Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Adaptive Network Visualization

Gregory Partridge

Supervisor: Prof. John Dingliana

April 17, 2023

A Masters dissertation submitted in partial fulfilment
of the requirements for the degree of
MCS (School of Computer Science and Statistics)

Declaration

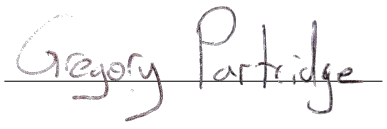
I hereby declare that this Masters dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

I consent / do not consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

I agree that this thesis will not be publicly available, but will be available to TCD staff and students in the University's open access institutional repository on the Trinity domain only, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement. **Please consult with your supervisor on this last item before agreeing, and delete if you do not consent**

Signed:  Date: 17/04/2023

Abstract

The project implements real-time visualisation by applying filtering and abstraction in order to reduce the complexity of network data sets while maintaining the core information and structure of the original data set. Filtering removes isolated and low priority nodes based on centrality and coreness values. Abstraction reduces the size of the data set by identifying cliques and visualising these using glyphs. It is important that these techniques are used without removing information, in other words by reducing the number of nodes presented in a chart while allowing the core connections between the nodes to remain.

The project initially includes a summary of the main chart types and the filtering and abstraction techniques that are described in the published literature, namely: the force directed layout; the adjacency matrix; the arc diagram; basic and curved radial layout; filtering using network coreness and centrality; abstraction using cliques; and then brushing. These techniques are then evaluated and adapted into a design for implementation. The resulting system is then evaluated using a range of data sets. It is shown to be a highly interactive tool that allows the user to control the level of filtering and abstraction. This tool is suitable for use with a wide range of data sets.

This project demonstrates a range of filtering and abstraction methods on multiple chart types. It utilises a multifaceted display that is updated concurrently to changes from filtering and abstraction. Reduction techniques are applied globally to all chart types and these allow manual filtering and the reintroduction of nodes to update the filtering techniques in the newly altered network. A novel clique filtering implementation for clique representation is also implemented.

Acknowledgements

I would like to acknowledge the support of my adviser, Dr. John Dingliana. His willingness to offer his time and knowledge was invaluable to the completion of this dissertation.

Additionally I would like to thank my family and friends for years of patience and support throughout my college years. The help, encouragement and support allowed me reach where I am now and I am doubtful I could have gotten to where I am alone.

Contents

List of Figures

List of Tables

1 Introduction

In network visualisation, the process of visually presenting network data sets is the core goal of the process. In doing so it is important to be able to display key information in a simple and informative manner. This is not always demonstrated as, for example, network data sets can be crowded with many overlapping edges, nodes clumping and generally forming a hairball graph, in which it is difficult to find information within. The core of this project is to apply the techniques of filtering and abstraction in a manner that reduces visual complexity and keeps the core information of the data set and the structure of the original data set. Identifying which techniques to pursue, how to apply these techniques, the degree of which to apply and evaluate them will be the issues that this dissertation will attempt to address.

1.1 Project Overview

At its core, a network consists of nodes and edges that connect the nodes together to create what is the network. Filtering can be applied to remove isolated nodes and nodes of low centrality or coreness. These nodes can be determined in multiple ways ranging from its coreness or centrality value. Abstraction is the process of reducing the size of data sets without removing information. While filtering may remove nodes, abstraction techniques will group nodes together in a glyph to contain the same structure. This will reduce the amount of nodes presented in the chart while allowing the connections between nodes to remain.

Networks can be demonstrated many different charts that uniquely compliment specific data set attributes. Utilising these charts, it is possible to view data sets from different perspectives to gather unique insights into the data set. I will be exploring chart types and how they interact with network data sets.

Filtering and abstraction will be applied across all data sets to demonstrate how changes alter each chart type. The levels of filtering and abstraction will be controlled by the user.

At its core, this project seeks to visualise a network data set through a range of charts, then to apply varying filtering and abstraction techniques in order to thereby visualise an altered data set in which changes are concurrent across all graph types.

The project explores undirected network data sets. While it can process directed network data sets and tree network data sets, it is not specialised for these tasks. It does not use sata set attributes exceeding the links between nodes in order to have a wide a range of data sets as its input. The project explores different chart types that are specialised for different uses in order to be adaptive across as wide a range of data sets as possible. This is in order to not just be able to view multiple data sets, but to accurately present them in the medium that best demonstrates the data set.

The techniques used to reduce the amount of activity on screen are filtering or abstraction. For filtering, there is the process of removing nodes, removing nodes based on node centrality and coreness. For abstraction cliques are identified and visualised by using glyphs. This reduces clutter by grouping interlinked clusters.

1.2 Motivation

When visualising a network data set within network visualisation, reduction of elements is a common technique to get a clearer view of the data set. Also, in doing this, it is important to keep the key data set features intact. Also, in doing this, one motivation for this project is to give the user confidence in being able to reduce elements in the graph while retaining the the core connections and consistent centrality across the visualisation.

When the user is reducing elements, it is important that they have as much control over the two reduction types, filtering and abstraction, and also the level of filtering and abstraction that they apply. This allows the user to control the adaption of the data set to explore the visualisation.

When using the interface, interactivity is a high priority. Giving the user the ability to control the view of the data set, how much they wish to show along with manual removing of nodes or brushing of nodes and their corresponding edges is important.

Providing a multifaceted display allows changes applied to one chart to be applied across all charts in real time. This gives us a better understanding of how changes interact with each chart and also demonstrate how the filtering and abstraction techniques can alter the visualisation of the data set. This should give the user better understanding of the various techniques and better understanding of their data sets.

Providing a wide range of uses is a core motivation in this project. This means dealing with a wide range of data sets as possible across a wide range of uses and cases. A lot of the

major decisions in this project are formed around the ideal to keep a wide range of possible data sets as intake.

1.3 Methodology

This chapter gives an overview of the macro components of the project. These are also the core components that impact the development of the project.

1.3.1 Adaptivity

Adaptivity is used to increase the quality of the users experience. It being a core area of this project, it is best seen in how changes applied to a chart are applied across all idioms. Adaptivity is also present in data set intake where if it contains the edges it can work with that data set. Adaptivity is also demonstrated with the force directed layout charts in order to adapt its structure based on the unique node placements that are dependent on the data set. This will auto-correct itself when filtering or abstraction is applied or redacted showing adaptivity to the requirements needed for the idiom.

The reason behind including adaptivity within this program ultimately depends on the particular features. For instance, for applying changes globally across all idioms it is to allow changes applied to one chart to be visible on other charts. This can show the user exactly what the changes applied are doing to their data set. For adaptivity across the different data sets it is in order to allow the application to be as universal and have as many use cases as possible. The reasoning for the force directed layout adaptivity is to show clearly how applying filtering and abstraction change the structure of the visualisation. This will clearly demonstrate core changes within the visualisation.

1.3.2 Interactivity

When referring to interactivity in data visualisation, it is in reference to the use of tools and processes to produce a visualisation of a data set where it is possible to explore, analyze and abstract within the original visualisation. It is used to allow the ability to discover insights and discoveries within a data set.

Interactive systems have been implemented prior to this work, such as Becker [11] where early examples of interactivity was demonstrated through manipulation techniques for revealing relationships. Similar techniques will be demonstrated within this project. Another example is by Namata [12], where the use of multi coordinate view is demonstrated that cross references subsets of the the same data set. Both projects highlighted areas to incorporate within this project's interactive display.

1.3.3 Multifaceted

Multifaceted refers to having a variety of views of a data set. In this case it demonstrates multiple idioms of the same data set. These idioms are able to provide different perspectives of the data set to provide a manner of unveiling and discovering insights. A multifaceted approach is important due to its allowing a quick overview of all the information present and allowing the user to inspect areas they personally find is relevant to them. This project allows changes to be seen globally. Alongside this, it enables the user to have a wide perspective of the data set. Multifaceted is a core aspect of the adaptibility of the application and creates an easier experience interacting with and extracting information from their data set.

An example of multifaceted interfaces is shown by Cao [13] which presents a visualisation tool called FacetAtlas. This allows a number of high level features to aid the user through a multifaceted display using filtering and content switching. These attributes highlight the uses of a multifaceted display in order to demonstrate multiple idioms.

1.3.4 Sub Charts

Sub charts are a way in which to show all idioms of the data set concurrently. They are vital for tasks such as the previously mentioned multifaceted approach 1.3.3. The core reason for implementing this is to allow the user to see changes applied to one visualisation applied to the other charts and to view this from a high level. To explain in greater detail, this allows the display of all graph types to viewing while also allowing for closer inspection in order to see the idiom in greater detail. Reducing the information, as shown for exampled by McLachlan [14] is necessary to reduce clutter within the smaller viewing window. Viewing the chart in a larger space enables more detail for secondary characteristics but needs to be tuned for the best experience. The implementation of sub charts at a high level is that a reduced visualisations of each chart type is transposed to their respective sub chart in which features of lesser importance are removed to not clutter the panel given it's reduced viewing area. This allows changes applied to the selected data set to be visible within the sub charts.

1.4 Contributions

The contributions of this research present in this dissertation are:

- A core contribution is to compare different filtering and abstraction techniques and their efficiency at reducing the clutter in the visualisation. Alongside this, it will demonstrate whether the techniques can maintain the graph's centrality, core connections, structure and information. This will be tested for varying levels of

filtering alongside similar testing for abstraction. This is to demonstrate how the various techniques alter the original data set.

- Utilising multi coordinated views, have changes applied to one idiom apply globally and concurrently across all other idioms. Changes applied on one chart can be seen on all others to get greater insight as to how the techniques apply to the data set. This is important as to highlight how the visualisations change with each additional level of reduction applied. This is due to some charts being specialised towards adapting their structure to the change applied while others demonstrate how the alterations applied interact with the structure of the original data set. These idioms can also demonstrate how they alter and/or tidy the connections with the techniques.
- This project provides a novel approach to clique prioritisation. A graph can have multiple cliques with overlapping nodes contained within it. One must decide which to prioritize and visualise as well as filtering out cliques of less size. This is required for abstraction as when replacing nodes in cliques with glyphs, multiple glyphs can not represent the same nodes and as such need to define a method for representing a clique over another clique. The methodology behind this is further expanded upon in section 3.10.

1.5 Dissertation Structure

The dissertation has been written in a style that assumes no knowledge about the intricacies of the used graph charts along with the specifics of the filtering and abstraction techniques brought forward. The dissertation is structured as follows:

- Chapter 1: Introduction- An overview of the topic along with the motivations that will be discussed within the later sections alongside the objectives and contributions of the work.
- Chapter 2: Background- Literature review of papers related or that contributed to the work. Includes also necessary background knowledge to understand the presented work and its contributions. This includes the various graph types, the filtering techniques, abstraction applied, adaptivity present, multi-faceted display and brushing applied.
- Chapter 3: Design- Discussion of the theoretical overview of implementation and the design choices employed across the project as well as the reasoning and justifications for pursuing specific directions within the project. This section includes pre-processing strategy, the user interface and the filtering and abstraction tuning employed.
- Chapter 4: Implementation- how the topics discussed prior to this were implemented. This will include pseudo code and specific strategies for implementation. This section

includes pre-processing for clique detection and clique prioritisation, idiom implementation, filtering and it's techniques, abstraction and clique visualisation, filtering and abstraction tuning and brushing.

- Chapter 5: Evaluation- how the application was tested and the methodology when evaluating the performance. This will consist of the average FPS rates for various data sets, the average pre-processing run times for a range of data sets and a visual comparison to demonstrate how the data set visualisation is transformed.
- Chapter 6: Conclusion- summary of the achievements of the project alongside a discussion of the limitations encountered and possible future work to explore.

2 Background

2.1 Chart Types

Graph networks consist primarily of nodes and their corresponding edges. There are other attributes that can be applied such as weight, distance, direction and many more though we will not be looking at these because they limit the amount of data sets that are usable by this application.

There are multiple techniques that can be applied to demonstrate a graph network with different strengths and weakness depending on what you wish to convey. Many graph types are available and while we will be discussing many, some are not implemented such as Sankey charts, chord diagrams and more. The idioms that will be discussed are the:

- Force Directed Layout
- Adjacency Matrix
- Arc Diagram
- Basic, Curved and Hierarchical Radial Layout

The decision about these idiom types was based on flexibility and individual specialisation in representing data sets. Another factor was the range the idioms cover alongside the different views tailored for the structure of the data sets. Force directed layout is an excellent idiom for small to mid sized networks. It also demonstrates how the visualisation structure changes when reduction is applied. The adjacency matrix is specialised towards large data sets along with showing where within the data set there are large clusters occurring. Arc diagrams are an excellent manner for identifying interconnected sub graphs and bridges between such clusters. It is used for smaller data sets but has uses due to its guaranteed space for labeling vertices. The radial layouts are implemented to demonstrate small to mid sized data sets and are favoured when trying to optimize the space around each vertex for the connected edges.

2.1.1 Force Directed Layout

A force directed layout is an algorithm that is used to create a tidy chart to represent a graph by minimising overlap and evenly distributing nodes and edges. It does this by evenly distributing nodes and edges in a manner where links are of similar length, reduce clumping of nodes and the overlap of edges (see Figure 2.1) [1].

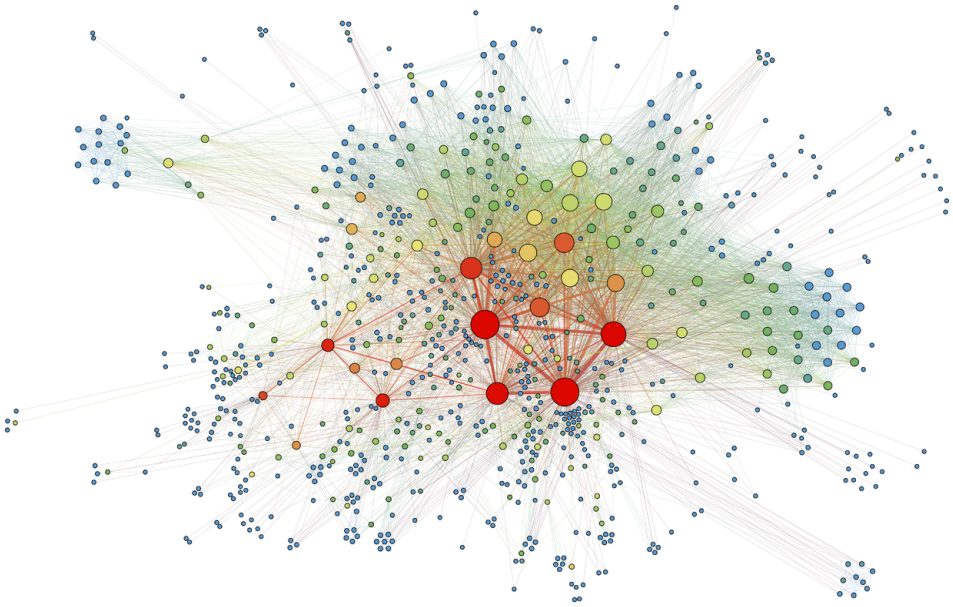


Figure 2.1: Force Directed Layout [1]

It most commonly calculates the layout of undirected charts, though there are alternatives with directed graphs to represent directed edges. They primarily only use the information of the structure of the graph itself. The resulting visualisations are aesthetically pleasing, exhibit symmetries, and produce layouts with reduced edges crossing over. Traditionally, most force directed algorithms use a spring layout method in which spring forces repel each other. They use repulsive force between nodes and attractive nodes between nodes containing a bridging edge.

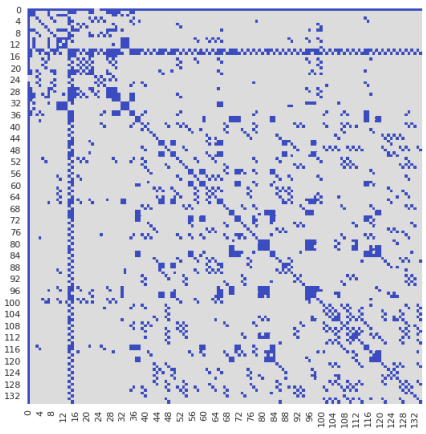
According to Kobourov, force directed graphing algorithms are best suited for small graphs, stating "The utility of the basic force-directed approach is limited to small graphs and results are poor for graphs with more than a few hundred vertices." [15](p384). Graphs with excessive nodes give diminishing performance. This is because of scalibility, as with each node needing to interact with every other node in the graph, in real time. While there are methods of reducing the amount of nodes each node communicates with, this leads to a reduction in the graph quality.

There are methods to create force directed layouts that circumvent these issues. By creating the graph over an extended period of time it is possible to create a force directed layout graph with a large quantity of vertices. But due to the fact that this can not be applied to

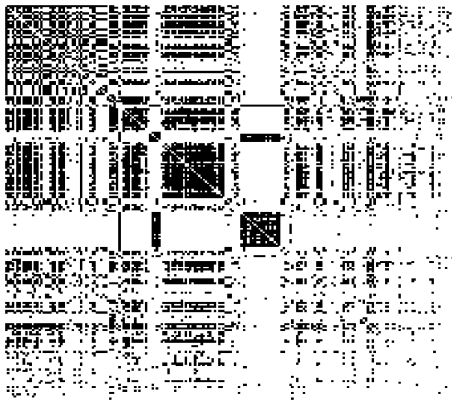
real time application due to time constraints in developing such a complex visualisation it will not be explored further.

2.1.2 Adjacency Matrix

An adjacency matrix is a matrix with rows and columns to indicate whether pairs of vertices are adjacent or not in the graph. They are used to show relationships between nodes through association and can very effectively show where clusters are in data sets. Larger clusters indicates high connectivity. Clustering of data beforehand can help identify and visualise the clusters. Adjacency matrices also help show the level of connectivity of a graph as a whole. An example of an adjacency matrix can be demonstrated by Ganaye [16] where a binary adjacency matrix is demonstrated in Figure 2.2a. An application of an extremely large and dense data set is also demonstrated with Figure 2.2b.



(a) Labeled Binary Adjacency Matrix [16]



(b) Adjacency Matrix of bird relationships between bird-boxes

Figure 2.2: Adjacency Matrix Implementations

Ghoniem [17], when comparing the readability of graphs using a node link diagram and a matrix based representation, found that adjacency matrices had been outperforming node link diagrams for large and dense graphs. Specifically, when comparing how people performed low-level tasks required for reading a network, they found that with a range of density and size of the networks, in eight out of nine occasions the matrix outperformed the node link diagram. This gave confidence that an adjacency matrix would be beneficial for the application due to the variety of size and density of data sets the application could ingest.

Mueller [18] has suggested one possible route, using virtual similarity matrices. Virtual similarity matrices is a technique applied to graphs of very large size. When given a graph with a set of vertices and edges, a virtual similarity machine is generated by labeling the horizontal and vertical axis and labeling each point that is an edge to a point. This is binary and allows for representation of multiple points to easily represent a data set.

2.1.3 Arc Diagram

Traditionally, an Arc layout has aligned nodes along a line, with the edges being displayed as arc connecting the nodes with the height of the arc being proportional to the distance between the nodes. Alongside this they are useful for labeling nodes due to having guaranteed space for each node, which is not a given in most two dimensional structures. It is not as effective as conventional two dimensional layouts, though with optimal node layout it allows an easy environment to identify cliques and bridges. To best display this, clustering of nodes within the data set should be applied prior to visualisation of the arc diagram. An example of an arc diagram can be demonstrated by data-to-viz [2] (see Figure 2.3).

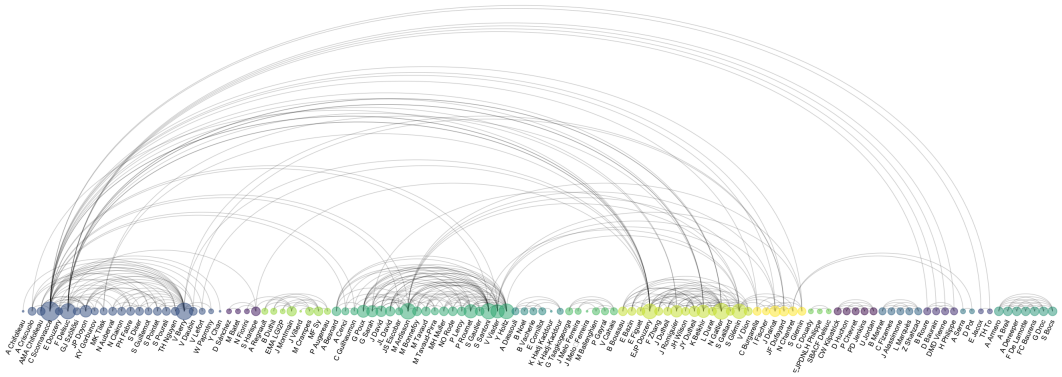


Figure 2.3: Arc Diagram [2]

The varied collection of arc diagrams displayed by Nagel and Duval [19] demonstrates how varied the idiom is along with the specifying the characteristics that is differentiated between the arc diagrams. The main categories listed being node distance, node serration, arc directionality, arc weight and the degree of connection. This does not include stylistic characteristics such as arc encoding channels, arc type or orientation. To conclude, this paper raised many questions in how to approach the arc diagram in representation.

2.1.4 Basic, Curved and Hierarchical Radial Layout

In the radial layout, the nodes are set in a circular manner (see Figure 2.4). When the data set is hierarchical there can be alterations of the formatting in which the nodes are set in a collections of circles. The basic and curved radial layouts are one dimensional in depth. These charts can commonly group nodes when the data set has categorical data. This is to better demonstrate connection patterns between the groups. These one dimensional radial layouts conventionally lay nodes out spaced linearly which is particularly useful when the data set has a cyclic nature to the data set.

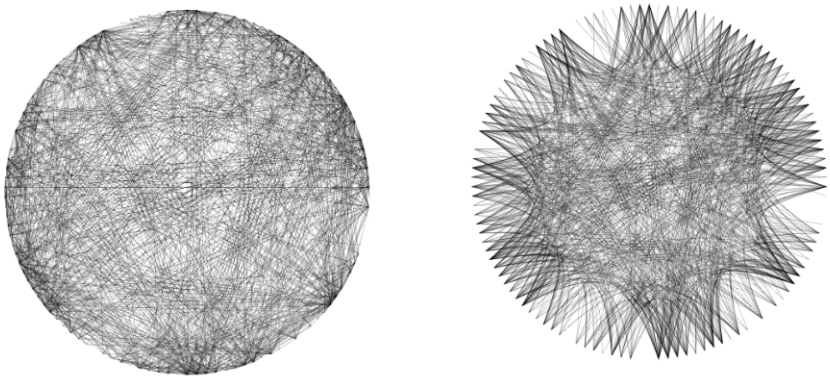


Figure 2.4: Basic Radial visualisation & Curved Radial visualisation

Conventional radial layouts are suited to smaller and mid-sized data sets ranging up to the low hundreds of nodes. When the data set is too large the nodes start to become indistinguishable. This leads to wasted space in the visualisation and issues with effectively displaying the chart on screen. These issues can be circumvented by employing methods of tracing links. An example of this is edge bundling, which reduces clutter as well as employing a pleasing aesthetic with the trade-off of making it harder to trace individual edges between vertices as demonstrated by Holten [3] (see Figure 2.5). This has been documented where scalability is tackled through trace exploration, feature location and top-down analysis [20]. Within this work there is a demonstration of edge bundling with clear a indicator of its ability to reduce clutter.

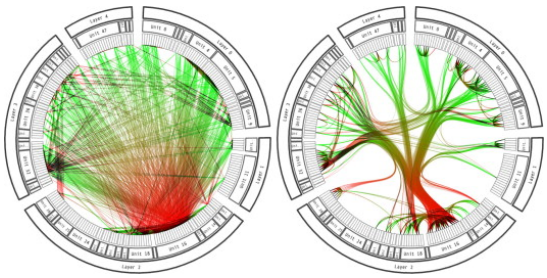


Figure 2.5: Example of Edge Bundling for a Radial chart [3]

Hierarchical layouts are primarily used for trees. They take the shape of a circular dendrogram chart (see Figure 2.6). In this layout, it starts by initially putting the root node in the centre of all the circles. Each circle encompassing the central root node hosts the child nodes of the root nodes. Which circle they are placed on corresponds to their depth from the root node. This layout is well suited for directed graphs and tree like structures

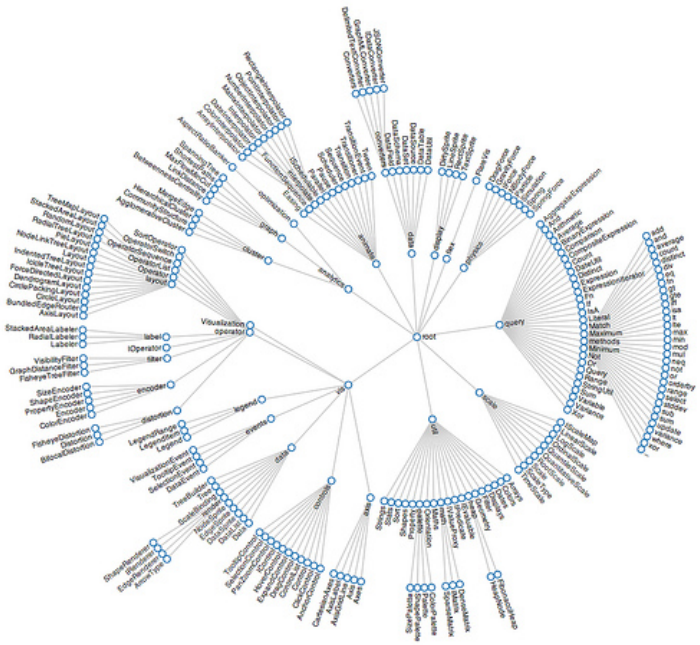


Figure 2.6: Hierarchical Radial visualisation [4]

Implementation of a hierarchical layout has been shown to work where it demonstrates its ability to handle large data sets while also containing it's graph structure and edge attributes, see Wills [21]. In this paper these topics are discussed and provide purposes of its application when trying to identify abnormalities within network data sets. This stays continuous even with increased scale of the data set into the thousands of vertices.

2.2 Filtering

When it comes to network graphs, filtering is the process of removing noise in graph data. The aim of filtering is to remove nodes of less importance to the graph. This is an important step as not only do you need to define how you proceed with removing nodes, but one must be careful, as removing nodes is removing information from the network.

There are a variety of filtering techniques in which their procedures prioritize separate issues. Coreness and centrality are the two main procedures that will be discussed. Coreness refers to the measure of identifying tightly interlinked groups within a network while centrality refers to the importance of each of the individual nodes of the network data set.

2.2.1 K-Cores

A k-core of a graph is a maximal sub graph of nodes, all of which are connected to at least k other nodes within the sub graph (see Figure 2.7). Every node to be included must be checked if it has greater or the same amount of nodes desired. After removing nodes that do not satisfy this condition, repeat the process until no nodes are removed. It is used to remove isolated nodes and nodes primarily related to isolated nodes outside of core clusters. It has uses in areas such as anomaly detection and identifying influential spreaders in social networks.

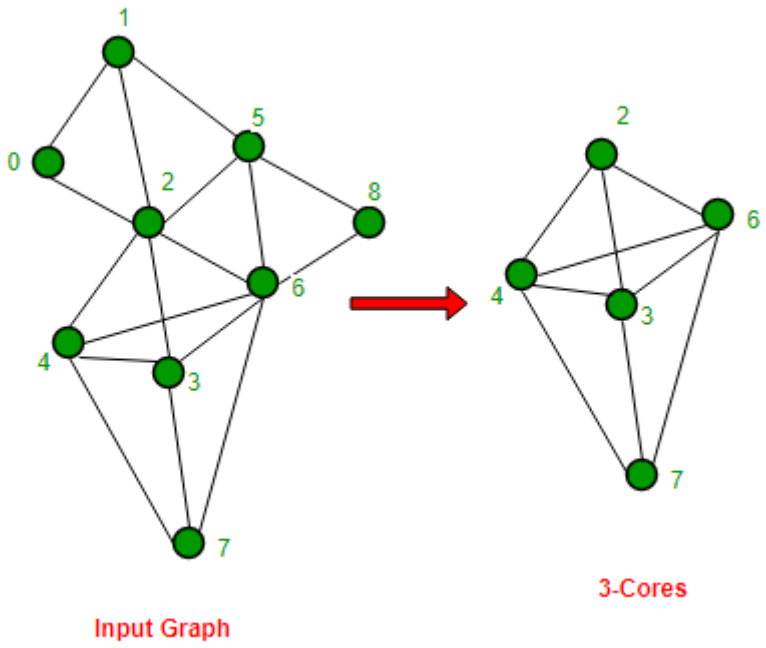


Figure 2.7: Representation of k-core value 3 to a graph, [5]

K-cores are a natural means of reducing and simplifying the structure of graphs. It is used to find and identify small interlinked clusters of nodes. Depending on the level of filtering of K-cores clusters can become isolated removing the structure in the visualisation and connectives of the original graph.

K-cores have had many applications, patterns and anomalies for real world data [22]. Their structure has also been tested by how resilient they are with core structural changes, core minimization and graph unraveling [23]. This reinforces their role as a high form of organisation within networks.

A further type of filtering is a k-shell. The definition of a k-shell is the sub graph of vertices within the k-core of size n, but not included in the sub graph of k-cores of size n+1. A 0-shell would represent only the isolated nodes and a 1-shell only represent dyads within the

graph. K-shells are used as their decomposition has often been used for data visualisation in studying core structures of large complex data sets [24]. Within this paper, they apply a technique that creates groups based on how far they are from the core creating a hierarchy of layers. Montresor [6] demonstrated the decomposition of k-cores into its k-shells (see Figure 2.8). From this figure it the k-shell sub graphs are visible for each value of the coreness.

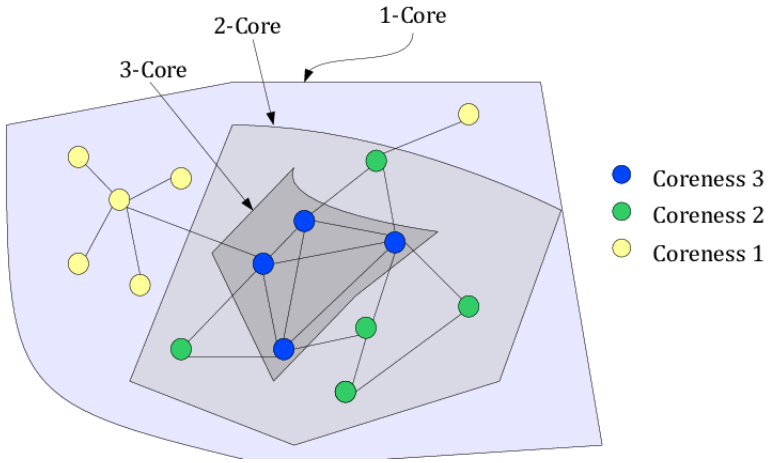


Figure 2.8: Decomposition of k-cores [6]

2.2.2 Centrality

Centrality is used for identifying important nodes within graphs. It refers to how central a node is within a graph. A node of high centrality is decided by how the centrality value is defined. The three primary definitions of centrality commonly used and that will be explored are Degree Centrality, Closeness Centrality and Betweenness Centrality [25]. There are many more that are less common such as Katz Centrality and Sub graph Centrality that weren't used due to their speciality, as well as many variations of Betweeness Centrality with Shortest Path Centrality being the form that will be focused on. [26]. Nodes centrality is often normalised to ensure normality in how it looks, reads and can be utilised (see Figure 2.9).

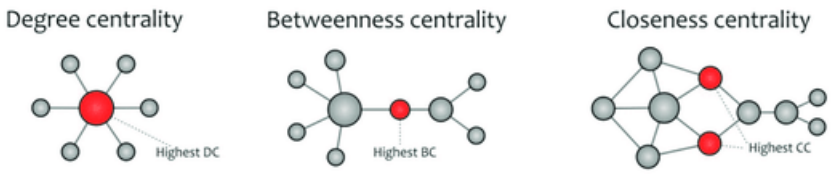


Figure 2.9: Centrality measures [7]

Strategies of implementing centrality values for filtering have been broadly implemented. This is due to node importance defined by the centrality measure being a optimal metric of filtering as it suggests nodes of lower centrality are of less importance to the network structure. Edge [27], demonstrated edge cutting strategies tied with machine learning and language processing techniques to large data sets to test their performance metrics. These node and edge metrics give comparison to each edge cutting strategy as well as their strengths, weaknesses and what data set characteristics it its best applied to.

Degree Centrality

Degree centrality refers to the amount of edges each vertex has that's connected to other vertices. This correlates to the more edges a vertex has, the higher its centrality value. It allows nodes with the most connections to be classified as more central. The effectiveness of this measure is tied to the issue that nodes with high degrees also share high centrality with other measures [28]. The process has the distinct advantage of being computationally cheap and fast. While degree centrality is a good measure for the total connections a node has it does not indicate whether the node is central to the rest of the graph.

Closeness Centrality

Closeness centrality refers to the average path length to other nodes in the network where lower values are regarded as higher centrality [29]. It is calculated by getting the inverse of the sum of all shortest paths. This is then commonly normalised to get each value. Within data sets where the network is not fully connected the centrality is limited to the largest sub graph of connected nodes as otherwise every node would a centrality value of infinite [30]. In practice the closeness centrality is used to estimate the speed at which information would travel from one node to other nodes. Closeness is used when you wish to find the most efficient distributor within a network.

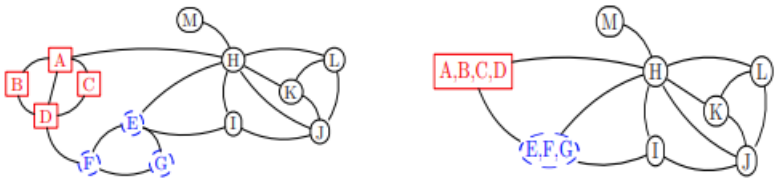
Betweenness Centrality

Betweenness centrality refers to the measure the importance of a node by the tally of times the node is used within the shortest path for all pairs of nodes in the graph. When there are multiple shortest paths of the same length all paths are applied but the value to add to each node is divided by the total number of shortest paths. Nodes with high centrality are the connecting nodes between large clusters of nodes and hence have are crucial to the structure of the graph. In practical situations few nodes with high centrality can be signs of an over dependence on the nodes and would require observation. This definition is shortest path betweenness and there exists other definitions of betweenness centrality such as communicability betweenness centrality and random walk betweenness centrality. Communicability betweenness centrality involves instead of taking just the shortest path to

take multiple paths through nodes [31]. Random walk betweenness centrality refers to taking multiple random paths between nodes [32].

2.3 Abstraction

Abstraction is used to transform a large visualisation of a network graph into a smaller visualisation by reducing it to a set of its essential elements to be more useful for visualising, analysis and understanding. Zhou [8] demonstrated three techniques for abstraction: pruning, partition and replacing. Pruning is refers to removing nodes and edges that are of low importance within the network. Partitioning is the act of partitioning a network into smaller sub graphs by removing connecting edges. This involves identifying long connections between nodes which are connecting the cluster together within the network. Applying partitioning creates isolated sub networks that are easier explore. Replacing is the process of grouping similar nodes and edges together and replacing with a single representation. This could include group of interconnected nodes with a glyph or parallel edges with a singular edge (see Figure 2.10).



(a) Original graph (b) Result graph

Figure 2.10: Abstraction through Replacing [8]

Pruning will not be pursued as it is too similar and is a weaker form of filtering than centrality, which was discussed in 2.2.2. Commonly, partitioning will not be used as it does not substantially aid the project, as high k value k-cores supply a similar purpose. Also, partitioning does not comply with the project goals as it greatly alters the data set from its original form in terms of isolating clusters removing core functionality of the data sets in terms of connectives. The replacing process is subjective but when applied correctly allows high levels of visual clarity without loss of the graph structure. Due to this, replacing will be the form of abstraction that will be implemented. This will be pursued with the use of cliques, which are highly connected node clusters.

2.3.1 Cliques

A clique is, at the most general level [33](p162), "is a sub-set of a network in which the actors are more closely and intensely tied to one another than they are to other members of the network". The smallest possible clique consists of two vertices, named a dyad, but for this dissertation we will be looking at triads, a clique of three, and up.

The formal definition of a clique in network analysis involves a clique to be the [33](p162) "the maximum number of actors who have all possible ties present among themselves". This type of clique is also be called a "maximum complete sub graph". This definition of a clique is what will be used moving forward (see Figure2.11).

Figure 2.11: Simple representation for a clique of 3 nodes within a graph

The other definition of a clique is an n-clique. This is used as the formal definition can be too rigid for some tasks and relaxing the necessities to be more helpful and general for certain tasks. A n-clique is defined as when every vertex within the clique is connected to every other vertex within the range of n edges. Most commonly a value of 2 is used for n and can be referred to as a "friend of a friend". By this principle a clique would be a 1-clique under the definition of n-clique (see Figure 2.12).

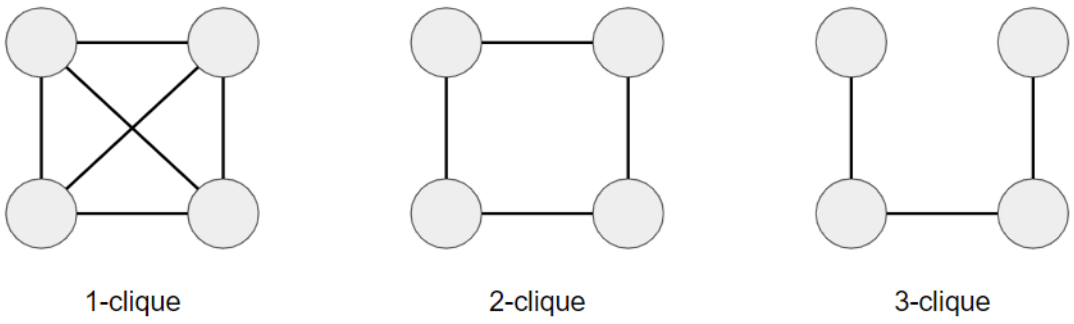
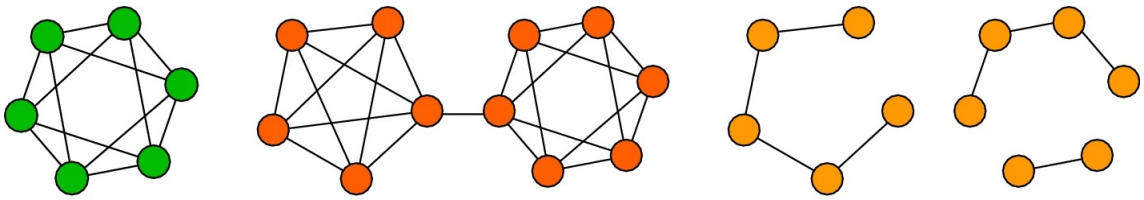


Figure 2.12: Example of different n-clique graphs

Some sources [34] include in the definition of a n-clique, being maximal and require every vertex that can be included to be included. These stricter definitions are used when there is the need to define a n-club and a n-clan. Within these definitions, a n-club is a set of the vertices that induces a sub graph of diameter less than or equal to k. This means a n-clique

that does not use edges that are not within the clique. It is a stricter definition of the previously defined n-clique. A n-clan is defined as a n-clique that includes a sub graph of diameter less than or equal to n. This requires the n-clique to be maximal. This refers to that the n-clique of the graph allowing vertices not included to be a member when defining the clique, but when defining the vertices within the n-club, if they do not match then the n-clique does not satisfy the definition of a n-clan.

A stricter definition than n-clique, n-club and n-clan but not quite as strict as a clique, is that of a k-plex. A k-plex [33] is defined by being a sub graph of size n, where every vertex within itself has a direct connection to n-k vertices within the sub graph. So a sub graph of size six where every node has at least three edges within the sub graph would be a 3-plex. By this definition a clique is also a 1-plex. Conte [9] discusses k-plexs further demonstrating finding algorithms (see Figure 2.13).



(a) 2-plex, s=6 **(b)** 7-plex, s=11 **(c)** 11-plex, s=11

Figure 2.13: k-plex [9]

For n-cliques, when the n value is two, the clique would be called a 2-clique. While this is a definition, when I refer to a clique as such it will mean something else. When I am referring to a clique in such a manner I am detailing the amount of vertices within the clique. To demonstrate, when referring to a clique as a 5-clique, I am referring to a maximum complete sub graph containing five vertices.

With cliques in data visualisation there needs to be a manner of representation for these cliques. As they are not normal nodes, but still act like nodes a form of representation that is similar to the nodes yet unique, even at first glance. In a paper by Dunne [35] the use of glyphs is demonstrated. The use of custom shapes to represent the cliques depending on their size to easily differentiate the different sized cliques intuitly.

2.4 Brushing

Brushing is a technique found within data visualisation in which it enables interactivity when selecting vertices and edges from a data set. By brushing a vertex it can enable multiple functions including but not limited to highlighting, deleting and masking. Through brushing it is possible to select and interact with an individual vertex, multiple vertices or an entire subset of the original data set. Brushing was formerly introduced to data visualisation by

Becker and Cleveland [11] in which they implemented a primitive brushing system incorporating masking and highlighting. It has been greatly expanded upon since with interactivity, optimal motion techniques and compound brushing. Ward [10] displays how brushing can be applied within a visualisation of a data set along with a multifaceted view of how the brushing applies with different idiom views to display the changes (see Figure 2.14).

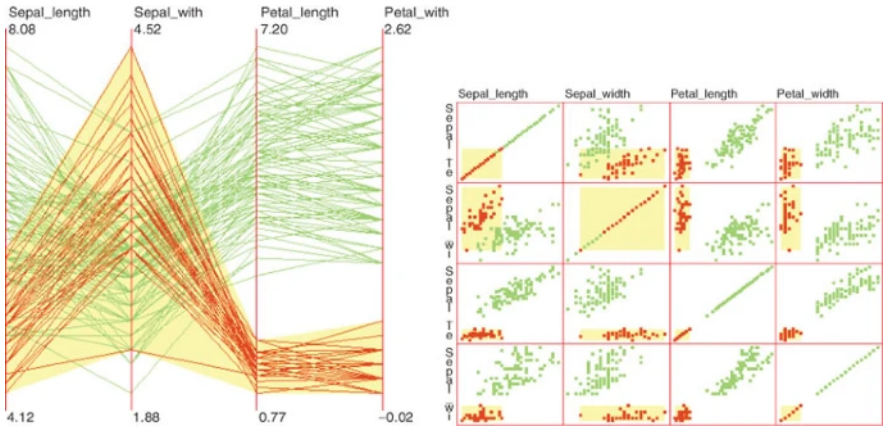


Figure 2.14: Brushing implementation [10]

Optimal motion techniques were demonstrated by Bartram [36], in which they took advantage of the strong grouping effect. Doing so they tested different methods of motion for effectiveness. with these being linear motions, circular motions and motions of expansion and contraction.

Compound brushing being displayed by Chen [37] demonstrates a nested graph formalisation to model different brushing processes and techniques as brushing higraphs [38], higraphs being structures extending graphs by enabling hierarchy of nodes. These higraphs use the data set encoding channels as atomic nodes. This compound brushing is useful in facilitating implementation of brushing techniques and to analyze, classify and organise the different brushing techniques.

2.5 Summary

This chapter has described the main techniques that will be built off of and implemented into the project for network visualisation. I will describe the procedure in doing so in the following chapters discussing design and implementation.

3 Design

3.1 System Overview

The system consists of multiple layers designed in a way to create an application to display idioms of network data sets in real-time with multiple forms of reduction of on-screen clutter. Each component is based on this design and optimised and justified towards providing an intuitive, interactive and adaptive display. Core areas of the system to be discussed are:

- User Interface- The interface which the user will be using when interacting with the tool. What is entailed along with the reasoning behind the design of the system including the idiom representation, sub chart display, interactive controls and other notable design decisions.
- Real-time- Discussion of real-time and the advantages and disadvantages of utilizing it. Demonstration of how real-time enables higher levels of interactivity along with allowing higher levels of information to be retrieved by the user from the tool. Also, real-time is a necessity to fully show the adaptability of the program and certain idioms such as the force directed layout. Also demonstrates what features become non-optimal in conjunction with a real-time application.
- Filtering Techniques- A look into the the filtering techniques discussed in 2.2. This will weight the advantages and disadvantages within the application, alone and in conjunction with other features. This will discuss a mixture of testing a range of abilities but also guaranteeing it has an acceptable run time and scaleability.
- Abstraction Techniques- A look into the the abstraction techniques discussed in 2.3. Weighting the advantages and disadvantages to the approaches and formulating strategies for representation tied to scaleability and computational time, isolated and in conjunction to other facets of the application.
- Filtering and Abstraction Tuning- Implementing the tools for deciding the level of filtering and abstraction desired. This will explain the approach to the feature and an

explanation for its inclusion.

- Brushing- Discussion of brushing and the desired traits that are best suited by enabling the tool. Will comment about what aspect of brushing will be selected and what areas of the project brushing will enlighten and enable better discovery with.
- Encoding Channels- Discussion of the encoding channels used within the project and the reasoning and justification for the assignment of channels towards each task. This will discuss the methodology for assigning and reducing overlap for visual clarity across the project.
- Pre-processing- The pre-processing steps will be described and the strategies involved within it are discussed. This section will be about data preparation before using the tool to get the most out of the application along with what pre-processing will be explored for use in the visualisation.

3.2 User Interface

The user interface is arguably the most vital component, as all aspects of the project culminate here. The visualisation should include all functioning features in a neat presentation. Doing so requires a planned presentation with space cordoned off for each feature.

The idioms are the core of the project and so need a large amount of space to feature. The use of the entire screen is to be given to the chosen idiom where the entire chart can be visualised with a sliding panel for additional control. The idiom should scale depending on how far the sliding door is enlarged showing adaptability within its geometric limitations.

The sliding panel should be an intractable element that is used to get a full view of functions applicable towards the data set. This allows the ability to alter the view of the panel and its options as well as the idiom displayed depending on clarity needed.

Within the control panel there are multiple options employable. One these is brushing, which is an interactive manner in which to interact with a data visualisation manually. It is an important tool for control of data visualisations. Brushing is discussed in section 2.4 and in terms of design in section 3.8. Control for filtering and abstraction through the control panel are present also. The degree to which it is applied along with the selection of techniques which is further discussed in section 3.7.

The other use of the control panel is the idiom selection. Within the control panel there contains a display of all supported charts presentable in a multi faceted manner. These consist of transformed smaller and less detailed charts compared to their primary counter

parts. These smaller charts within the multifaceted display will adapt to changes in the structure in real-time when reduction is applied or removed. This enables the ability to utilize both selection and observation within the control panel.

Across all these features the ability for interactivity is vital for optimal performance. If there are pauses between functions comparisons would have drastically reduced efficiency and counteract the appeal of many systems that rely upon it for function. This promotes real-time performance as a important value to strive for.

3.3 Real-Time

Real-time within computing refers to the ability to process data in such a timely manner that the feedback is returned near instantly. Real-time has advantages and disadvantages that are to be weighed when incorporating it. The advantages of achieving such a performance is increased interactivity that heightens the user experience and better demonstrates how changes apply across all idioms. The adaptability is also clearly shown with real-time with adapting to filtering and abstraction globally alongside certain idioms adapting to the change in structure.

The disadvantages of applying real-time are tied to having a level of minimum performance that is required for acceptable functioning. This is very apparent when discussing which reduction techniques to apply. Some techniques require large computational time to complete. This become more apparent when discussing the often unknown data set size. Larger data sets can increase run time of functions exponentially and so is a determining factor in what reduction techniques to deploy.

3.4 Filtering Techniques

The filtering techniques discussed in 2.2 went in depth into the many ways to apply filtering based on centrality and coreness of a graph. With the many variations, deciding which to implement is vital and will greatly impact the product as a whole. The discussion will be split into discussing the coreness and the centrality. Within these talking points explanations into deciding the best way to demonstrate and explore both these ideas will be projected. This will delve into the methods that best enhance the overall experience within the limitations applied to the project.

Corness

For filtering based on coreness there were two methodologies discussed within 2.2.1. These consist of k-cores and k-shells. The use of the k-core is signaling where clusters are within

the data set. This can be tuned to the degree of coreness desired. The advantages of k-cores is that they filter isolated nodes along with nodes that are not structurally core to the network. Higher levels of coreness create large interconnected sub graphs. This helps demonstrate the core structure of the data set. K-shells do a similar task but instead of demonstrating the coreness based on the k value, they demonstrate the hierarchy within the network. This has the advantage of clearly demonstrating which nodes are of high importance for the clusters within the data set. They follow a similar procedure within implementation with an extra step for k-shells to calculate the nodes within the k-core but not the k-shell.

In terms of implementing the two techniques, both have similar computational times and can be calculated in the background within real-time and have a linear scalability with the data set size used for the visualisation. How they interact with the other tools within the system, the idioms for displaying the information and their readability should be discussed in their implementations.

K-cores is a simple implementation tied to a similarly simple display. With the applied filtering the nodes no longer satisfying the condition are removed. It can easily be displayed across all charts and does not need to use additional encoding channels. This means it does not lead to a cluttered display and application can be easily seen in the sub charts within the multifaceted display.

There are two ways to represent k-shells: each layer displayed in solitary, and displaying all at once with an encoding channel to highlight level of coreness within each k-shell. With displaying isolated k-shells providing very little for understanding of the network and removing all structure it will be ignored and instead will look at displaying all layers simultaneously. K-shells are a similarly simple implementation to k-cores though the display of this filtering technique becomes complicated due to a need of additional encoding channel. The need to take an encoding channel to display the hierarchy not only adds constraints on other tools, the encoding channel may not be universally usable across all idioms. This would lead to other features removal for a secondary hierarchical form of coreness filtering.

Due to these factors replacing was pursued as a technique for abstraction. However, it is necessary to further explore the forms of replacement to employ.

Centrality

Filtering based on centrality has three techniques discussed within 2.2.2. These are degree centrality, closeness centrality and betweenness centrality. These techniques are based around keeping nodes of high centrality and removing nodes of low centrality. The centrality of vertices is the measure of importance of each node in the network. While all techniques

aim to achieve the same goal of determining the vertices centrality, they employ different strategies that come with unique advantages and disadvantages within this tool.

The centrality techniques implemented are diverse. To describe at a high level degree centrality is based on the amount of edges for each vertex, closeness is the average length of each node from other nodes within the network and betweenness centrality is based on how many shortest paths between nodes pass through the vertex.

Degree implementation is a simple procedure in terms of complexity and computational time. As it reads the edge lists and increments the vertices at either end of each edge it is simple to understand the process and leads to it only requiring a single read through an edge list. This leads to the scaleability growth being linear. These features allow for it to be very adaptive across data sets. It does not encompass additional encoding channels and changes are easily demonstrated across all idioms as it consists of removal and reintroduction of vertices and their edges. These features promote the technique as it scales well, does not clutter the charts as well as changes can be seen globally easily.

Closeness centrality is a more complex procedure. The process involves finding the sum of the shortest path for every vertex to every other vertex within the graph. This requires a path finding algorithm to be applied multiple times for every node in the data set. It involves greater time complexity when calculating and so becomes unaffordable within larger data sets. Aside from this difficulty, the method is sufficient for implementing with other tools. It does not use additional encoding channels and is applicable to each idiom along with scaling well based on each chart's assigned area.

Betweenness centrality is similar to the closeness centrality in its strengths and issues. The method does not use additional encoding channels and does not clutter the charts used. The issue with betweenness centrality is it requires finding the shortest path for each node to every other node within the network. This, once again uses a path finding algorithm to be applied multiple times. Each time a path uses a node it will increment the node. If the path contains multiple shortest path it will increment each node for each path by the increment value divided by the amount of paths used. This all together is very computationally demanding and takes a lot of computational time. Similarly to closeness centrality, betweenness centrality scales poorly with larger data sets and is unaffordable.

Due to the limitations computationally with closeness centrality and betweenness centrality I had to make a decision if I was to pursue them. This was an issue as employing them would drastically reduce the amount of data sets I could use within this project. I decided against using the techniques for this very reason. I would pursue degree centrality as there were no major downsides and would allow me to run the program with no additional restrictions on data set and operate within real-time with no major issues.

3.5 Abstraction Techniques

Abstraction was discussed briefly in 2.3. Here the methods to employ are discussed and three solutions are proposed. These are pruning, partition and replacing. Pruning is the removal of edges that are of low importance. This is close to centrality filtering techniques and so will be compared to these. The implementation of pruning is not so elegant and would converge to close to an already used measure and so the method is not pursued. Removal is similar to the coreness filtering measures. Alongside having a similar adoption within the the methods, the removal of core graph structure was not a desired trait so this too is not pursued. The replacing methodology is an approach that is notably unique from previous techniques to be used in the work. It would not remove network structure and would only group entities. Due to these factors replacing was pursued for abstraction but the forms of replacement to employ were still necessary to explore.

Within replacing, there are multiple attributes to abstract. The two most obvious are the edges and the nodes. This would entail replacing nodes and/or edges that serve similar purposes. Determining what entails a similar purpose is the key area to discuss for abstraction. This section can be broken into what aspects of the graph to abstract and how to determine what portion is to be abstracted. A limitation on the representation of the cluster to present would also be established. Thus, there must be three nodes within the replacement to make sure it is worth abstracting. Any connection is a dyad, a clique of two nodes, and if this condition is not stated would replace all non-isolated nodes.

When deciding what to abstract through replacement, large clusters of interconnected nodes are an area of interest. These would need to be clearly defined so as to not create confusion as to what occurs when changes through abstraction are applied to the visualisation. The best metric for doing so are cliques. The definition of cliques is not consistent and so defining a clique would be another layer to this task. As to the other forms of replacement that were proposed, there was edge replacement when edges that served similar connections between nodes and node clusters could be replaced with an edge to represent both. This technique was not pursued due to not being able to properly represent the abstracted edges within the visualisation and this causes misconceptions in the structure of the data set.

The clique definitions that were defined within the scopes of the project are a standard clique, an n-clique along with n-clans, n-clubs and k-plexs which served intermediate forms of abstraction between the strictness of cliques and the loose definitions present in n-cliques. Each of these supported separate advantages and disadvantages. Each would have data sets that they would specialise towards but discussion as to which would best serve the generalist nature of the input data has been discussed.

A clique, by the strict definition, is a maximally connected sub graph. This entails that every

node within a sub graph of a graph is connected to every other node of that sub graph. This definition guarantees that all nodes inside the clique are connected so edges that enter the clique have at most one connection within the clique before leaving. This allows a major aspect of the structure of the original network to be present. This definition guarantees that the lowest level of reduction is applied to the network by abstraction through the clique definitions.

An n-clique tends to be a looser form of a general clique. An n-clique is a sub group of nodes within a network which are within n edges away from another node contained in the sub group. This means a 1-clique is the same as a clique by the strict definition. A further restriction will be applied to the definition for implementation of the sub group of nodes would need to be maximal. This is to enable the discussion of n-clubs and n-clans. Commonly when implemented, n-cliques are of value two. Higher values on non-sparse data sets can end up over-clustering the data set and creating too large clusters of cliques. This leads to a loose clique as well as the ability to use nodes outside the sub graph to form the connections. Utilising such an approach would produce the most cliques.

N-clubs and n-clans are very similar so will be tied together. An n-club is an n-clique, except that it does not allow connections to nodes outside the sub graph of its set. An n-clan is even stricter and requires the n-club to be the same as the n-clique it is derived of. The use of these definitions ensures that outside connections aren't used and ensure that the clique representation is based solely on the nodes within the cluster. It will be looser than a traditional clique but would guarantee that every node is reachable within n edges inside the representation of the cluster.

Last of the clique representations are k-plexs. These are cliques where each node within the clique has the amount of edges equal to the total amount of nodes within the clique minus the k value, within the subset of nodes in the clique. A standard clique would be a 1-plex due to all nodes within a clique connecting to all other nodes within the subset. K-plexs are a lenient version a clique while being more strict than all other implementations.

Completing the overview of all possible definitions of replacements for the clusters, a decision needed to be made to decide which to use. In regards to n-cliques their definition is too loose and would not fully represent the clusters as would encompass nodes not fully integrated. N-clubs and n-clans while more strict, the techniques fail to demonstrate the highly connected clusters from the lesser connected. Alongside that, they can still have the opportunity to represent nodes not fully within clusters when too high n value. The last two clique definitions after the elimination of n-cliques, n-clans and n-clubs are cliques and k-plexs. Both have justifications for being used depending on the data set so formulating a decision on which to pursue was difficult. After implementing the clique definition across multiple data sets and seeing it drastically reduces the amount of nodes while maintaining

the structure, it was decided to use the formal definition of clique as the form of replacement.

3.6 Clique Representation

Having developed a definition of what to use for abstraction, a means to represent the clique was necessary seeing as visualising it as a node would not demonstrate the combined collection of nodes and edges within the clique with the weight they carry. To counteract this, the use of glyphs was decided upon. A glyph is a hierarchical symbol and so suits the necessary implementation of representing the clique. These are necessary as cliques can be of different sizes and so need to be distinguishable from each other.

Implementing the glyph involved another decision. The glyph needed to take up a new encoding channel and deciding which is an important discussion. The optimal encoding channels available to glyphs are size, mark, brightness and colour. These are channels that are explored. The encoding channels used globally across the project are further expanded upon in detail within section 3.9.

Size as an encoding channel was an intuitive encoding channel for glyphs to gravitate to. The size would represent the amount of nodes a glyph would represent. There was initially a problem with large cliques, as these were displayed as too large and took up too much screen space, but when normalisation was applied this became much less of an issue with a predetermined max limit to the glyph size.

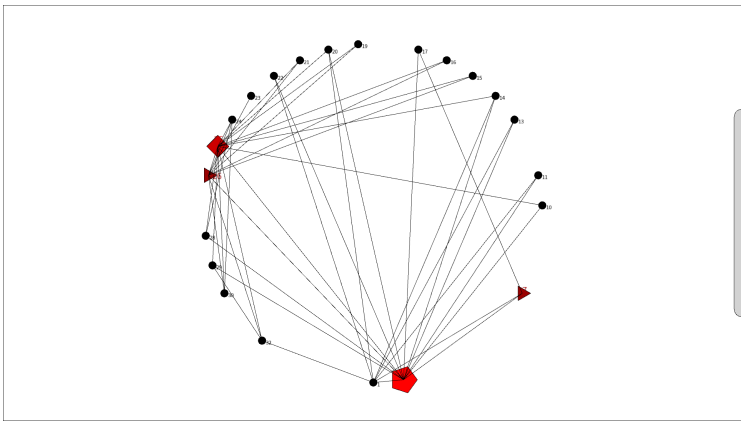


Figure 3.1: Glyph Implementation

The encoding channel of marks at first glance is very optimistic for implementation. The process of utilizing marks is implemented by a clique being represented as a custom glyph. Initial designs had the glyph as a custom polygon shape with the edges indicating the amount of nodes within the clique. This implementation can be seen in Figure 3.1. This worked until data sets which included a large amount of nodes were causing problems relating to memory. Due to this, custom glyphs were shelved. The reasoning being that a

core motivation within the project is to be able to intake a range of data. Implementing custom shape glyphs would greatly reduce the possible data sets that the system could represent. Instead the solution of using a square as a glyph to represent the cliques was implemented. The unique shape within the system made the cliques easily differentiable from the general nodes but due to all cliques utilising the same form of mark in a square a different form of classifying them by the nodes represented was necessary.

The last encoding channel suggested was the simultaneous use of both colour and brightness. Initially, I was reluctant to use this channel due to conflicting with the brushing implementation in 3.8. However when tied with other encoding channels it would become distinct enough to differentiate itself. The colour would be used to notify what is a clique and to make it easily distinguishable from afar. Then I would apply brightness through a form of normalisation where the more nodes within a clique would have lower luminance intensity. This allowed the ability to use the brightness of the colour to identify the quantity of nodes within the clique.

These features were then evaluated for inclusion. Despite not using complex marks through glyphs, both size, marks, brightness and colour were utilised at varying levels of degree. The mark encoding channel was used to employ squares as glyphs to represent nodes and both size and colour was utilised for visual clarity for nodes. To be distinguishable from afar both brightness, when applied to the colour of the glyph, and and size of the marks were used to indicate the quantity of nodes inside the glyph.

3.7 Filtering and Abstraction tuning

The filtering and abstraction were controlled by sliders within the control panel. This was to allow the user to, at will and within real-time, control the level of reduction applied to the selected data set. Within the control panel the upper and lower bound is shown along with the degree the user selected. This is so the user can see the exact level applied and be able to interact to get the results they desire. The range of reduction are calculated within the program in the background.

The reasoning for this implementation was the need to interactively, and within real-time, adjust the levels of reduction. This allows the user to see the changes applied globally across all idioms in order to quickly see the evolution of all charts with the levels of filtering or abstraction.

Alongside the sliders are radio buttons. This is due to the techniques not being applicable concurrently. Thus, applying filtering methods with different aims will contradict each other based on approach. An extreme example of why two systems can't be applied concurrently would be applying betweenness centrality and coreness concurrently. Centrality will try to

keep core links between clusters while corness would remove nodes that aren't highly interconnected. This could lead to coreness removing the links that betweenness centrality would value highest making its filtering aimless.

While abstraction and filtering can be applied concurrently, it is not implemented due to constraints in real-time performance. The reason why there are limits to the reduction techniques is due to the need to re-evaluate calculations when altering another. This is due to abstraction being done through pre-processing and so has not been optimised to real time. When filtering is applied to the data set the network changes. This means the abstraction applied is no longer relevant as the structure of the graph has changed. Due to this it would have to re-calculate the cliques which is not feasible within real time. This is why filtering and abstraction are not capable of being applicable concurrently.

3.8 Brushing

Brushing is a vital tool. It is as a form of interaction with the visualisation and consists of manually selecting nodes and edges within the charts for a range of functions. When creating the brushing function within the tool, filtering and highlighting are the traits that were decided upon to use.

The reason for adding filtering was its ability to manually remove nodes and to see the consequences across the idioms of how these changes alter the structure of the network as well as how the filtering is affected. These changes apply globally and will remain until the filter is reset, which is done through a selectable button on the control panel. The benefits of implementing this technique is to allow more control for filtering that surpasses the techniques already in place. This can be used in a highly understood data set where the user would like to alter the implementation to see what would happen in the case when a node fails. This application works with filtering. Applying filtering through brushing causes the application to recalculate the k-cores and degree centrality. This is done within real-time under the hood. It is implemented in order to best demonstrate the effect nodes have on the graph and its coreness and centrality. This brushing filtering does not cooperate with abstraction.

The other form of brushing that is displayed is highlighting. Within this mode, selecting nodes will highlight the node as well as the links across all idioms. This is an excellent tool in more dense data sets as it allows the user to see connections across all charts. Also in dense data sets, the tool demonstrates benefits when connections are not always visible. It is also of benefit when specific idioms displayed become cluttered by mid to large sized data sets. The choice of highlighting is very useful to tasks where identification is crucial and where one wants to specify where nodes are represented across all idioms. This feature also

contains a reset button to undo highlighting for all nodes when a user has completed their desired process.

3.9 Encoding Channels

An encoding channel are defined as the means that attributes can be perceived by a user through a visualisation. These are within every visualisation and consist of how the charts translates data from a collection to a visualisation.

The encoding channels are limited and so need a detailed account of their specialties and distribution to functions across the application. This will be discussed in detail in detail in terms of design decisions and justifications for each choice. There will be discussion across eight different encoding channels in which the uses, if used, will be explained.

The primary encoding channel of all forms of visualisation is position. Every aspect of the visualisation uses position in some manner. While vital it is also intuitive so I will not expand upon its uses here.

Another encoding channel is mark. This is the use of geometric elements to display data. They utilize these shapes to create distinction within the visualisation and when implemented well are easily distinguishable between each other. Marks are utilised in demonstrating the nodes and the cliques. Across all idioms bar the adjacency matrix, the individual nodes are displayed as circles while the cliques are displayed as squares. This allows the two unique structures to be distinguishable at a glance. For the adjacency matrix, the nodes are demonstrated as squares and cliques are not demonstrated in this medium.

The colour and brightness encoding channels are similar when discussing implementation so I will be grouping them for this discussion. Colour refers to the saturation of the data while brightness refers to the intensity of the luminosity. Colour is used for the individual nodes, the highlighted nodes and the cliques. Within the application the nodes are black, highlighted nodes are orange and the cliques are different intensities of red. The shades of red are examples of brightness as an encoding channel within the application. Alongside these features the edges also share the colour variables within their encoding.

Motion is the last encoding channel implemented. It is information shown through change in position. This is shown primarily through the force directed layout where the system will adapt to removal or reintroduction of nodes and edges where the layout, with motion, will redistribute itself. Motion can also be seen when using the sliding panel where the idiom will readjust to fit within the space provided.

Some encoding channels were not utilised such as size, orientation and texture for various reasons. Size was not incorporated due to unknown max sizes within the selected data set.

Within visualisation size can become overbearing if it crowds out other elements displayed. Orientation was not used due not being easily identifiable due to the only marks utilised being squares and circles which would are either unnoticeable or too minor to easily identify. Texture was also not used as loading multiple textures for larger data sets being very demanding alongside would make the visualisation extremely crowded and lack a homogeneous tone.

3.10 Pre-processing

Pre-processing was an important step in procuring the cliques which were a vital aspect of the visualisation. This process was split into two steps, detecting the cliques and filtering the cliques. Detecting the cliques was the process of identifying all cliques within the data set and storing them. Filtering the cliques was removing redundant and low priority cliques so only the core clusters were represented.

3.10.1 Clique Detection

Clique detection was employed as a collection of every clique within the network will provide great flexibility in representation. The process involves using an established algorithm and expanding on it for every value possible for clique size within the network. Each clique within each clique size is then saved and stored to a file within a directory for future work. This allows all cliques to be usable and stored for future processes as well as only needing to run the application once.

3.10.2 Clique Filtering

Clique filtering comes with a range of tasks. When running the clique detection script, most data sets used create more cliques than nodes. This is due to all cliques being represented including sub-cliques of cliques along with the sub-cliques of those sub-cliques. Every clique of k size will have k-1 cliques of k-1 size. There are also cliques that share nodes with other established cliques. This leads to an excessive amount of cliques present within the application. The core of the problem is how do you represent two cliques that share the same node or nodes? The short answer is we are not going to and instead filter cliques in which this situation occurs. This will be done through defining what we value in a clique and then filtering based on how each clique satisfies the conditions brought forward.

To reduce the amount of cliques present there were two steps within the script. Firstly the removal of all cliques that share the same nodes with a larger clique and secondly, define a method to prioritize cliques by an importance value and remove cliques that share nodes with a more important clique based on the importance metric.

Removal of cliques of smaller size is based on two principles. The main principle is due to the fact that removing sub-cliques of a clique will always remove the less important clique between the two as the sub-clique will always be missing at least one node, hence is will be missing a value of importance to utilize. There is less obvious for nodes that are not a sub-clique of a clique as it is possible for the clique to be valued as higher importance. The reasoning not to compute the difference here is a mixture of factors. The first being this is a niche case and will take a large amount of computational time to first check if it is a sub-clique as well as the time to calculate the importance values of the cliques. Secondly, the reduction of clutter on screen is a universal metric and is not as universally applicable as further reduction by replacement in which a larger cluster is reduced to a glyph, even if it is not as valued as the smaller cluster.

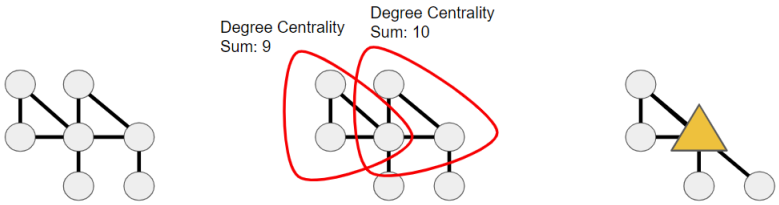


Figure 3.2: Clique Priority System

When filtering the cliques of the same size there needs to be a methodology. The implementation of this is done through filtering based of the sum of node centralities within a clique with the removal of the lesser clique. The three centrality values applicable to this were degree centrality, betweenness centrality and closeness centrality. These are the three techniques to be applied to the abstraction process and used to determine clique priority in the filtering process. The high level process can be seen with Figure 3.2.

4 Implementation

4.1 Tools

Before implementation, the decision about which tools to use is core for the project. After testing a range of technologies, in terms of purpose, strengths and weaknesses, two core technologies were settled upon: Java is used for visualisation and Python for pre-processing. These technologies have library support and a degree of specialisation that are well suited for my project. Other technologies were investigated, including C++ and Python for visualisation along with the associated libraries as well as Java for pre-processing. However, due to implementation limitations not aligning with the project motivations, these were not pursued.

4.1.1 Java

Java is used within this project primarily for its use of the processing library; a graphing library that utilises a minimal environment to an IDE through a sketchbook. It is an intuitive library which allows for quick implementation of processes. OpenGL is used in order to utilize the speed of an OpenGL graphics card expanding the the potential for drawing by rendering geometry differently.

4.1.2 Python

Python is used for its optimisations for data science. It is employed for the pre-processing of the project. Its strengths are its intuitive framework, computational speed and wide range of library support. Focusing on the libraries, there are a range of data and network oriented libraries to choose from that are suited towards the motivations of this project.

The core libraries used within the project are: pandas, a specialised library for data manipulation and analysis; NumPy, a library specialised for adding support through multi dimensional arrays and matrices; and NetworkX, a library for studying graphs and networks specifically. All three of these are employed throughout the project.

4.2 Pre-processing

The pre-processing is completed in two steps: clique detection and clique filtering are applied before rendering the visualisation. The design choices for implementation are detailed in section 3.10. These techniques were implemented using Python due to its efficient processing speeds, library support and file management systems.

Clique Detection

Clique detection uses an established recursive clique finding algorithm. The algorithm takes the desired clique size target and calculates all cliques within the data set of those dimensions. This is repeated for every value of k from three until the value where the function returns no clique values. The algorithm works by checking all vertices whose degree, amount of edges per node, is greater than the expected clique size minus one. This guarantees only nodes of relevant size are checked. The process starts by going through each node recursively and adding later nodes to the list in a similar manner. This will continue until the list is of the same length as the desired clique size. If all the nodes within the list are interconnected to each other the clique is saved. If the list of nodes is not fully interconnected, instead it is discarded. This process will end up checking every list permutation of possible nodes and determining the possible cliques through brute force. After completion of this step, the cliques for the requested clique size are are then saved to file for use within the visualisation.

Alongside detecting cliques there is another task completed within the program. This is to find betweenness centralities and closeness centralities. This is completed using NetworkX, a graphing library, and pandas, a data manipulation library. With these tools it is possible to create a graph object by creating a pandas data frame object from the file with the original graph. With the NetworkX graph, by applying specific NetworkX function for betweenness centrality and closeness centrality, it returns a pandas data frame with each node's normalised centrality. These will be used within the clique filtering process. During this stage a text file is saved with the maximum clique value in order for the visualisation stage to not have to look for a non-existing folder.

Clique Filtering

The clique filtering stage of pre-processing requires a consistent methodology. Within the script there are multiple relevant functions. The first is to find the amount of nodes. This is a simple function whose goal is to cycle through all nodes in the data set and find the maximum node value used. After obtaining this value there are five functions used for each centrality metric and max clique size. The functions are the set up, assign node centrality values, removal of cliques smaller overlapping cliques, removal of nodes based on centrality

and saving of cliques for use in the visualisation.

The set-up consists of gathering all the previously gathered cliques from the previous script to store in a list. The application is to collect each list for each amount of nodes within the clique, and to append to another list. This will result in a list of all lists of cliques as the output. This list will be referenced multiple times during the future process and so will be referred to as the list of cliques going forward.

After the set up, each node needs to be assigned their centrality value. This consists of assigning a value to each node depending on the centrality approach. The process involves cycling through the previously calculated centrality values and assigning them to each node for each centrality approach. The process is similar but slightly unique to each method. For degree centrality the amount of edges per node is calculated by incrementing each node value for each it correlates to within the edge list. For betweenness and closeness centrality the values have been pre-calculated by networkx and are assigned to the node list centrality list directly from the networkx outputs. After this process there should be a list of the dimensions three by the amount of nodes.

The next step is removing cliques that share nodes with other cliques based on the cliques sum of centralities. This process is complex but similar across all methodologies. The process involves going through every clique of every value in succession. A boolean list for each list is made first with a false value. This is to track whether to remove a clique. It will then cycle through every clique and compare the values. Within the lists, if they contain the same node they will be flagged. This means both nodes will be cycled through and summing their nodes centrality values. After doing so the clique of less value will be flagged in the boolean array for removal from the list. The node centrality values that the values are taken from are those collected in the previous stage. After completing all cliques for the node quantity the cliques flagged are popped out of the list.

After completion of removing overlapping nodes, the next step is filtering smaller cliques that share nodes with larger cliques. The process is universal across all methodologies. Within this process the cliques with more nodes are looked at first. Here each node within a clique is flagged. After doing so the next layer of nodes is checked and any cliques that use nodes that are used by cliques of greater size are removed. Then the nodes used in unfiltered cliques are removed. This is repeated for all layers. At the end of this process there is a guarantee that no two cliques in the network share a node between them.

After completing all the prior steps the cliques are saved to file to be used in the visualisation. All the cliques are stored in files corresponding to their size all within a directory. The cycle of these five steps is performed across all methodologies and also across all levels of clique sizes.

4.3 User Interface

The user interface is a core aspect of the visualisation. It is where all the functions and tools are demonstrated. The core of the user interface is a wide area in which to show the selected idiom with a sliding panel that shows the control panel for all chart interactions.

Within this space the selected idiom is represented. The boundary is set from the height of the screen and the left border to the sliding panel for the control panel. This guarantees the greatest possible space for the idiom. The control panel allows the user control on how much they wish to show at any given time. The different graphs can interact differently to the space. The force directed layout will interact with the reduced space of the sliding panel opening by getting moved into the empty left side of the screen as the wall pushes the nodes to the left side of the screen. The other idioms act more rigidly with centering in the given space and when the horizontal space becomes too small, reducing scale to fit within the new constraints. The control panel implemented can be seen in figure 4.1.

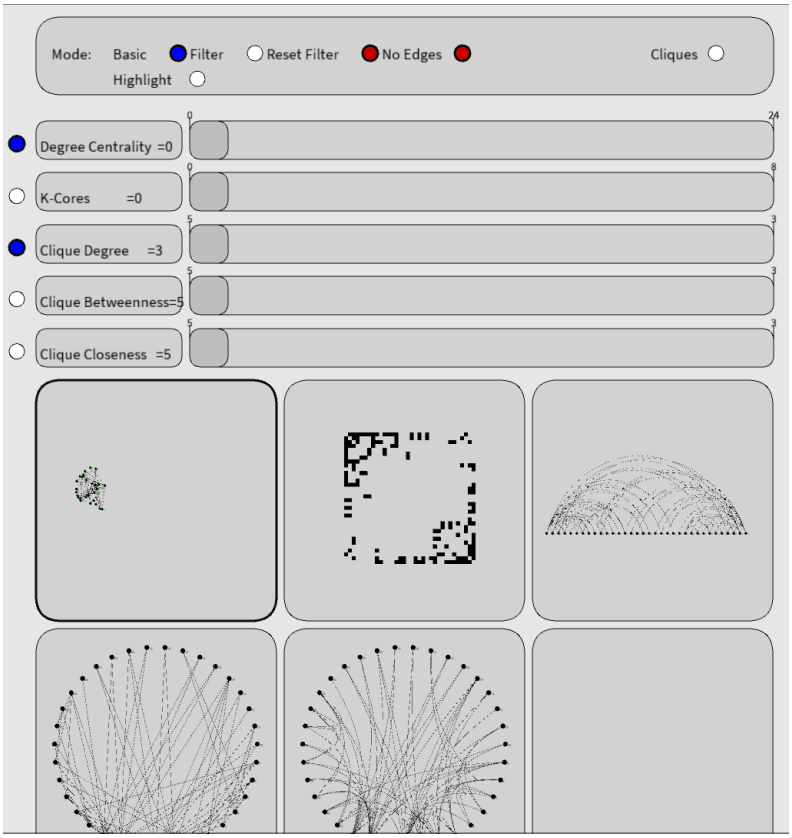


Figure 4.1: User Interface and its selections

The control panel demonstrates for the different idioms, filtering and abstraction techniques and with brushing options along with the interactions within. These can be seen in further detail in sections 4.4, 4.5, 4.6 and 4.7. The reduction options are implemented as radio buttons or sliders. The radio buttons utilize integers to determine the value to show and the

slider utilise the space given to it and determines a value to apply. Within this application sliders are used for the filtering and abstraction tuning. This is done by calculating the max values for each and using the position of the slider between the min point and the max point to determine the current value from the slider. The brushing selection consists of an array of radio buttons as well as the filtering techniques.

The idioms are demonstrated as sub charts by translating the charts to fit the constraints as well as reducing the amount of content shown in order to not clutter the chart. They are placed under the radio buttons and sliders, though will need to scroll down to have all visible. The scroll is implemented by adding a value to the y-axis of all menu options so by increasing all buttons, boxes and idioms will be offset concurrently showing what is above or below the screen range. An example of sub chart implementation can be seen in figure 4.2.

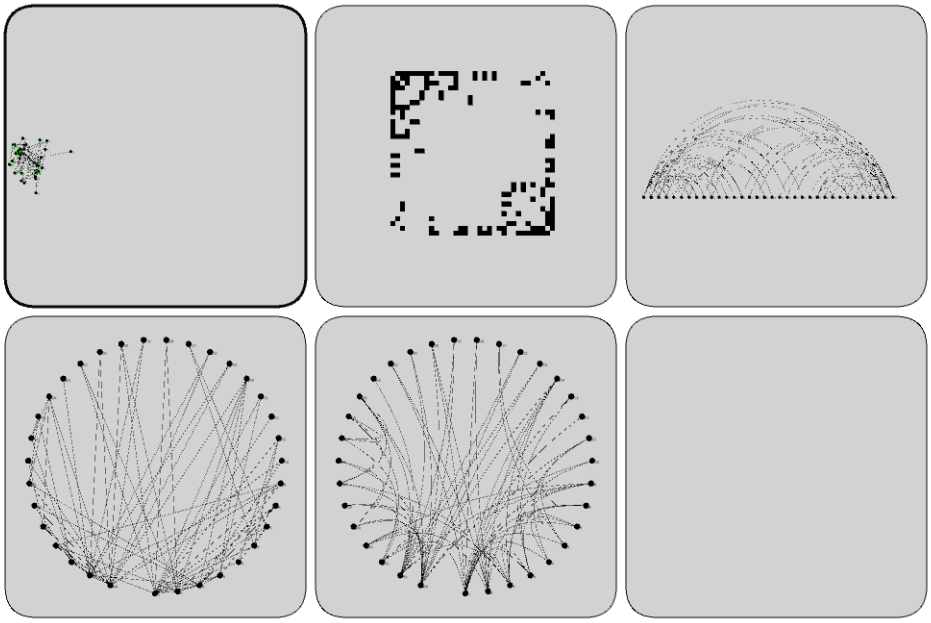


Figure 4.2: Sub Charts Selection within Implementation

4.4 Graph Types

The graph types employed are force directed layout, adjacency matrix, arc diagram, basic radial diagram and a curved radial diagram. The implementation of each is unique and this chapter will discuss each implementation.

For each graph, the implementation as the main idiom and as a sub-chart is discussed. Alongside this, how they interact with each filtering technique and abstraction techniques is discussed. The charts discussed are the force directed layout, the adjacency matrix, the arc diagram and the basic and curve radial layout. The graph types are described in greater detail in section 2.1.

Force Directed Layout

The force directed layout is implemented through a particle system. Each node acts as a particle within the system. The nodes have velocity, position and acceleration properties and, as such, they are fluid-like. Specific nodes within the idiom have connections with each other. These are demonstrated with a line between their points and a spring to keep them close to each other within the visualisation. When this occurs a spring is placed between the nodes which works under the principle of forcing the distance between the nodes to be as close as the desired amount in conjunction with the other forces in the system. However, this is not a fully implemented force directed layout. The node repulsion is not fully implemented within this visualisation. For this reason it does not act conventionally.

When the force directed layout is used as the main display idiom, the nodes, node edges and node name are displayed. For the sub-chart, the nodes and links continue to be displayed but the node names are removed for clarity, due to the names being either too big adding clutter crowding the nodes and links, or scaled down and being incomprehensible. The force directed layout interacts with both filtering of nodes and abstraction adapting its structure to both implementations. Filtering will remove nodes of low value, either of coreness or centrality, so while the layout will change to adapt to the new structure it will not be drastic. For cliques it will be repealing clusters with glyphs so this will drastically change the structure and the force directed layout will adapt dramatically to demonstrate this. The implementation of the force directed layout can be seen in figure 4.3.

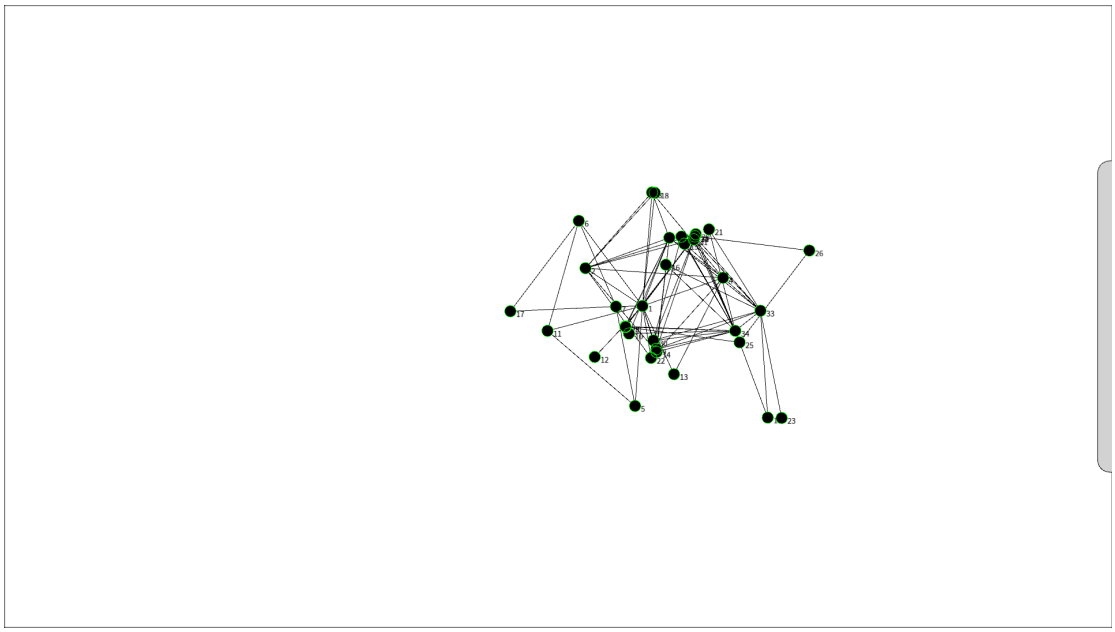


Figure 4.3: Force Directed Layout Implementation

Adjacency Matrix

The adjacency matrix consists of two steps: making the adjacency matrix and then displaying the matrix. Making the matrix involves creating a square boolean matrix with the dimensions being the amount of nodes in the data set. Then, after doing so, cycling through the edge list and highlighting within the matrix where two nodes interact. After doing this with each node the application will then return the boolean square matrix to be used for displaying the matrix. For displaying the matrix, the dimensions to display the chart must first be decided upon. This needs the top left co-ordinates and bottom right co-ordinates. From these points, the area for the points are calculated within. The vertical and horizontal real estate for the chart are calculated and divided by the amount of nodes in the data set. From this point the boolean matrix developed for the edges is used where each point is cycled through. If there is an edge then a rectangle is placed in the location and left empty if not. This is done for each row and column in the boolean array.

The matrix in the main display and sub charts display the same information with the only difference being that in the main display if the horizontal distance of the left edge and the sliding bar is less than the height, the matrix dimensions will adjust to fit in the smaller space. The matrix will display highlighted nodes as well as update if filtering is applied through either brushing or the defined techniques by recalculating the matrix with the filtered nodes and edges removed. Th adjacency matrix does not co-operate with abstraction in this implementation. The implementation of the adjacency matrix can be seen in figure 4.4.

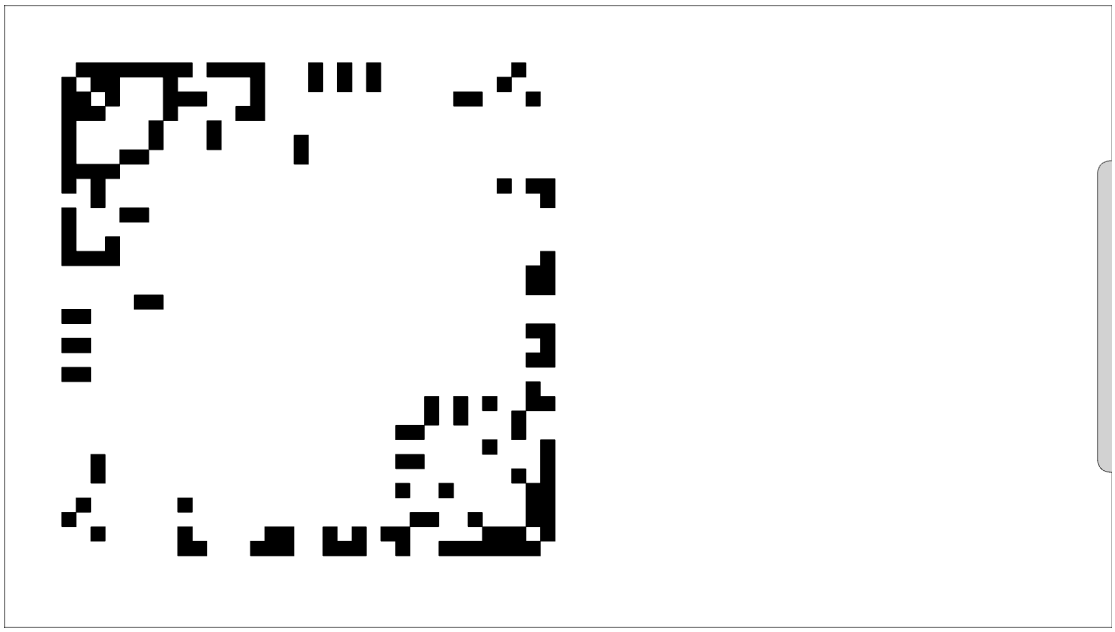


Figure 4.4: Adjacency Matrix Implementation

Arc Diagram

The arc diagram consists at the highest level of laying nodes in a horizontal layout and creating arcs between connected nodes. The arc will determine the size it has to display its layout and then divide the space for each node in the data set. The nodes are then connected by a curve. The height of the curve is based on the distance between the nodes normalised. The normalisation is to ensure the maximum height is consistant across data sets.

The chart, though similar in the main area and as a sub chart, do have key differences. The nodes aren't labeled within the sub chart while they are labeled within the main visualisation area. This is to reduce clutter and improve visibility in the sub chart. Alongside these differences the main chart also reduces in width if the sliding bar reduces the visualisation space too drastically. This is done by having the horizontal space given equal to the space given by the sliding bar minus the buffer range if the space given is less than the height for the main visualisation space. The arc diagram adapts to filtering and abstraction as well. It removes nodes that are removed through brushing or the established filtering techniques along with the associated edges. For abstraction the central nodes are reassigned to cliques and the edges are redirected towards them. The implementation of the arc diagram can be seen in figure 4.5.

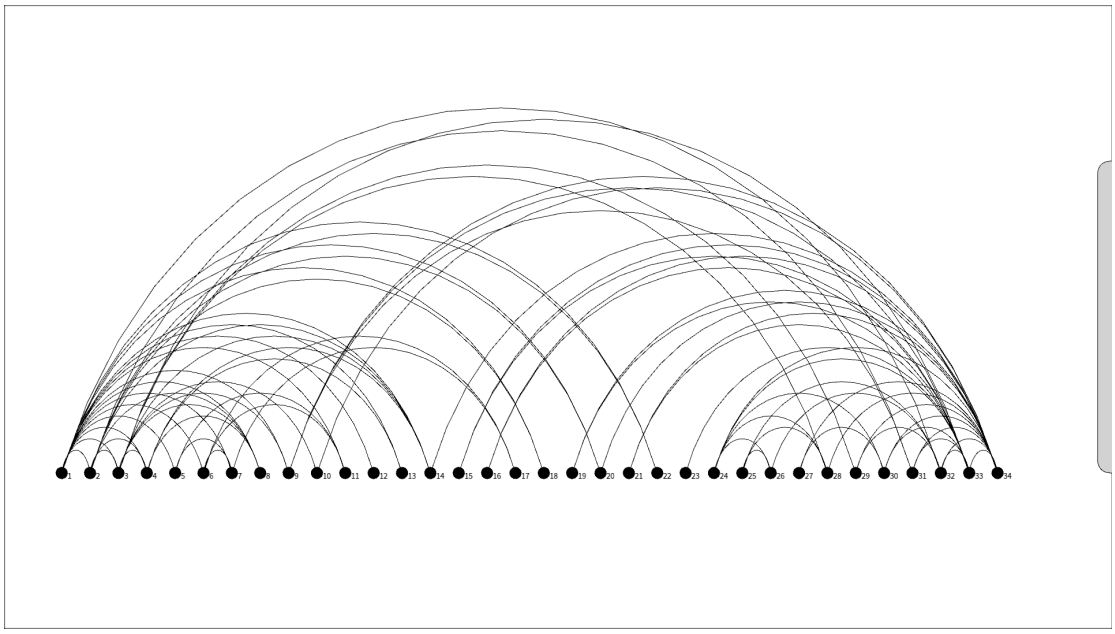


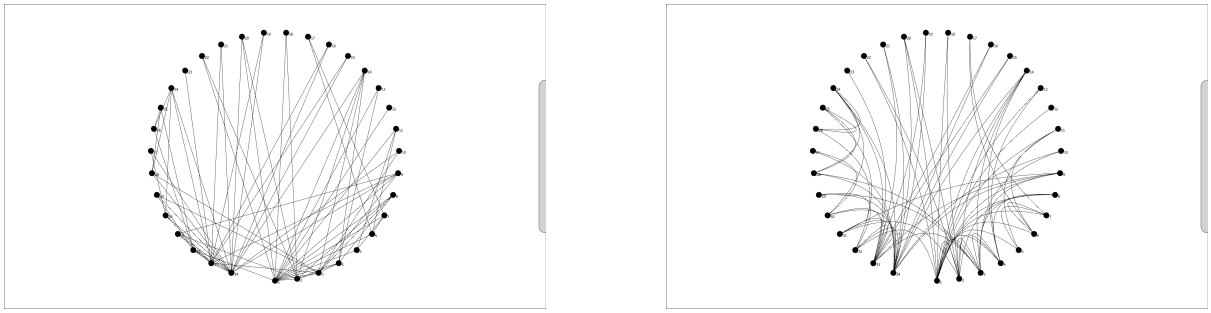
Figure 4.5: Arc Diagram Implementation

Basic and Curved Radial Layout

Both radial layouts use a similar implementation. They use a circular layout where the nodes are placed radially with gaps equal to the three hundred and sixty degrees divided by the

amount of nodes. The nodes are laid out similarly for both basic and curved. For basic radial layout the edges are direct from each node to another. For curved radial layout the function uses curves in which they will curve with the arc nearing the centre of the radial circle when connecting to its associated node.

Both layouts do not display their node labels in the sub chart and both layouts will decrease their diameters when the sliding bar takes too much space from the idiom in the main visualising area. This is done if the horizontal space is less than the height reducing the distance of the nodes from the centre of the chart. Filtering is applied for both brushing and the intended techniques. This is implemented through not drawing the nodes that have been flagged as filtered along with their associated edges. For abstraction there is a similar implementation but instead drawing the nodes and edges it is instead redirects the edges to a glyph that represents the node cluster and removing nodes that are contained within the clique. The implementation of the basic and curved radial layout can be seen in the figure 4.6.



(a) Basic Radial Layout Implementation (b) Curved Radial Layout Implementation

Figure 4.6: Basic and Curved Radial Implementation

4.5 Brushing

Brushing is what affects the manual filtering and highlighting of nodes. It is implemented by interacting with nodes based on mouse position. The implementation consists of using boolean arrays for both filtering and brushing where each value correlates to a node in the data set. The mouse position is compared to each node position and if they overlap it will flag the boolean array correlating to the chosen brushing style with the corresponding node value in the filtering boolean array if filtering is selected and the corresponding node value in the highlighting boolean array for highlighting. The brushing can be reset for both implementation with the click of the reset button that resets the boolean arrays. An implementation of highlight brushing can be seen in figure 4.7.

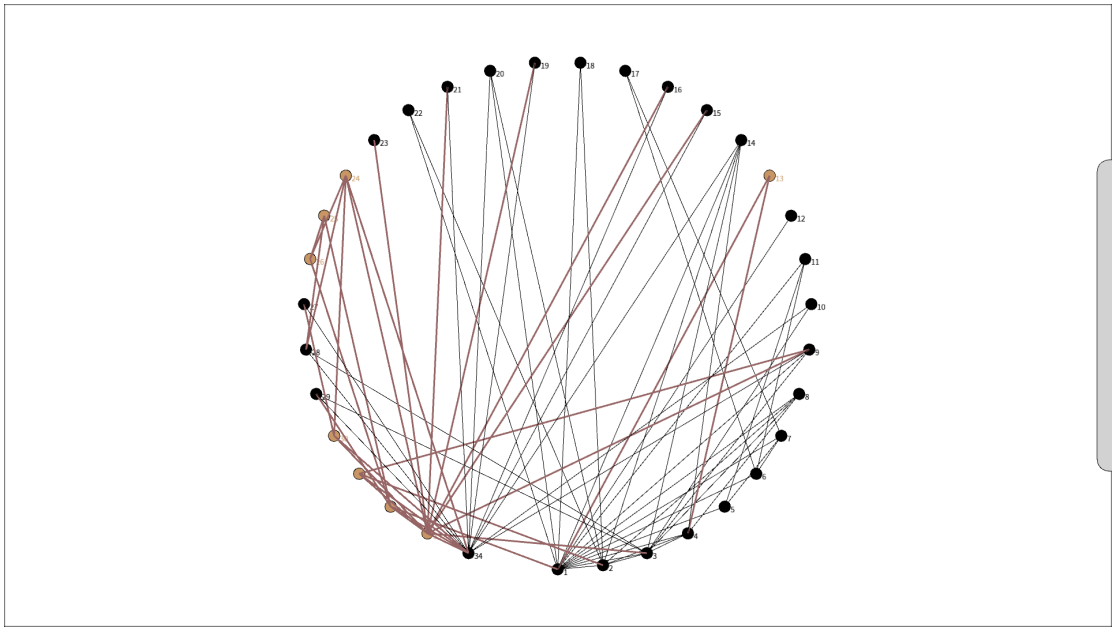


Figure 4.7: Brushing Highlighting Example

4.6 Filtering

Filtering is applied by two means, degree centrality and k-cores. Within this section the two techniques will be discussed in terms of implementation along with how they interact with other features.

Degree centrality is implemented through a simple procedure. The nodes are incremented for each edge they have when cycling through the associated edge list. During this process there are two key steps. The first is making sure not to count edges that use nodes that have been filtered. This step is implemented for the brushing to have an effect on the graph and not have filtering misrepresent the current graph structure. The other is to have the upper bound pre-calculated. This is based of the second highest level of degree to have certainty that at least two nodes are always connected. This number also adapts to filtering through brushing. Demonstrating this is a similar step. The minimum display value is checked for the value for each node. If the value of the node is less than the desired amount the node is not drawn. This is similar for the edge where if either of the nodes in an edge are not shown the edge will no be displayed also.

For k-cores the application has a function to find the k-cores based on the k-core value given. It operates by checking how many edges each node has. After doing this, it will repeat the process but not counting edges that had the minimum required by the k-core value. This step will repeat until there are no removed nodes. At this point, a functional k-core graph is present. Calculating the maximum k-core for the upper bound value in the slider in the control panel for k-cores, the procedure is called with increasing values of

k-cores until the returning k-core includes no nodes indicating the value prior was the greatest k-core value possible.

The k-core graph is also adaptable to brushing through filtering. In this case nodes removed will not count towards the k-core and likewise for the edges connected to the nodes. This means filtering can drastically effect the maximum k-core value. This means the maximum k-core value needs to also be re-evaluated after a node is filtered through brushing as well as the k-core itself. Both these actions are implemented by calling their functions.

4.7 Abstraction

Abstraction is implemented through cliques. The cliques are gathered from the pre-processing stage as seen in section 4.2. There are three elements within the abstraction process. These being procuring the information desired by the user in the interface, loading and storing the cliques in the background of the application and representing the cliques in the various idioms.

The elements collected from the user interface are the maximum core value and the clique priority methodology. The maximum core value is selected through a slider which will determine the maximum sized clique to be displayed. The radio buttons beside the sliders will determine the technique employed in representing the cliques in which they will declare the centrality metric to prioritize cliques with.

The first step in the implementation of abstraction is obtaining all the cliques. The text folder with the maximum clique value is located in order to know how many files to look for. The function will then find the clique files, extract them and then add them to an array list of all the cliques. This process will be repeated for all clique sizes and clique priority systems. The result from this phase is a two dimensional array for each clique priority technique and clique size, containing an array list of all the cliques relevant to that destination.

The next step is to store relevant values.The current requested maximum clique size and the methodology for clique representation is obtained from the user interface in the program. When these values change the function is called to update the clique values as new items are needed for the visualisation. The process involves using the maximum clique size and clique prioritisation method as keys to the array to gather the relevant array list. With this array list, cycle through and collect the nodes within a clique, the clique they belong to, the size of that clique and the node that will represent the clique. The implementation of this is to have a list of all nodes set to minus one. At the start of cycling through all the cliques a counter is set to zero. Two arrays will be made for the size of a clique and the representing node of size equal to the amount of nodes in the system. All nodes in a clique will be set to a counter value in the array of all the nodes. The size of the clique is recorded for the

counter value and the representing node for the clique of the counter value is the the first node in the clique. After these steps the counter is incremented one and the next clique is used. This process is repeated for all cliques.

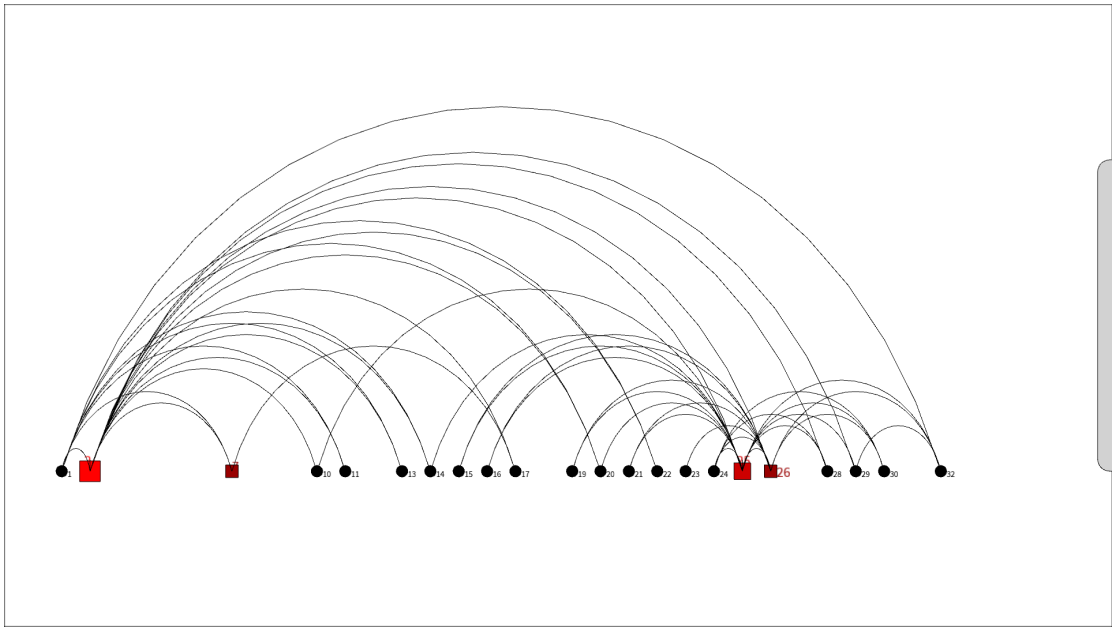


Figure 4.8: Abstraction Applied to Arc Diagram Chart

Now that the cliques are loaded and the relevant values are retrieved the clique representation step is in process. The cliques now need to be represented across the charts. When the cliques are chosen to be presented the drawing method is complex. When drawing a clique and it's connections there are four different routes the drawing algorithm can take. If both nodes are not in a clique, if the first node is in a clique, if the second node is in a clique and if both nodes are in a clique. When neither nodes are within a clique, the nodes and connections are drawn as normally. If a node is in a clique it will first check if it is the node that will represent the clique. If it is not then it is ignored but if it is then it will call a clique drawing function. Within this function it will use the node's value to check what clique it is in. Then it will determine the size of the clique and colour of the clique based on how many nodes it represents. The edges were drawn by checking the nodes whether they were in a clique for both ends of the link. If a node was within a clique, the clique it belonged to was checked through the nodes counter values and using the key obtained from this, determined the node representing the clique. This would be the location of the link for that node.

Through these processes the abstraction was able to display the clique depending on the users specifications. An example of the application of cliques to a chart can be seen in figure 4.8.

5 Evaluation

5.1 Data Sets

The data sets used for evaluation seek to demonstrate the range of tools used by the program. These data sets will be of multiple sizes, of real-world applications, of varying levels of connectivity and will represent the visualisation as a whole. In particular, the data sets used are Karate Club, Sparrow, Enzyme gn97 and the Euro Road network. These demonstrate unique areas and are also of different sizes of nodes and edges. Karate Club data set was procured from Zachary [39] and the Sparrow, Enzyme and Euro Road data sets were sourced from the source network repository [40], an interactive scientific network data repository.

Karate Club

Karate Club is a classic social network of a university Karate Club and is a popular data set for visualising community structure. The data set represents thirty-four members and seventy-eight connections of members who interact with each other outside the club. It is a small data set with an average degree of 4.58. The network has a maximum clique size of 5 and k-core value of 8.

Sparrow Flock

The Sparrow data set demonstrates a flock of sparrows in California. Each node represents a sparrow and edge representing when a sparrow was within five meters of another sparrow. The network represent an ecological data set. The data set contains 52 nodes with 516 edges. The maximum degree is 43 with an average degree of 19. The maximum k-core size of 15 and clique size of 12.

Enzyme g297

The Enzyme g297 shows the cheminformatics representation of the Enzyme g297. Checminformatic is the use of computational techniques to better understand the problems of chemistry.The data set demonstrates the enzymes chemical make up. The data set

consists of 121 nodes containing 298 edges. The largest degree is 7 with an average degree of 2.46. The maximum k-core value is 4 and maximum clique size is 3.

Euro Road Network

The Euro Road network shows the road infrastructure of a European road system. Each node represents a road with the edges representing where the roads connect with each other. The networks is large but is sparse. The data set consists of over 1174 nodes and 1417 edges. It is a large yet sparse data set. The average degree is 2 and the max clique size is 3. The max k-core value is also 3.

The properties for each of the four main data sets can be seen within table 5.1.

Data Set	Nodes	Edges	Max K-core	Max Clique
Karate Club	34	78	8	5
Sparrow Flock	52	516	15	12
Enzyme g297	121	298	4	3
Euro Road Network	1174	1417	3	3

Table 5.1: Data set values.

5.2 System Hardware

The evaluations in 5.3 and 5.4 were calculated using the same system hardware. The CPU used to run the evaluations is an Intel(R) Core(TM) i7-10750H CPU, the GPU used is a NVIDIA GeForce GTX 1650 and the RAM is 16.0 GB.

5.3 Frame rate

The frame rate is an effective manner in determining how efficient the data set can be visualised. The metrics were collected by recording the frame rate for the application in the last thirty seconds multiple times and finding an average. This was done by running the program for thirty seconds and then dividing the frame count by the time the application run time. These metrics were captured utilising the frameCount attribute and millis() function in the processing library inside the visualisation program. The resulting number would be calculated three times per metric and averaged. This is collected for every data set (see Table 5.2).

From these results the largest indicator of the frames per seconds is the amount of nodes on screen. Data sets with less than two hundred nodes would consistently perform near a 50 fps performance with slight dips when applying computationally intensive functions. This is further evidenced with the Euro Road network data set when filtering of nodes was applied,

Data Set	FPS: Idle	FPS: Interaction	FPS: High Filtering/Abstraction
Karate Club	58.16 FPS	51.14 FPS	57.86/55.94 FPS
Sparrow Flock	52.15 FPS	52.55 FPS	48.78/50.90 FPS
Enzyme g297	50.07 FPS	49.13 FPS	57.17/57.21 FPS
Euro Road Network	10.45 FPS	9.84 FPS	25.26/9.07 FPS

Table 5.2: Frame Rate performances per data set.

the frame rate improved in proportion to the amount of nodes that needed to be rendered. This promotes the ideal performance with this application, which is a data set of less than 500 nodes with any range of edges. Saying this, the large data set was still usable in function and intractability but not ideal for performance.

5.4 Pre-processing run time

The pre-processing run time is the sum of the run time for finding the cliques in a data set and filtering the overlapping cliques. The run time is collected when running the scripts within Visual Studio code where it returns the time to complete the task after completing the task. This procedure is back end and performance is not pivotal due to the processes not requiring to be completed within real time but would give an indication as to what properties to prioritize for quick implementation as well as the practicalities of merging the Python processing with the Java visualisation to complete these processes within the application in real time as a possible future feature. The run times are collected for each data set(see Table 5.3).

Data Set	Clique Detection Run time	Clique Filtering Run time	Total Run time
Karate Club	2.074 seconds	0.592 seconds	2.666 seconds
Sparrow Flock	9.547 seconds	127.12 seconds	136.667 seconds
Enzyme g297	0.79 seconds	0.553 seconds	1.343 seconds
Euro Road Network	5.035 seconds	0.585 seconds	5.62 seconds

Table 5.3: Pre-processing run time performances per data set.

Across both scripts the timing can be attributed towards specific data set attributes. Both Karate Club and Enzyme g297 are of similar size and hence have a similar computational time. The increased time for the clique detection can be attributed to the increased maximum clique size of 5 compared to Enzyme g297's 3. This shows the best indicator of run time for clique detection aside from data set size, this being the maximum clique size in a data set. Each increase in clique size is another iteration of the core loop of the script.

For clique filtering all the data sets are of similar run time bar the Sparrow Flock data set. The best comparison to understand the reason for this is the difference in Sparrow Flock and the Euro Road network data sets. While the Euro Road network contains twice as many

edges as the Sparrow Flock data set, the Euro Road network contains twenty times as many nodes. This shows that the nodes have very little effect on clique filtering. Rather, it is the amount of cliques inside the data set. The sparrow data set had an average node degree 19 and had 15 cliques to check while the Euro Road network had an average of degree of 2 and maximum clique size of 3. This indicates that highly connected data sets take significantly longer to calculate with more sparse data sets being better for pre-processing efficiency.

In summary, to reduce run time use smaller data sets that contain a reduced node and edge count as this can reduce run times noticeably for the clique detection stage. A more significant data set attribute is having a smaller maximum clique size, which is achieved with smaller average node degree values, can drastically reduce the clique filtering stage, especially if tied with having less nodes and edges. While this isn't a significant issue with the chosen data sets, larger data sets can cause the run times to increase exponentially.

5.5 Visual Comparison

The visual comparison evaluation is inherently subjective. It involves an evaluation of each data set at different degrees of reduction for filtering and abstraction techniques. This does not involve a statistical valuation but rather the user to make a judgmental decision on whether and by how much it improves the visualisation in terms of clarity.

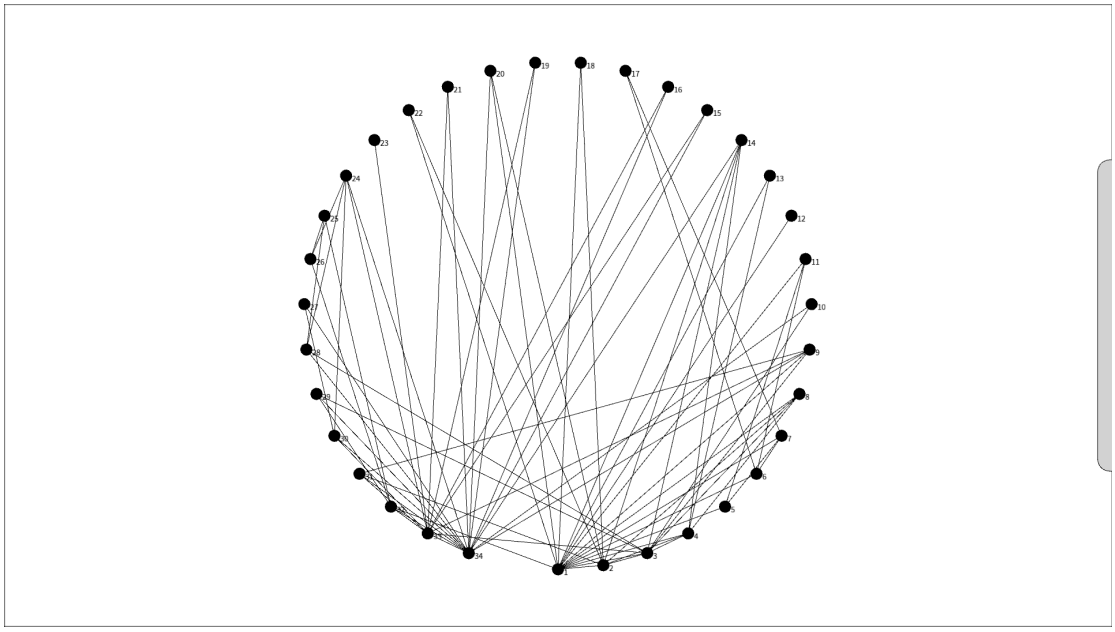
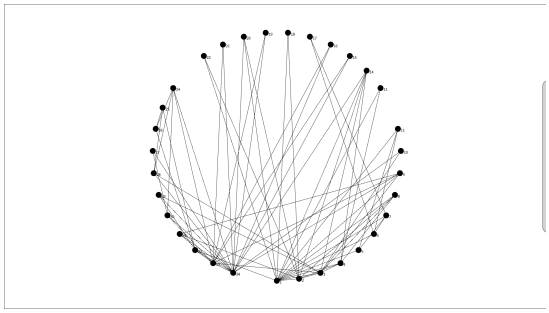


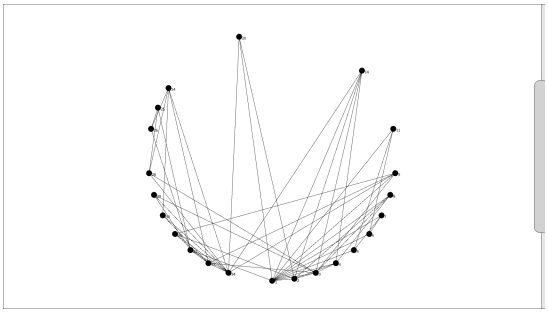
Figure 5.1: Karate Club, Basic Radial Layout, No filtering or abstraction applied

Filtering

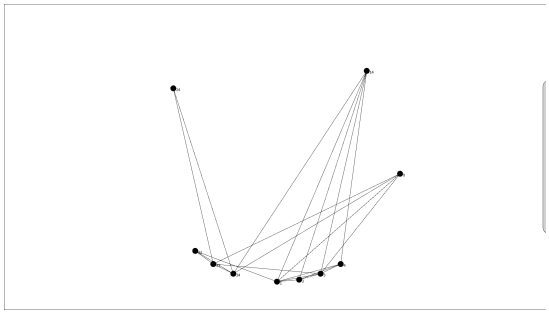
The initial data set is the Karate Club data set with the use of the basic radial diagram to demonstrate the effects of the various reduction techniques seen in Figure 5.1.



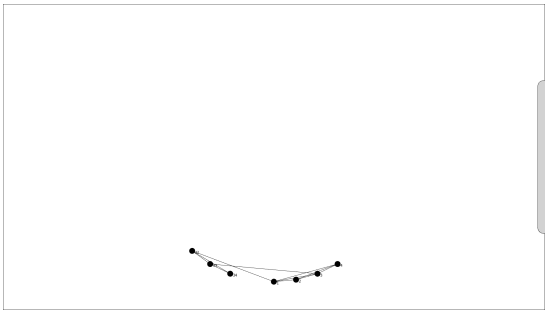
(a) Filtering of Degree Centrality less than 2



(b) Filtering of Degree Centrality less than 3



(c) Filtering of Degree Centrality less than 5



(d) Filtering of Degree Centrality less than 6

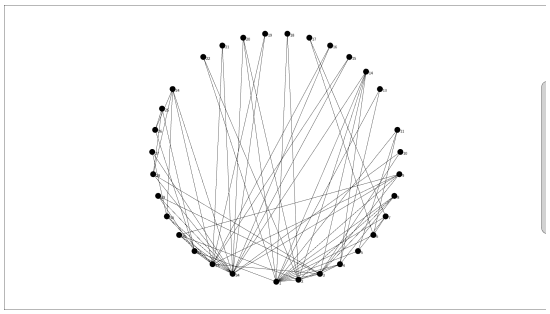
Figure 5.2: Karate Club data set with degree filtering at varying thresholds.

For degree centrality, the graph was filtered to varying levels of its maximum degree. This created the Figures 5.2.

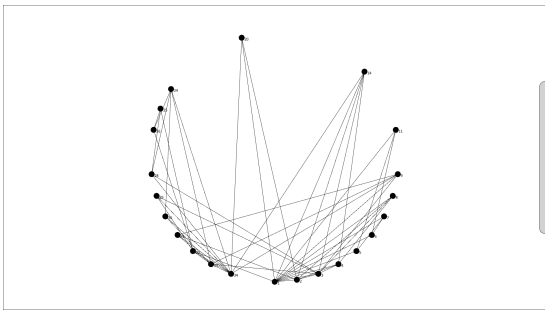
Within these Figures the effects of filtering are increasingly apparent. In 5.2(a) the filtering is very discrete. In the next Figure 5.2(b) the filtering has removed most non-clustered nodes and by 5.2(c) and 5.2(d) the chart no longer displays the majority of its links and nodes. Between the Figures 5.2(b) shows the best example of filtering by degree centrality in terms of pruning isolated and less centralised nodes from the idiom. The Figures 5.2(c) and 5.2(d) remove slightly too much to standard approaches but could pose necessary for extreme cases. This is an example of how the user must tune the filtering to their needs within the application for their chosen data set.

K-cores

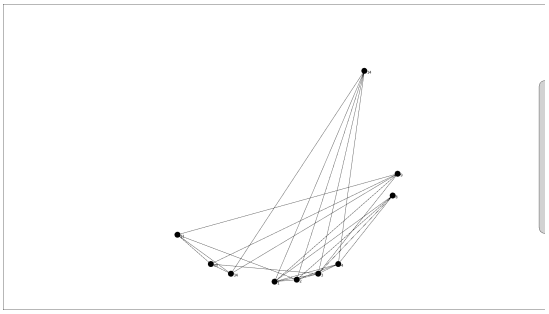
For K-cores, the graph was again filtered to varying degrees of k-core values. The value indicates the amount of edges to nodes within the k-core sub set. This is demonstrated within the Figure 5.3.



(a) K-core of value 2



(b) K-core of value 3



(c) K-core of value 4

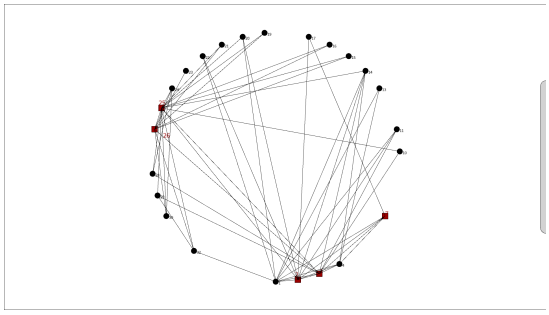
Figure 5.3: Karate Club data set with K-Core filtering at varying thresholds.

Within the above figures the effects of increasing k-core values should be apparent. The values displayed at each level are the core clusters within the graph. Each figure has a use depending on task, where coreness of the graph is subjective. In Figure 5.3(a) the coreness is still low and as such the graph includes most nodes. This level of k-core is great at simple pruning as it removes isolated or near isolated nodes. Figure 5.3(b) demonstrates k-cores at a higher value and how the k-core value has removed most nodes outside the main cluster. Within Figure 5.3(c) the maximum k-core value is displayed and demonstrates its ability in demonstrating the ability to identify only the core nodes in a data set.

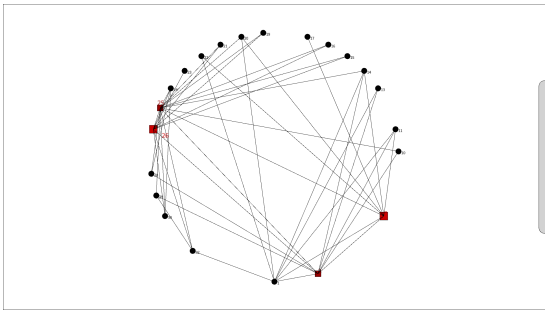
5.5.1 Extensive Data Set Comparison

Cliques based on Degree Centrality

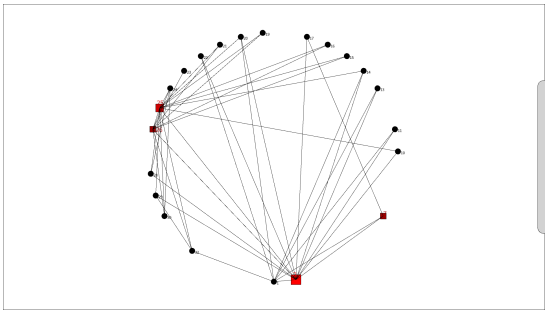
For abstraction, the idiom replaces maximal interconnected clusters with cliques. The values are the maximum clique sizes that the user wishes to display. The method for displaying the cliques is prioritising the cliques of higher sum of nodes for their degree centrality. This is displayed within Figure 5.4.



(a) Cliques, Degree Centrality, max value 3



(b) Cliques, Degree Centrality, max value 4



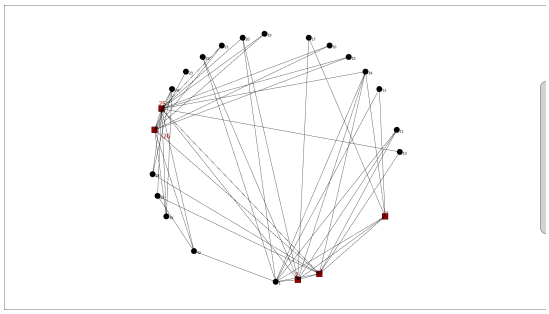
(c) Cliques, Degree Centrality, max value 5

Figure 5.4: Karate Club data set with cliques displayed as glyphs with degree centrality based priority of varying max clique size.

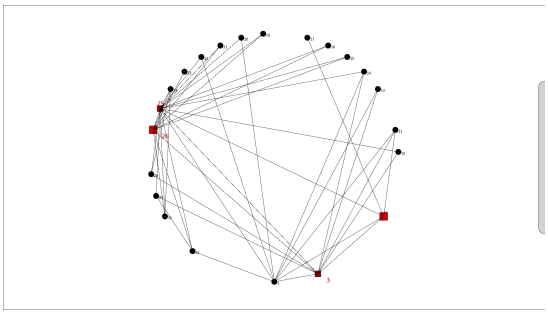
The above figures demonstrate the reduction of clutter while maintaining its connections. In contrast to the original unaltered Figure 5.1 the amount of nodes and edges is reduced yet no information outside the cliques is lost. In all figures the cliques are consistent with the higher values, generally the more the nodes are replaced. This abstraction is very important for removing clutter from data sets where they can not afford to lose information.

Cliques based on Betweenness Centrality and Closeness Centrality

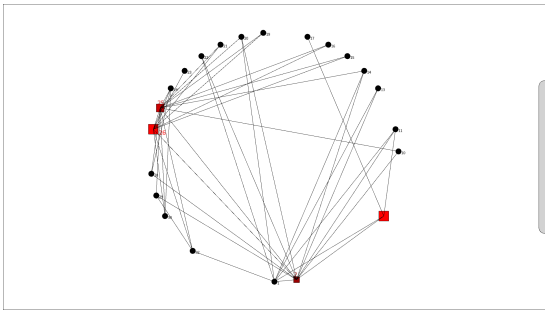
For cliques prioritised on the principle of higher sum of betweenness centrality and closeness centrality the cliques displayed are often different. This is to give the user a different methodology. This is best demonstrated in Figure 5.5.



(a) Cliques, Betweenness Centrality, max value 3



(b) Cliques, Betweenness Centrality, max value 4



(c) Cliques, Betweenness Centrality, max value 5

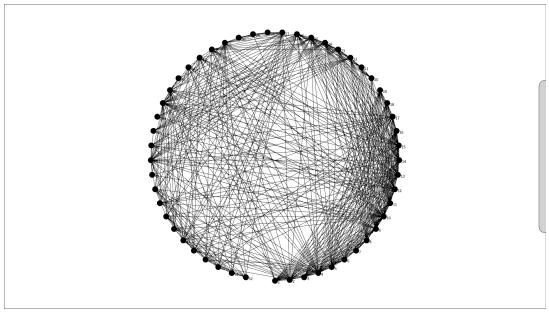
Figure 5.5: Karate Club data set with cliques displayed as glyphs with betweenness and closeness centrality based priority of varying max clique size.

Figure 5.5, and in comparison with Figures 5.4, the differences in the clique priority are evident. This allows the user to use different metrics on which they wish abstract their graph.

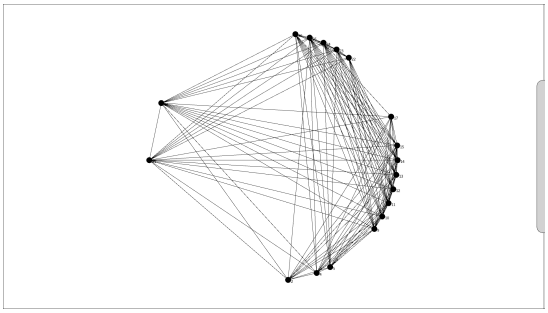
Within the Karate Club all forms of filtering and abstraction have been demonstrated. They all specialise in different areas of reduction and have sample tasks they specialise for each data set as well as possibly better representation for different idioms.

5.5.2 Sparrow Flock Comparison

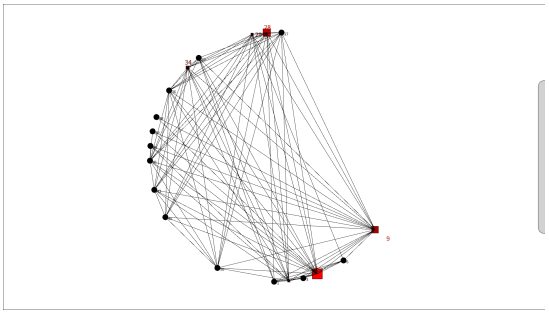
The Sparrow Flock data set contains a large amount of edges compared to its amount of nodes. This makes it ideal for filtering and abstraction. The large amount of nodes would lend itself well to other idioms alongside the base radial layout as seen in Figure 5.6.



(a) Sparrow data set represented as a basic radical



(b) Sparrow data set with k-cores of 15 applied



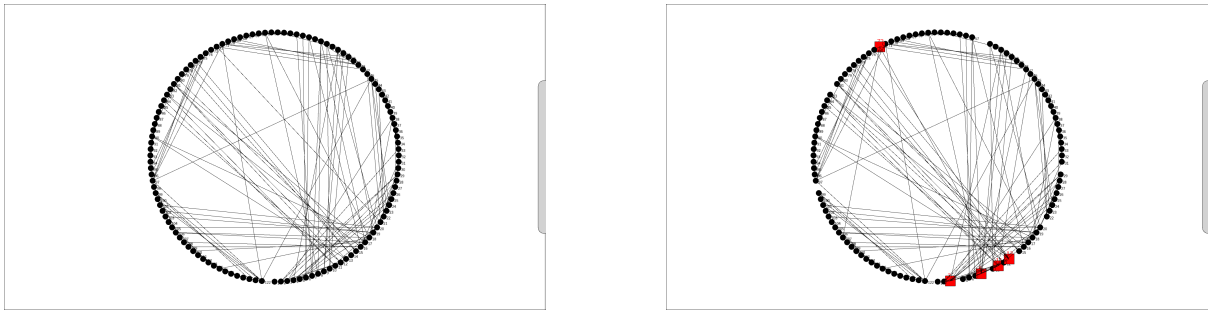
(c) Sparrow data set with degree centrality priority cliques

Figure 5.6: Different idioms for representing the sparrow data set unaltered.

Within Figure 5.6 the sparrow data set is displayed with both filtering and abstraction applied. Within the visualisation the maximal core is displayed in 5.6b where the core of the sparrow flock is represented. Within 5.6c the cliques are represented as glyphs. The reduction in clutter while maintaining the structure is very apparent. Both these techniques have demonstrated reduction in the visualisation while maintaining the structure of the data set.

5.5.3 Enzyme g297 Comparison

The Enzyme g297 data set maps the structure as an enzyme so filtering would be redundant as removing nodes would be inherently change the structure of the enzyme so it is not applicable. Howerver, abstraction is extremely relevant to the data set as it would improve clarity to the enzyme as seen in 5.7.



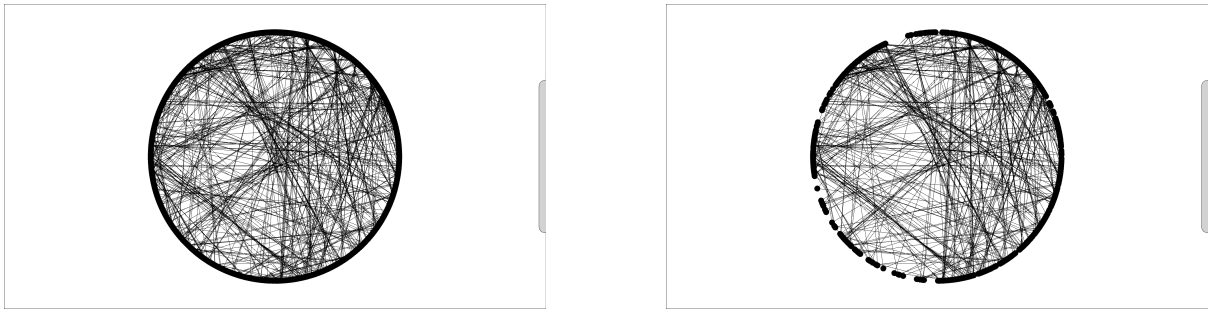
(a) Enzyme g297 without abstraction

(b) Enzyme g297 with abstraction

Figure 5.7: Enzyme g297 data set visualised without abstraction and with data set degree centrality priority abstraction.

5.5.4 Euro Road Network Comparison

The Euro Road network is a large network of roads. It will contain cul de sacs and intersections between two roads and, while these are not troublesome, they can in large quantities clutter the visualisation. Filtering, especially with k-cores, can greatly increase the readability while maintaining connections within the roads as demonstrated in Figure 5.8.



(a) Euro Road network, no filtering

(b) Euro Road network, K-core value 3

Figure 5.8: Euro Road network displayed in a radial layout with no filtering(a) and a k-core value of 3(b).

Within this comparison the reduction in visual clutter is apparent, thus increasing clarity greatly. Just by removing nodes that have less coreness than 2 the reduction in edges and nodes is apparent increasing clarity greatly. This is a prime example of filtering improving the coherence within the idiom while maintaining core structure of a data set.

5.5.5 Summary

This section evaluated the performance of both visualising the data sets as well as pre-processing them for the abstraction methods. The various reduction techniques were discussed and demonstrated for their filtering and abstraction properties along with displayed on real data to display their uses.

6 Conclusion

6.1 Conclusions

This project has displayed a range of idioms which can adapt to changes in individual data sets in real-time. The result is highly interactive, and includes manual filtering, highlighting and/or the utilisation of established filtering or abstraction techniques to reduce or reintroduce elements within the visualisation. The resultant tool allows greater control of the level of filtering or abstraction to a level that best suits the data set. Alongside this, the application can also deal with a wide range of data sets with the only encoding channels required being a list of edges for the network.

A variety of filtering and abstraction techniques are displayed and a multi coordinated view of all idioms is utilised, one where changes can be tracked globally across all idioms. This uses a novel clique prioritisation system within a fluid and interactive display. The filtering and abstraction techniques succeed in reducing clutter using unique approaches so that the visualisation demonstrates both the methodology and application scenarios. These methods were tested to ensure filtering and abstraction maintain the coreness, centrality and general structure of the original data set. The application also allows the degree of filtering and abstraction to be adjusted in real-time and it demonstrates the impact of these on the visualisation through multiple idioms. That is, the multi coordinated view contains multiple idioms that adapt to the reduction techniques that are applied. This occurs in real-time through either filtering, abstraction or brushing. Changes applied to an idiom are applied globally. This allows greater understanding of what the changes applied actually do to the data set globally as all idioms visible are updated concurrently with the applied changes. The clique priority system introduces a novel technique for determining clique representation within a network visualisation. This utilizes sum of degree, betweenness and closeness centrality and it allows multiple methodologies to decide the manner that the cliques are prioritised within the application as well as creating a set of nodes for each maximum clique size value.

Using these novel properties of the project, the application is able to successfully remove clutter from the screen while maintaining centralities, coreness and key features. This is

implemented through the use of filters which are curated to exemplify desired metrics in which to do so. These filters will have the upper and lower bounds pre-calculated for the data set and are re-evaluated if filtering with brushing occurs in order to adjust to the changes in structure of the network. This brushing allows for greater interactivity and has the ability to either filter or highlight nodes. Filtering also causing the filtering techniques to be updated to accommodate the changes in network structure. The multi-faceted display also lends itself to these properties and its multiple idioms allow for a choice of charts to be displayed. It also allows a mitigated view of the chart and how changes apply to it are brought about through filtering and abstraction. These features can work on a large range of network data sets, ranging from small to large amounts of nodes and varying levels of sparsity through edges.

6.2 Limitations

This project was limited with regard to efficiency issues that are tied to implementation decisions. For example, the graphing library processing was used for the visualisation due to its ease of implementation when alternatively implementing in C++ would have increased the project's efficiency. However, the result of the gain in efficiency would have been a loss of features from the application. Another example is pre-processing. By making these less computationally complicated, performance could have been improved. Thus, there is a trade-off between feature implementation and efficiency and as this project was seeking to showcase novel methods of visualisation it was decided to err on the side of implementation over efficiency.

Aside from improving efficiency, features could have been implemented over separate aspects of the application. An example of this are the filtering methods utilised within the application. Currently only degree centrality and k-cores are used to preserve the adaptive display and have the networks be updated when brushing filtering applied. The real-time updating of the networks would not work with betweenness and closeness centrality due to these algorithms taking a logarithmically completion time indicating the approach would not work for larger data sets in real time. This would lead to noticeable pauses in the application. If brushing was removed then these networks could be pre-processed and circumvent this issue along with allowing the introduction of other various filtering techniques.

Within the idioms, the Force Directed Layout is implemented. While the layout maintains it's structure under most data sets, the chart type has implementation limitations for sample data sets. The initial problem is that there are no repulsive forces between nodes. This was never implemented and as such nodes can overlap within the idiom in select cases. Another problem is the springs between nodes of edges can be too rigid. When data sets are too dense the imbalance between springs fighting to maintain a distance between nodes can

because the idiom to break and fail to demonstrate relevant information.

6.3 Future Work

The field of adaptive network visualization is opening up new research opportunities with future work ranging across visualizing, data input and pre-processing. These opportunities could include complementary filtering methods, expanding the clique priority metrics, more thorough clique filtering, additional encoding channels, tree network data sets, additional idioms, improved efficiency, statistical comparisons and clustering of nodes within data sets.

Implementing complimentary filtering methods would allow a greater range of comparison across methods and would better exemplify the strengths and weaknesses and also which cases to utilise. Examples of employable techniques include filtering based on betweenness centrality, closeness centrality and k-plexs.

Clique priority is a novel implementation and has room to expand. Currently there are three metrics for prioritisation: degree, betweeness and closeness centrality. There are many other centrality values as well, for example using k-plex values for cliques within cores.

The clique filtering system currently does one iteration when filtering cliques of the same size due to efficiency concerns for larger data sets. When filtering cliques of the same size a clique might be removed for sharing a node with another clique that could be filtered later for a similar reason. At this point the clique should be allowed to be displayed but is not within the current restraints. By applying a loop where the cliques are rechecked until no changes are applied to the list of cliques in the network, this would improve abstraction but would drastically increase the computational time of the application, specially for larger dense data sets.

One could utilise encoding channels for the input, such as node size or the connectivity of an edge between nodes. Normally these would dramatically reduce usable data sets but with correct tuning of intake it should be possible. These could enhance the visualisation and better demonstrate attributes of a data set.

Tree networks are a more complex data network. They are a hierarchical network generally containing a root node, parent nodes and leaf nodes. Each node within the tree network is connected back to the root node. Utilising this data type opens up more idioms and research questions.

Currently five idiom types are presented to the user to utilise. There are plenty supplementary charts that can be implemented within an interactive display. With a tree network this is increased greatly with possible idioms by altering current implementations

such as a node link tree or a radial dendogram or hierarchical radial layout and new implementations such as slankey charts, sunburst charts or tree maps.

Improving efficiency of pre-processing through algorithm optimisations alongside exporting visualisation code base to C++ to increase the efficiency of the application drastically along with general optimisations within the application as a whole.

Network statistics can be utilised to check if network structure, coreness and centrality values remain consistent. A plan to implement this was in place and the network file with the connections from the filtering and abstraction can be compiled. The application would be a statistical method to compare techniques and methodologies to determine if the network keeps its properties after reduction.

Applying clustering to nodes in data sets would better demonstrate node groups within data sets and allow certain chart types to be visualised more aesthetically due to similar edges being bundled.

Bibliography

[1] M. Grandjean, "Introduction à la visualisation de données : l'analyse de réseau en histoire," *Geschichte und Informatik*, vol. 18/19, pp. 109–128, 01 2015.

[2] Y. H. Healy and Conor, "Arc diagram." [Online]. Available: https://www.data-to-viz.com/graph/arc.html

[3] D. Holten, "Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 741–748, 2006.

[4] B. Walker, "Awesome radial tree. the reingold-tilford tree layout algorithm in polar coordinates." [Online]. Available: https://seeingcomplexity.wordpress.com/2011/02/05/hierarchical-edge-bundles/

[5] "Find k-cores of an undirected graph." [Online]. Available: https://www.geeksforgeeks.org/find-k-cores-graph/

[6] A. Montresor, F. De Pellegrini, and D. Miorandi, "Distributed k-core decomposition," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, 03 2011.

[7] F. Farahani, W. Karwowski, and N. Lighthall, "Application of graph theory for identifying connectivity patterns in human brain networks: A systematic review," *Frontiers in Neuroscience*, vol. 13, p. 585, 06 2019.

[8] F. Zhou, "Methods for network abstraction," *Core*, no. A-2012-6, 2012.

[9] A. Conte, D. Firmani, M. Patrignani, and R. Torlone, "A meta-algorithm for finding large k-plexes," *Knowledge and Information Systems*, vol. 63, pp. 1–25, 07 2021.

[10] M. O. Ward, *Linking and Brushing*. Boston, MA: Springer US, 2009, pp. 1623–1626. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_1129

[11] R. A. Becker and W. S. Cleveland, "Brushing scatterplots," *Technometrics*, vol. 29, no. 2, pp. 127–142, 1987. [Online]. Available: http://www.jstor.org/stable/1269768

[12] G. M. Namata, B. Staats, L. Getoor, and B. Shneiderman, "A dual-view approach to interactive network visualization," in *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, ser. CIKM '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 939–942. [Online]. Available: https://doi-org.elib.tcd.ie/10.1145/1321440.1321580

[13] N. Cao, J. Sun, Y.-R. Lin, D. Gotz, S. Liu, and H. Qu, "Facetatlas: Multifaceted visualization for rich text corpora," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1172–1181, 2010.

[14] P. McLachlan, T. Munzner, E. Koutsofios, and S. North, "Liverac: Interactive visual exploration of system management time-series data," *Research Gate*, no. 10, p. 1483–1492, 2008.

[15] S. G. Kobourov, *Handbook of Graph Drawing and Visualization*, 2013.

[16] P.-A. Ganaye, M. Sdika, B. Triggs, and H. Benoit-Cattin, "Removing segmentation inconsistencies with semi-supervised non-adjacency constraint," *Medical Image Analysis*, vol. 58, p. 101551, 08 2019.

[17] M. Ghoniem, J.-D. Fekete, and P. Castagliola, "On the readability of graphs using node-link and matrix-based representations: A controlled experiment and statistical analysis," *Information Visualization*, vol. 4, no. 2, pp. 114–135, 2005. [Online]. Available: https://doi.org/10.1057/palgrave.ivs.9500092

[18] C. Mueller, B. Martin, and A. Lumsdaine, "Interpreting large visual similarity matrices," in *2007 6th International Asia-Pacific Symposium on Visualization*, 2007, pp. 149–152.

[19] T. Nagel and E. Duval, "A visual survey of arc diagrams," 2014.

[20] B. Cornelissen, A. Zaidman, D. Holten, L. Moonen, A. van Deursen, and J. J. van Wijk, "Execution trace analysis through massive sequence and circular bundle views," *Journal of Systems and Software*, vol. 81, no. 12, pp. 2252–2268, 2008, best papers from the 2007 Australian Software Engineering Conference (ASWEC 2007), Melbourne, Australia, April 10-13, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121208000502

[21] G. J. Wills, "Nicheworks—interactive visualization of very large graphs," *Journal of Computational and Graphical Statistics*, vol. 8, no. 2, pp. 190–212, 1999. [Online]. Available: https://doi.org/10.1080/10618600.1999.10474810

[22] K. Shin, T. Rad, and C. Faloutsos, "Patterns and anomalies in k-cores of real-world graphs with applications," *Knowledge and Information Systems*, vol. 54 (3), pp. 677–710, Mar 2018.

[23] R. Laishram, "The resilience of k-cores in graphs," *Surface*, 2020.

[24] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, and E. Shir, "A model of internet topology using k-shell decomposition," *Proceedings of the National Academy of Sciences*, vol. 104, no. 27, pp. 11 150–11 154, 2007. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.0701175104

[25] S. P. Borgatti and M. G. Everett, "A graph-theoretic perspective on centrality," *Social Networks*, vol. 28, no. 4, pp. 466–484, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378873305000833

[26] S. Oldham, B. Fulcher, L. Parkes, A. Arnatkeviiciute, C. Suo, and A. Fornito, "Consistency and differences between centrality measures across distinct classes of networks," *PLOS ONE*, vol. 14, no. 7, pp. 1–23, 07 2019. [Online]. Available: https://doi.org/10.1371/journal.pone.0220061

[27] D. Edge, J. Larson, M. Mobius, and C. White, "Trimming the hairball: Edge cutting strategies for making dense graphs usable," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 3951–3958.

[28] J. Golbeck, "Chapter 3 - network structure and measures," in *Analyzing the Social Web*, J. Golbeck, Ed. Boston: Morgan Kaufmann, 2013, pp. 25–44. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780124055315000031

[29] G. Sabidussi, "The centrality index of a graph," *Psychometrika*, vol. 31, pp. 581–603, 1966.

[30] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, 1978. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0378873378900217

[31] E. Estrada, D. J. Higham, and N. Hatano, "Communicability betweenness in complex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 388, no. 5, pp. 764–774, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437108009473

[32] M. J. Newman, "A measure of betweenness centrality based on random walks," *Social Networks*, vol. 27, no. 1, pp. 39–54, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378873304000681

[33] R. Hanneman, "Introduction to social network methods," 2005.

[34] R. J. Mokken, "Cliques, clubs and clans," 1980.

[35] C. Dunne and B. Shneiderman, "Motif simplification: Improving network visualization readability with fan, connector, and clique glyphs," in *Proceedings of the SIGCHI*

Conference on Human Factors in Computing Systems, ser. CHI '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 3247–3256. [Online]. Available: https://doi.org/10.1145/2470654.2466444

[36] L. Bartram and C. Ware, "Filtering and brushing with motion," *Information Visualization*, vol. 1, no. 1, pp. 66–79, 2002. [Online]. Available: https://doi.org/10.1057/palgrave.ivs.9500005

[37] H. Chen, "Compound brushing [dynamic data visualization]," in *IEEE Symposium on Information Visualization 2003 (IEEE Cat. No.03TH8714)*, 2003, pp. 181–188.

[38] J. Power and K. Tourlas, "An algebraic foundation for higraphs," in *Computer Science Logic*, L. Fribourg, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 145–159.

[39] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of anthropological research*, pp. 452–473, 1977.

[40] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015. [Online]. Available: https://networkrepository.com

A1 Appendix

Link to GitHub repository with all code and sample data set:

https://github.com/GregoryPartridge/Dissertation