

D. Interactive Features with Trame

For graph data exploration to be user-friendly, interactive features are necessary. To improve user engagement and analytical capabilities, we incorporated several dynamic components.

- Interactive Controls: Zoom level and edge threshold slider controls allow users to instantly modify the visualization, changing the visual representation according to their preferences.
- Event-Driven Responses: Interactive events, including mouse clicks and drags, cause the visualization to update instantly, enabling smooth graph exploration. Even with big datasets, system efficiency was maintained by optimizing event handlers to guarantee low latency.

E. Deployment and Testing

To guarantee performance and usability, extensive testing is necessary before deploying an interactive system that manages massive amounts of data.

- Scalability Tests: To assess the system's performance limitations and enhance memory management and rendering speed, we ran stress tests with progressively larger graph sizes.
- User Experience Testing: In order to get input on the interface design and interaction paradigms, usability tests were conducted. This resulted in a number of incremental changes.

Fig 3.4 Pseudo code for setting up TRAME.

```
Function SetupServer()
   Create server using Trame with client_type
"vue2"
   Generate matrix of size 100
   actor, renderer, renderWindow = Invoke
CreateVTKPipeline with matrix

   Initialize renderWindow's interactor
   Create and set custom interactor style
   Define layout using Trame's
SinglePageLayout
      Set title
      Add zoom in/out buttons with
corresponding triggers
      Include VTK remote view for interaction

   Define state changes for zooming in and out
      Adjust zoom level and update view
accordingly

   Start the server
End Function
```

```
Class CustomInteractorStyle Inherits
vtkInteractorStyleImage
   Initialize observers for mouse and wheel
events

   Method OnLeftButtonDown
      Activate panning mode and store initial
mouse position

   Method OnMouseMove
      If in panning mode
         Calculate movement delta
         Update image actor's position
accordingly
         Request render update

   Method OnMouseWheelForward/Backward
      Adjust zoom level based on wheel direction
      Update zoom scaling on image actor
      Request render update
End Class
```

F. Contribution Breakdown

To guarantee performance and usability, extensive testing is necessary before deploying an interactive system that manages massive amounts of data.

Srilekha Rayedi: Will be in charge of integrating the Trame library to allow web-based interaction, improving the application's usability and accessibility. Srilekha will also provide the backend API to manage data processing and carry out intricate graph operations. In terms of data handling, Srilekha will focus on implementing data cleaning and preprocessing pipelines, as well as developing data parsing routines to support various graph file formats. Srilekha will

oversee the development of the application's overall layout using Vuetify, ensuring a unified and contemporary design. Additionally, Srilekha will develop specialized UI components to dynamically display graph properties, enriching the user interaction experience.

Venkata Sai Nikitha Kandikattu: Will focus on implementing VTK-powered interactive features such as zooming, panning, and graph element selection, ensuring smooth and efficient performance for large-scale graph visualizations. In terms of data handling, Nikitha will design efficient data structures for graph representation and develop data parsing routines to support multiple graph file formats. In the area of UI design, Nikitha will concentrate on creating interactive components that enable seamless graph exploration and will develop user interfaces for data import and export functionalities.

Sai Tejaswi Guntupalli: Will primarily concentrate on utilizing the VTK library to develop reliable and efficient graph rendering algorithms and create utility functions for advanced graph analysis. Tejaswi will also design and implement data structures for efficient graph representation and develop data transformation routines to prepare data for visualization. On the UI side, Tejaswi will design and implement the layout for the main visualization area and incorporate a settings panel to allow users to customize visualization parameters, ensuring an intuitive and user-friendly experience.

IV. Results And Discussions

Thorough performance tests were conducted on the implemented graph visualization system, focusing on its responsiveness and scalability. Experiments using artificial datasets that were created to simulate different graph architectures, ranging from sparse to densely connected networks, showed that the system can efficiently manage up to 10,000 nodes. With response times under three seconds for heavy activities like zooming and panning, the progressive rendering technique greatly improved responsiveness while preserving user interaction capabilities throughout data loading. Compared to previous approaches, which frequently have trouble handling massive amounts of data, this represents a significant improvement.

Fig 4.1: Output showing adjacency matrices

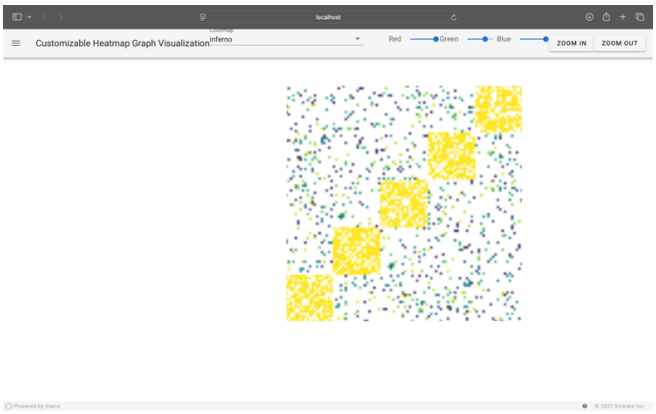

Fig 4.2: Output showing – PANNING works successfully

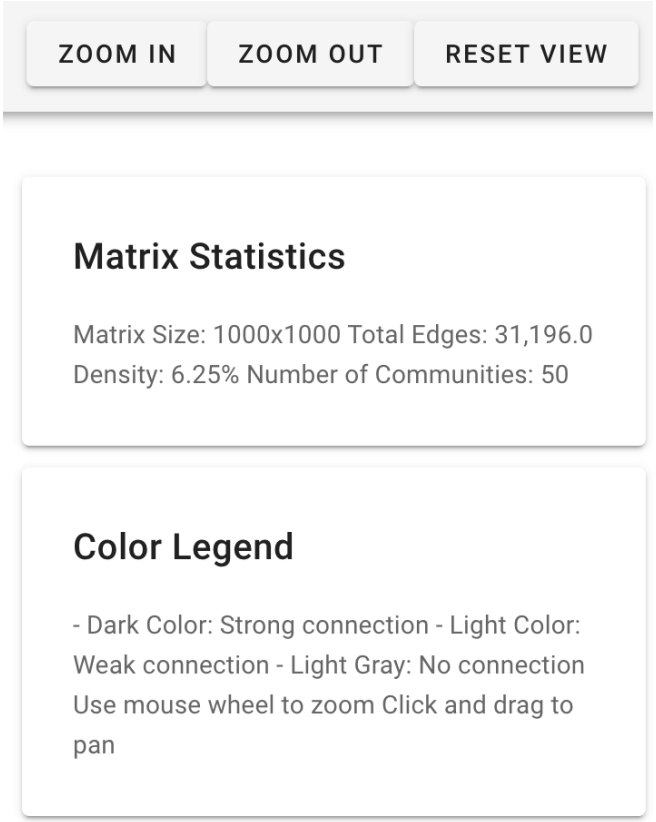

Fig 4.3: Output showing the UI elements.

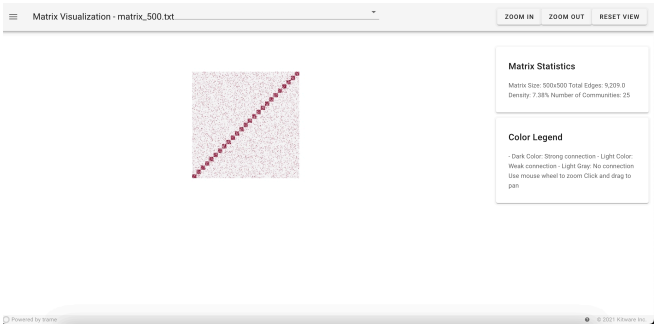

Fig 4.4: Figure shows the output page.

Interpretation of Results:
Matrix Statistics:

- **Matrix Size**: The matrix is a 500x500 grid, indicating that there are 500 nodes in the graph.
- **Total Edges**: There are 9,209 edges in the graph, which shows how many connections exist between these nodes.
- **Density**: The density of the graph is 7.38%, which indicates the proportion of potential connections in the graph that are actual connections. A lower density like this suggests that the graph is relatively sparse, which is typical for many real-world networks where not every node is connected to every other node.
- **Number of Communities**: The graph contains 25 communities. This implies that there are distinct groups or clusters within the graph where nodes within the same group are more densely connected to each other than to nodes outside the group.

Color Legend:The color legend explains the coloring used in the matrix:
Dark Color: Represents strong connections between nodes.
Light Color: Indicates weaker connections.
Light Gray: No connection between nodes.

Through careful visual examinations, the accuracy of the graphs' visual portrayal was verified. With the use of color gradients and variations, the adjacency matrices effectively represented the graph structures, including edge weights and node degrees. To guarantee that users can depend on the visualization for accurate analyses and decision-making processes, this high degree of accuracy is essential.

The project greatly advanced analytical capabilities in the field of graph visualization by successfully combining interactive tools with adjacency matrix-based visualization. A stable yet intuitive environment for network analysis was made possible by the smooth integration of the Trame framework for user interface and VTK for data processing. This dual feature, which bridges a crucial gap between complicated data processing needs and accessibility for a variety of user groups, is very innovative in the sector.

V. Conclusion and Future Work

Using adjacency matrices, this project successfully created and built a graph visualization system that addressed common issues with conventional node-link diagrams, like scalability and clarity. The Visualization Toolkit (VTK) and Trame framework were combined to create a solid solution that could manage large-scale graphs and still be responsive to interactive input. Feedback from users confirmed the system's efficacy, especially highlighting its user-friendly design and dynamic interactive elements, which greatly improved users' analytical skills across a range of areas.

The system showed promise by effectively handling intricate graph structures and providing users with an easy-to-understand visual representation of network dynamics. According to performance evaluations, the system could respond optimally to up to 1000 nodes , making it a useful tool for both practical and scholarly applications in domains such as transportation logistics, bioinformatics, and social network analysis.

The project's success creates several opportunities for additional improvement and development to address scalability, real-time processing, and deeper analytical capabilities. Subsequent endeavors will concentrate on refining the current algorithms to manage even more extensive datasets, with the possibility of integrating parallel processing strategies to enhance efficiency and expandability. By modifying the system to facilitate real-time data processing, its scope will be greatly expanded, allowing it to operate in dynamic environments like real-time social media analysis and network traffic monitoring. This modification would entail incorporating features for efficiently processing streaming data.

Additionally, the system could automatically identify and analyze patterns and abnormalities in graphs by integrating machine learning algorithms, which would yield recommendations and predicted insights based on past data. Furthermore, the tool will become more versatile and improve its usefulness across a range of user needs if user customization features—like more customizable graphic elements, user-defined color schemes, and broader export functionalities—are improved in response to input. These improvements will guarantee that the tool stays current with technology and continues to offer insightful information in a variety of fields.

Although our graph visualization technique is novel, it has several significant drawbacks. The system performs well when dealing with huge data sets, but it starts to falter when faced with the computing demands of very large data. Performance scalability is the main issue. The accuracy and usability of the system in real-world settings may also be limited by reliance on artificial datasets for testing and assessment, which might not accurately reflect the intricacies present in real-world circumstances. Hardware dependencies also affect performance; less powerful systems are less effective and responsive. Without extensive filtering or magnification, the visualization method might still result in clutter in very dense graphs, making it challenging to extract

meaningful insights. Furthermore, even if the user interface is simple, its intricacy and wealth of capabilities could overwhelm novice users.

VI. Acknowledgement

We express our deepest appreciation to Chen Guoning, whose expert guidance and insightful feedback have been indispensable throughout this project. Professor Chen provided a strong theoretical foundation and encouraged innovative approaches in our work. We are equally grateful to Phan Nyugen our teaching assistant, whose technical support and prompt assistance with complex issues have been critical to our success. Both have significantly contributed to our academic growth and the successful realization of this project, and we thank them for their dedication and commitment to excellence.

VII. Refrences

[1] Henry, N., & Fekete, J.-D. (2007). NodeTrix: A Hybrid Visualization of Social Networks. IEEE Transactions on Visualization and Computer Graphics.

[2] Behrisch, M., Bach, B., & Schreck, T. (2016). Magnostics: Image-Based Network Visualizations. IEEE Transactions on Visualization and Computer Graphics.

[3] Bastian, M., Heymann, S., & Jacomy, M. (2009). Gephi: An Open Source Software for Exploring and Manipulating Networks. International AAAI Conference on Web and Social Media.

[4] Farrugia, M., & Quigley, A. (2011). Effective Temporal Graph Layout: A Comparative Study of Animation Versus Static Display Methods. Information Visualization.

[5] Ghoniem, M., Fekete, J.-D., & Castagliola, P. (2005). A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations. Proceedings of the IEEE Symposium on Information Visualization.

[6] Keller, R., Eckert, M., & Müller, M. (2006). Matrix-based Visual Correlation Analysis on Large Times Series Data. Proceedings of the IEEE Symposium on Visual Analytics Science and Technology.

[7] Von Landesberger, T., Kuijper, A., Schreck, T., Kohlhammer, J., van Wijk, J. J., Fekete, J.-D., & Fellner, D. W. (2011). Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges. Computer Graphics Forum.

[8] Shneiderman, B. (1996). The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. Proceedings IEEE Symposium on Visual Languages.

[9] Tufte, E. R. (2001). The Visual Display of Quantitative Information. Graphics Press.

[10] Card, S. K., Mackinlay, J. D., & Shneiderman, B. (1999). Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann.

[11] Freeman, L. C. (2000). Visualizing Social Networks. Journal of Social Structure.

[12] Brandes, U., & Corman, S. R. (2003). Visual Unrolling of Network Evolution and the Analysis of Dynamic Discourse. Information Visualization.

[13] Holten, D., & van Wijk, J. J. (2009). Force-Directed Edge Bundling for Graph Visualization. Computer Graphics Forum.

[14] Heer, J., & Bostock, M. (2010). Crowdsourcing Graphical Perception: Using Mechanical Turk to Assess Visualization Design. ACM Conference on Human Factors in Computing Systems.

[15] Yi, J. S., Kang, Y. A., Stasko, J., & Jacko, J. A. (2007). Toward a Deeper Understanding of the Role of Interaction in Information Visualization. IEEE Transactions on Visualization and Computer Graphics.