Visualization of Transformation of Graphs Based on Similarity Functions

by

Saima Parveen (MS2011012)

A thesis submitted

in partial fulfilment of the

requirements for the degree of

Master of Science by Research

in

Information Technology

ज्ञानमुत्तमम्

International Institute of Information Technology, Bangalore.

June 2013

Thesis Certificate

This is to certify that the thesis titled **Visualization of Transformation of Graphs Based on Similarity Functions** submitted to the International Institute of Information Technology, Bangalore, for the award of the degree of **Master of Science by Research** is a bona fide record of the research work done by **Saima Parveen (MS2011012)** under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Jaya Sreevalsan Nair

IIIT-Bangalore,

The 15th of June, 2013.

Abstract

Existing network visualization systems use variants and combinations of the node-link diagrams of the graph layout and visual representation of the adjacency matrices. However they cannot directly be used to show transformations on the graph, e.g., applying similarity functions on the adjacency matrix. Our proposed visualization technique uses linked views of the node-link diagrams and matrices to simultaneously view a network layout and its adjacency matrix. We use the clustering results from the similarity matrix to perform multilevel clustering on the data to reduce its complexity. We further propose parallel set-like representation to visualize a series of similarity matrices of the same data, e.g. time series data, different subspace clustering, application of different similarity functions, etc. Our methods are motivated by data mining applications for visually assessing clustering via similarity functions on the network. The similarity functions used in network data include both Euclidean-distance based ones, as well as role- or interactions-based ones, where the latter may not be a linear function of the link-based adjacency matrix. It can be used to analyze effectiveness of similarity function used for clustering by finding characteristic clusters from the matrix visualization using seriation algorithms, key one being the VAT (Visual Assessment of cluster Tendency) algorithm. For larger data sets, we propose a parallel implementation of the VAT algorithm using CUDA and multilevel clustering for simplifying the data set.

Acknowledgements

I would like to express my sincere gratitude to my advisor, Prof. Jaya Sreevalsan-Nair for the continuous support of my M.S. study and research, for her patience, motivation, enthusiasm, and priceless advice. Her guidance helped me in all the time of research and writing of this thesis, especially through the ordeal of graduation.

I am also thankful to my family and friends for their invaluable moral support.

–Saima Parveen

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Visual Assessment of cluster Tendency (VAT) [5] algorithm is regularly used for pattern recognition applications. In this work, we use VAT in various capabilities to understand and analyze large scale network data. A network can be characterized by various matrices, the primary and foremost being the adjacency matrix. We use clustering via similarity functions, a standard technique for reducing complexity in large data sets. Application of similarity functions on a network gives similarity matrices. Thus, in addition to adjacency matrix, using similarity matrix for visual analysis of a network gives deeper insights to the data.

1.1 Problem Definition

Given a large-scale data set, we use clustering based on similarity functions to compress the data. Apart from studying clustering capabilities of the data, we use these algorithms to track objects across clusters in a series of similarity matrices. The series of similarity matrices can be constructed for the same set of nodes in a network in several ways, e.g., time series data set, application of various similarity functions or permutation functions, or different subspace clustering in multi-attribute data.

For this study, we focus on small world networks in its undirected form characterized by a local structure where two connected nodes can have several neighbours and our study is limited to using similarity functions, which yields similarity matrices, as transformations to such a network. Though our work can be easily extended to any network and any normalized symmetric matrix which is characteristic of the network, its performance will vary depending on the type of the network.

1.1.1 Basic Definitions

A *small world network* is a type of mathematical graph in which most nodes are not neighbors of one another, but most nodes can be reached from every other by a small number of hops or steps.

A *two-way one-mode representation* uses a single mode, such as a set of objects, and is represented in two dimensions, say in the form of a matrix.

An algorithm for optimal ordering of entities in a two-way one-mode representation as a square matrix, like the adjacency matrix and similarity matrix, is a permutation algorithm, also known as a *seriation algorithm.*

Similarity measure gives the value of similarity between two objects. *Similarity functions* are binary functions on two objects that compute their similarity measure. *Similarity matrices* are two-way one-mode representation of similarity function values between any two elements in the set.

1.2 Motivating Applications

Consider Wikipedia vote network [37] or other similar voting networks, in its undirected form, where using a link-based analysis of the network, one can deduce who is voting for whom from the link. The Wikipedia vote network is a directed graph representing a network of wikidpedia users as vertices and where an edge from node i to node j indicates that user i has voted on user j for an administrator election. Generally a direct analysis cannot tell us all the nodes corresponding to who is being voted on, which are "similar" based on their "roles". Visualizing such a network using linked views of popular techniques, such as graph layout and visual representation of its adjacency matrix and similar applications, will be limiting as they exclusively view the existing direct links without showing any transformations on the network.

Finding role similarities on a network can be considered equivalent to the network being transformed in a non-Euclidean space, where its edges get transformed and the nodes give a different layout. Our work is motivated towards showing such transformations and subsequently tracking clusters in the network to help understand the network in metric spaces other than the Euclidean space. Along similar lines, recent research has shown that computing centrality can be considered to be a transformation or a function of the adjacency matrix [8]. Hence, analysis using matrix visualization of a network based on various centrality measures is incomplete without the facility to visually analyse transformations on the adjacency matrix beyond viewing the matrix itself.

Tracking membership of objects in clusters across series of similarity matrices can help in finding trends, and patterns and also in gaining more insight about the underlying varying properties of the data. If we observe that two objects are present in the same clusters in multiple instances in the series of similarity matrices, we can conclude that the objects are positively correlated. Suppose we have time series data of a network of co-authorship of n authors over x years and we apply k-ring neighbourhood similarity function on the network in each time instance. Finding two or more authors in the same cluster over large part of the x years may indicate that they are working very closely to each other owing to their belonging to the same workplace, or having the right set of complementary expertise in pursuing research on the same topic. These authors not co-authoring a paper together after a while might indicate that they no longer work together owing to change of employment or change in research interests, etc. Information about such an event becomes very apparent from the clustering tendencies found in similarity matrix series.

1.3 Clustering

Data analysis is a process of inspecting, cleaning, transforming, and modeling data with the goal of highlighting useful information. Our goal is to find pattern in linked views which will help in understanding clustering structure in the data. Using the clustering patterns, one can obtain multiple levels of detail for looking at the graphs, thus reducing the complexity of graph.

Objects are clustered or grouped to maximize the intraclass similarity and minimize the interclass similarity. Objects that are similar are grouped together and objects that are dissimilar lie in different clusters. Clustering is also referred to as *data segmentation* because it partitions large data sets into groups on the basis of similarity of their properties. Grouping the data can thus help in better understanding of the complexity of the data.

The advantage of clustering-based process for data segmentation is that it is adaptable to changes and helps single out useful features that distinguish different groups. By tracking these groups we can see the change in the trend or behavior of data, e.g., in time series data tracking membership of objects in clusters shows the temporal evolution of the data.

1.4 Our Contributions

Our key contribution is a visualization system or tool to enable data analysis using series of similarity matrices. We have made appropriate design choices by using known methods, such as shaded similarity matrices, seriation algorithms, multilevel clustering, and parallel sets, as components of this tool. Our contributions can thus be summarized as the following:

1. A visualization tool to analyze the graph layout of a small world network and its adjacency matrix and the transformations caused by applying a similarity function to the network. We use linked views of matrix and node-link diagrams to show changes in a network before and after transformation, with provisions to identify clusters across transformations. We have compared various seriation or permutation algorithms for showcasing clustering tendencies in the visualization of the similarity matrix and presented how VAT is optimal for our application.

2. We propose parallelized VAT (pVAT), an optimal parallel implementation of

VAT using CUDA based on Borůvka's algorithm.

3. For data simplification we use multilevel clustering, i.e., recursively cluster the data objects and after each iteration merge the clusters to form one node.

4. We propose parallel sets-like representation for tracking the cluster membership of objects in series of similarity matrices, i.e., to view how constituency of the clusters changes across the series. These series of similarity matrices can be generated for every time stamp in time series data, application of different similarity functions on the same data, or different subspace clustering in multivariate data.

Chapter 2

Related Work

In this chapter, we provide a broad overview of work related to this dissertation. We discuss matrix visualization techniques, graph drawing and visualizing small world graphs, linked views of graphs and its visualization similarity functions and parallel sets.

2.1 Matrix Visualization Techniques

Liiv [38] have given an exhaustive overview of seriation methods across various disciplines; defining seriation to be "an exploratory combinatorial data analysis technique to reorder objects into a sequence along a one-dimensional continuum so that it best reveals regularity and patterning among the whole series." Thus seriation is used heavily in matrix reordering to facilitate finding patterns in clustering, which he considers different from seriation due to the intermediate step that occurs between the two, which is "clustering with optimal leaf ordering". Liiv et al. [39] have presented a cross-disciplinary collaborative tool for seriation, called Visual Matrix Explorer

(VME), which facilitates comparisons of standard matrix reordering.

Bezdek and Hathway [5] have proposed a seriation algorithm along with a visual representation to assess the layout of clusters in the matrix, known as VAT (Visual Assessment of cluster Tendency). VAT uses a grayscale image of a permuted matrix to show blocks along the main diagonal, which are representative of clusters. There have been several improvisations done to VAT owing to its complexity being $O(n^2)$. Successors of VAT relevant to our work are reVAT (revised VAT) [27], bigVAT [28] and sVAT (scalable VAT) [21]. reVAT uses profile graphs to replace the dissimilarity image thus reducing the number of computation. bigVAT and sVAT are scalable algorithms based on sampling representative entities to reduce the size of the data. We alternatively propose parallelized VAT where the implementation of VAT uses a parallel implementation of finding minimum spanning tree in the graph using Borůvka's algorithm, as implemented by Vineet et al. [53]. We propose the parallel implementation of VAT (pVAT) because VAT is not scalable for large graphs and scalable version of VAT, i.e. bigVAT and sVAT rely on good choices of representative data to downsample the given graph.

Mueller et al. [45, 46] have performed a thorough analysis of vertex reordering algorithms, which is relevant to representation of clusters in matrix visualization. In [45], Mueller et al. have used pixel-level display to scale up the visualization for larger data sets. In [46], Mueller et al. have referred to visual representation of adjacency matrix as visual similarity matrices (VSM) and specified common structural features of the VSM along with their graph-based interpretations. The matrix-node link synchronized views show that the specific "features" in the VSM correspond to a relational pattern between the vertices in the graph. One such feature, diagonal lines carry information of clustering tendencies, as showcased in VAT. Though VAT focuses on blocking patterns along the diagonal of the matrix, Mueller et al. have analysed diagonal entities on and off the main diagonals.

Wishart has discussed about shaded distance matrices [59] where similarity matrices are reordered by constructing a dendrogram and reordering the dendrogram to minimize the sum of weighted similarities. Wang et al. [55, 56] have used (a) nearest neighbour clustering and ordering using decision tree algorithm to visualize a shaded similarity matrix in [56], and (b) a combination of conceptual clustering in machine learning, and cluster visualization in statistics and graphics whose complementary properties help in interpreting clusters better, in [55]. Wang et al. [55, 56] also use the hierarchical clustering to choose representative nodes to display in the shaded similarity matrix. Shaded similarity matrices [56, 59] use a range of shading, just like the VAT image, where dark to light shading implies decreasing similarity between given two objects. Wang et al. [56] have used conceptual clustering by using a concept tree to find clusters and have found a reordering of the matrix based on these precomputed clusters. The shortcomings of this method are: (a) high dependency on specific definition of similarity, (b) basic assumption of conceptual clusterability of data, which is not guaranteed for all data, (c) limited scalability of shaded similarity matrices. However shaded similarity matrices can be scaled using sampling and ensemble approaches. Though results of VAT also depends on the similarity function used and the inherent clustering in the data, VAT does not involve computation of agglomerative hierarchical data structures, which becomes computationally intensive for large data sets. Also VAT can be considered to be related to single-linkage algorithm [22] as opposed to the average-linkage algorithms used in shaded similarity matrices [55, 56, 59].

2.2 Surveys of Graph Drawing and Visualization

Some of the survey papers on graph drawing and visualization are very relevant to our work. These include the following: on graph visualization and navigation techniques used in information visualization by Herman et al. [25], on graph layout problems by Diaz et al. [10], and on the state of the art in visual analysis of large graphs by Landesberger et al. [54].

In [25], Herman et al. have discussed about handling large graphs in the context of information visualization and layout choices for particular types of data. In [10], Diaz et al. have discussed about graph layout problems from an algorithmic point of view, which include: (a) Minimum Linear Arrangement problem, (b) Bandwidth, (c) Cutwidth problem, (d) Vertex Separation problem, (e) Sum Cut and Profile, and (f) Edge and vertex bisection. In [50], Purchase has discussed the results of the experiments conducted to check which graph drawing algorithms perform better than the human-in-the-loop case. Different experiments were conducted to check the most effective aesthetics graph drawing algorithms for relational information and graph drawing algorithms from a human readability point of view.

2.3 Small World Graphs

Watts and Strogatz [58] have explained the characteristics of small world networks based on the small world phenomenon, also known as six degrees of separation [18].

nBarrat and Weigt [4] have used analytical and numerical tools to validate the geometrical properties of small world graphs. There are several multiscale, multilevel visualizations of graph layout of large small world network (SWN) [3, 2, 52, 11], which are scalable applications. Erdelyi and Abonyi [12] have worked on node similarity based visualization is very similar to ours in the use of VAT for similarity matrix representation. They have proposed a new similarity function and implemented dimensionality reduction to obtain a similarity matrix in a lower-dimensional embedding, on which subsequently VAT is used. Correa et al. [8] have computed and used sensitivity of centrality measures to reduce a network. They have explored computing centrality using a variational definition of an adjacency matrix, eigenvector centrality and markov centrality.

2.4 Linked Views of Graphs

Koren and Harel [32] have discussed the integration of two approaches: hierarchical clustering using a dendrogram and low-dimensional embedding. Abello and van Ham [1] have proposed a visualization system that is characterized by the available screen real estate S and the available RAM M to provide graph and matrix views. Henry and Fekete [23] have proposed a network visualization system, MatrixExplorer, which uses two representations, namely, node-link (NL) diagrams and matrices, using inputs from social scientists. Apart from standard interactive filtering and clustering functions, MatrixExplorer has functionalities for both automatic and manual reordering matrices, annotating and comparing observations across different layouts and arriving at consensus amongst several clusterings. They give an elaborate literature review on work done on graph visualizations. They used participatory design techniques to arrive at MatrixExplorer. Some relevant results from the requirements they obtained from the users, who are social science researchers, are: higher preference for matrix-based representations over NL diagrams as it was faster to display and easier to manipulate for larger data sets; cluster detection was essential for social networks analysis; and aggregating networks using clusters presented results well. However in [24], they have evaluated MatrixExplorer with the shortcoming of huge cognitive load during context switches. Hence they [24] have proposed the idea of integrating matrix and graph visualizations to give a hybrid representation called MatLink, which consists of a matrix visualization with linear NL diagram overlaid on the edges of the matrix and a dynamic topological feedback on mouseover. They used energy-based clustering of graphs with non-uniform degrees (LinLog) algorithm [48] for the NL representation in MatLink. Elmqvist et al. [11] have proposed Zoomable Adjacency Matrix Explorer (ZAME), an interactive visualization tool for exploring large scale graphs their representaion is based on adjacency matrix. ZAME can explore data at many levels, aggregation of nodes is based on “symbolic data analysis” and aggregates are arranged into a pyramid hierarchy.

2.5 Visualization Techniques for Linked Views

Ghoniem et al. [15, 16] have analyzed comparison of graph visualization techniques between node-link (NL) and matrix based representations using controlled experiments. In [15], it was shown that NL diagrams are suited for graphs that were “almost trees”, but performed very poorly for almost complete graphs and denser networks. Though the experiments conducted in [15] primarily were for low-level readability tasks and not specifically for social networks, since SWN are locally dense, matrix visualization is better suited for clarity and is less sensitive to density variations. However NL scores better for networks with relatively lesser number of nodes.

Spectral graph partitioning is one such application where our proposed method of linked views of graph, its adjacency matrix and a function of the adjacency matrix will be effective. Spielman and Teng [51] have explained how matrix representation of a graph is used for spectral partitioning where the eigenvector of the second-smallest eigenvalue of the Laplacian matrix of the graph is used for finding separator of the graph.

2.6 Similarity Functions

Liu et al. [40] have proposed guidelines to users on the choice of link-based similarity measures depending on applications. We have identified for our experiments two similarity measures which are not a linear function of adjacency matrix, namely, role similarity [30] and matrix powering [64] for identifying k-ring neighbourhoods [13]. Computing centrality measures in a network can be done using linear or exponential function of the adjacency matrix [13]. Thus one can compute k-ring neighborhoods of a node using k^{th} power of the adjacency matrix, which can be used as a similarity matrix for clustering purposes.

2.7 Parallel Sets

Finding trends and outliers in multivariate data is generally done using parallel coordinates and parallel sets. In our work, in the case of visualizing trends in a series of similarity matrices, such representations give better insight than using multiple instances of matrix visualization. Parallel sets like representation collate the clustering results across the series, which preserves the details of the clustering process such as number of clusters, cluster size, etc. Such a representation weeds out other information pertaining to the nodes in the graph.

In [29], Alfred Inselberg et al. have proposed parallel coordinate for visualizing high dimensional data in a plane. In [62], Zhao et al. have used polyline clustering and coordinate sorting for complex and large dataset using spectrum based clustering.

Parallel sets is a variation of parallel coordinate system. Parallel coordinates is a common way of visualizing high-dimensional geometry and analyzing multivariate data. Adding more dimensions in parallel coordinates involves adding more vertical axes. The order of the axes is critical for finding features, and in typical data analysis many reorderings will need to be tried.

In [9], Dasgupta et al. have used a perceptually motivated metric i.e., screen space based metrics for visualizing high dimensional time varying data as they feel that it performs better than data-based metrics. They proposed an user centric approach and integrated multiple views to benefit search in temporal patterns. To reduce the cluttering of edges and effectiveness in visual clustering. Zhou et al. [65] have proposed a framework where the shape of edges are changed and order is maintained. To find clusters in parallel coordinates, Zhou et al. [63] have provided a mechanism to enhance the cluster effect by reducing the edge clutter. In [33], Kosara et al. have proposed parallel sets for visualizing and exploring categorial data. Parallel sets allow the user to remap the data to new dimensions.

2.8 Research Gaps

We find that in the information visualization community, matrix visualization has been routinely applied for adjacency matrices for undirected graphs. Adjacency matrices are generally visualized as a binary shaded representation of a matrix, indicating presence or absence of edges between vertices. The matrices are represented in two-way one-mode form. For our need for visualizing grayscale matrix representations for similarity matrices, we find that shaded similarity matrices and VAT variants are apt. While shaded similarity matrices involve construction of hierarchical data structures, VAT is a simpler algorithm with low spatial overhead. Also, VAT is unconventional compared with the popularly used seriation algorithms as VAT is equivalent to a single linkage hierarchical algorithm, while the algorithms used in shaded similarity matrices are average linkage. Since VAT has a quadratic time complexity, it is inefficient for large scale data sets.

Multilevel clustering has traditionally been done on the node-link diagrams using nearest neighbor consideration, which is satisfied by VAT owing to its relation to single linkage algorithm. Clustering done in the matrix form of the network has been carried out in ways specific to research communities: (a) in information visualization community, multilevel clustering of adjacency matrices is done for visual assessment and further processing pertaining to building focus+context representations, and (b) in machine learning community, multilevel clustering is done using similarity functions. We have not found any work pertaining to multilevel clustering in a network combining the two approaches.

We have narrowed down our current domain to be small world networks, owing to the known behavior of similarity functions in such a network. While there is existing research on visualizing similarity matrices for this domain, we have not found any visual analysis for a series of similarity matrices. This is unique to our work, where we have adapted parallel sets representation to see clustering trends in a series of similarity matrices.

Chapter 3

Basics

Our visualization system comprises of the algorithms for graph layout, seriation and similarity functions and cluster isolation and tracking for enabling the system. We have shortlisted algorithms for the user choices based on the popularity of these algorithms in the concerned communities as well as our requirement to showcase a system for proof of concept.

3.1 Graph Visualization

A graph can be visualized using a layout as well as a graphical representation of its adjacency matrix, as shown in Figure 3.1 for the social network of coauthors data set [42] consisting of 475 nodes(authors) and 1250 edges, edges connect the authors who have co-authored one or more papers. As we explained in Chapter 2, there has been a lot of work on the best representations of the graph in different scenarios. Though the node-link layout is useful for us to study the transformation of graph on application of a similarity function, our work primarily focuses on clustering tendencies in similarity matrices. Hence, we will be using the grayscale representation of matrices more than the node link layout.

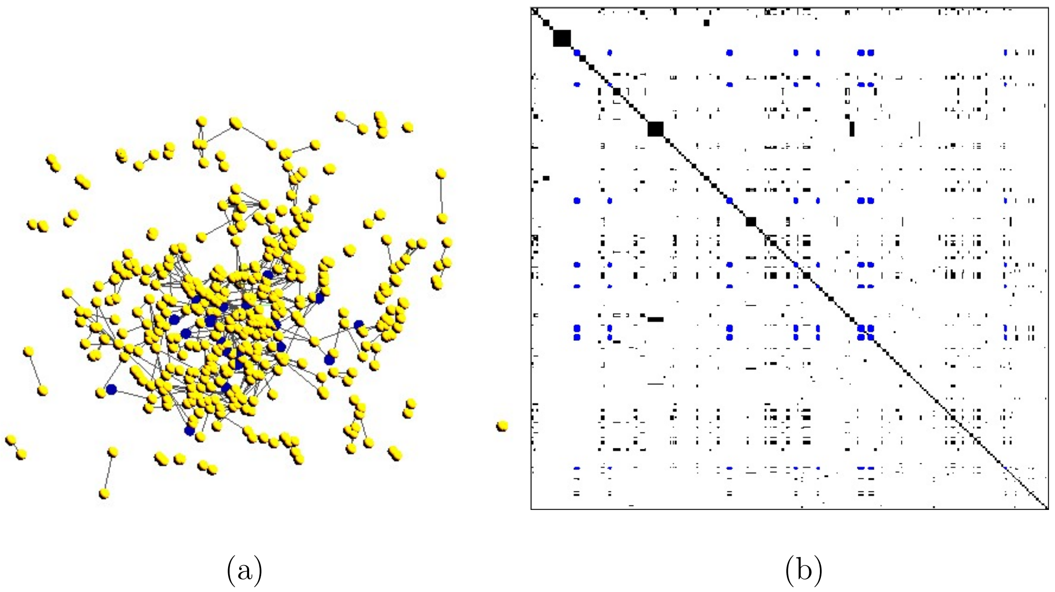


Figure 3.1: Data set of network of coauthors [42] is visualized using (a) node-link diagram using Fruchterman Reingold layout, and (b) grayscale representation of its adjacency matrix. Same set of nodes has been highlighted in blue in both the representations.

3.1.1 Graph Layout

A layout algorithm determines perception of various geometrical properties of a graph. Thus an appropriate graph layout of a network can give us a better understanding of its inherent clusters. We will be using the following nine layouts: in two-dimensional space, circular layout; and in three- dimensional layouts: random, sphere, Fruchterman Reingold, grid Fruchterman Reingold, Kamada Kawai, springs, large graph, and

Graphopt.

3.2 Similarity Functions

For an ordered set of elements, applying a similarity function on pairwise choice of elements gives an idea of similarities and dissimilarities inherently present in the data. Similarity matrices are two-way one-mode representation of similarity function values between any two elements in the set. Subsequent clustering on a similarity matrix gives insight to the data and can be used for reducing complexity of the data; and hence is a useful data mining tool. Similarity matrices are normalized and symmetric.

In graphs, clustering can be done in a similar fashion using similarity matrix based on its nodes or vertices. Usually the similarity matrix is a function of the adjacency matrix, since both matrices are two-way one-mode representation of the nodes. We focus on the following similarity functions: Identity, Jaccard, Dice, Inverse log weighted, Cocitation, Bibliographic coupling, k-ring neighbourhood, and RoleSim. A brief description of these functions:

1. Identity: The identity function implies the adjacency matrix is the similarity matrix.

2. Jaccard: The Jaccard similarity coefficient of two vertices is the number of common neighbours divided by the number of vertices that are neighbours of at least one of the two vertices being considered.

3. Dice: The Dice similarity coefficient of two vertices is twice the number of common neighbours divided by the sum of the degrees of the vertices.

4. Inverse log-weighted: The inverse log-weighted similarity coefficient of two vertices is the number of their common neighbours, weighted by the inverse logarithm of their degrees. It is based on the assumption that two vertices should be considered more similar if they share a low-degree common neighbour, since high-degree common neighbours are more likely to appear even by pure chance. Isolated vertices will have zero similarity to any other vertex.

5. Cocitation: Two vertices are cocited if there is another vertex citing both of them. Cocitation simply counts the number of times two vertices are cocited.

6. BibCoupling: The bibliographic coupling of two vertices is the number of other vertices they both cite.

7. k-ring Neighbourhood: k-ring neighbourhood function implies that two vertices are similar if they have common neighbours in k-hops.

8. Role Similarity: RoleSim function [31] is based on the recursive node structural similarity principle, which states that "two nodes are similar if they relate to similar objects".

Figures 3.2 and 3.3 show the effect of applying different similarity functions on the co-authorship network [42], before using VAT as the seriation algorithm. The figures show that for this particular co-authorship network except for the k-ring neighborhood function, the other similarity functions do not show well-defined clusters.

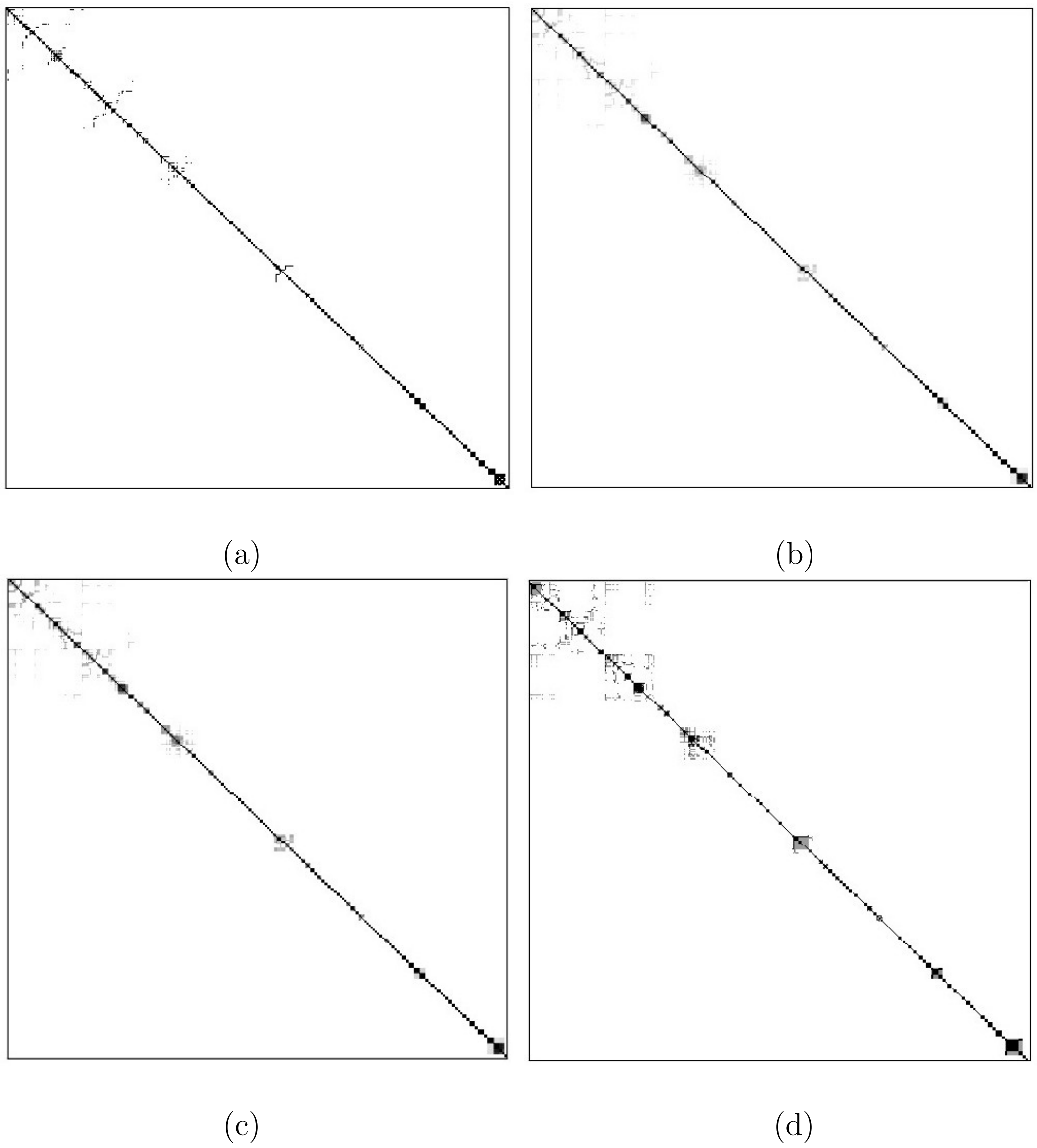


Figure 3.2: For the social network of coauthors in [42], the adjacency matrix, as shown in Figure 3.1(b), transforms on application of the following similarity functions and seriation using VAT: (a) Identity, (b) Jaccard, (c) Dice, and (d) Inverse log-weighted.

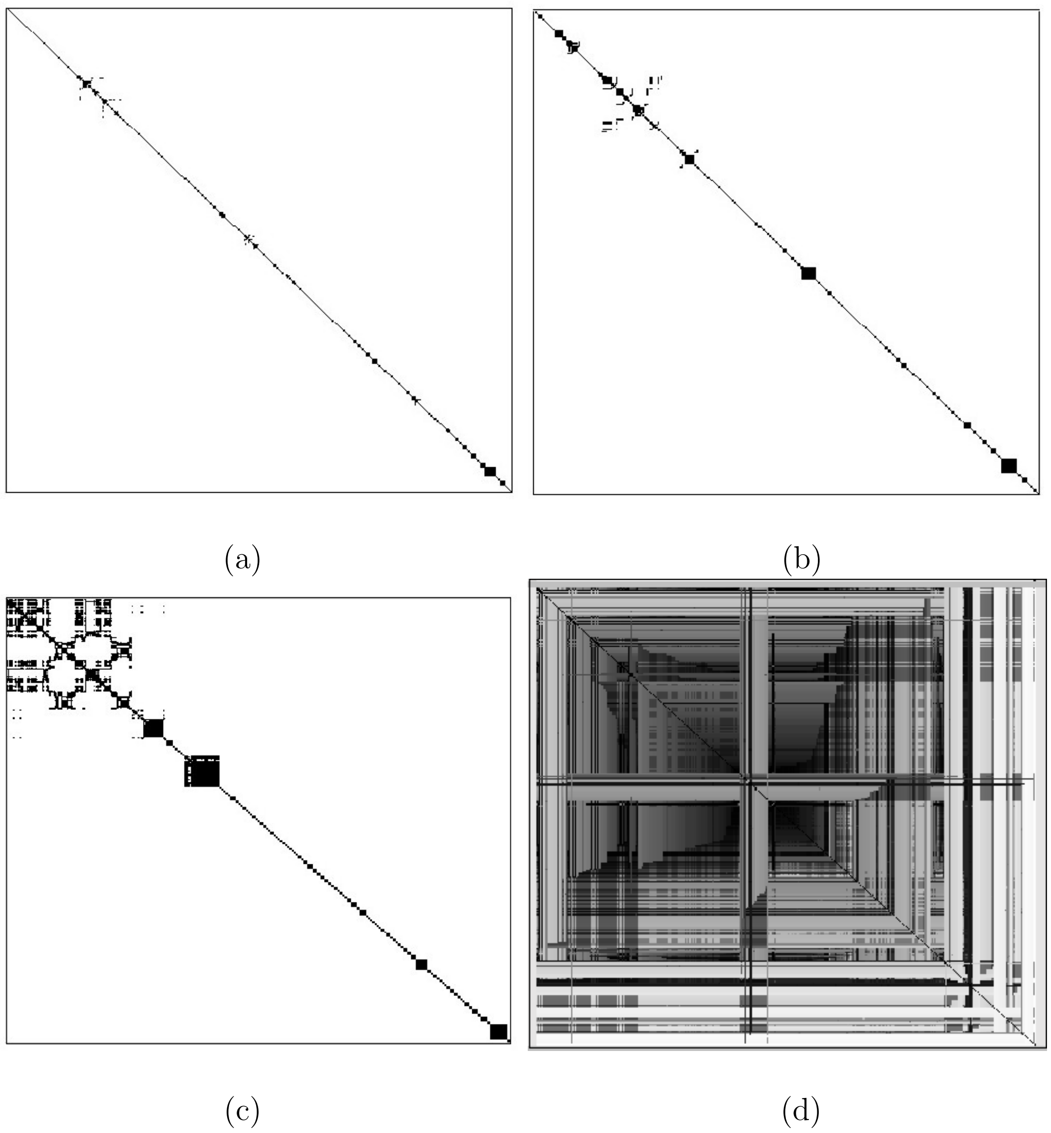


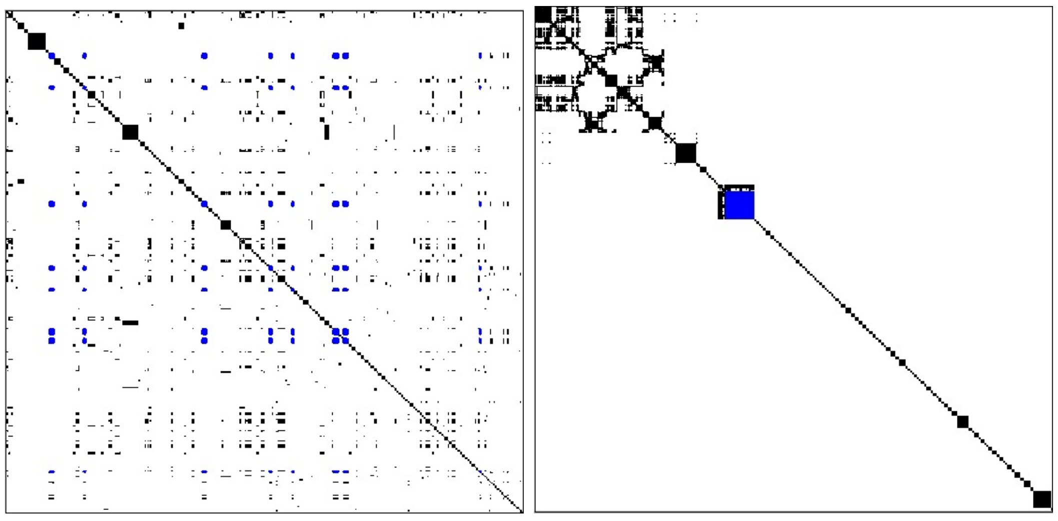
Figure 3.3: For the social network of coauthors in [42], the adjacency matrix (as shown in Figure 3.1 (b)) transforms on application of the following similarity functions and seriation using VAT: (a) Cocitation, (b) Bibcoupling, (c) k-ring neighbourhood for ($k = 2$), and (d) Role simialrity for ($\beta = 0.6$).

3.3 Permutation Algorithms

An algorithm for optimal ordering of entities in a two-way one-mode representation [38] as a square matrix, like the adjacency matrix and similarity matrix, is a permutation algorithm, also known as a seriation algorithm. Ordering a set of N vertices can be done in $N!$ different ways. Graph theory states that ordering of vertices to optimize a cost function is called minimum linear arrangement (MINLA/MinLA), which is a known NP-hard problem. Hence, one uses heuristic algorithms to solve ordering of vertices to achieve domain-specific optimization criteria. An ideal ordering algorithm should be linear or better, in terms of runtime complexity. The choice of the starting vertex is critical to most ordering algorithms. We assume node 0 as starting vertex and focus on the following eight permutation algorithms: VAT, reVAT, Breadth first search (BFS), Depth first search (DFS), Reverse Cuthill-Mckee (RCM) [7], Kings [44], Modified minimum degree (MMD) [41] and parallelized VAT (pVAT). Figure 3.4 shows the adjacency matrix of the co-authorship network [42] transforming after application of k-ring neighbourhood function and VAT seriation algorithm.

3.3.1 VAT

VAT [5] uses Prim's algorithm for finding the minimum spanning tree (MST) of a weighted graph to permute the order of objects to obtain a two-way one-mode representation [38], which is a square matrix with rows and columns in the same order, referring to the same entities. VAT shows clear clusters as black blocks along the diagonal when matrix is represented using a grayscale image, as shown in Figure 3.4(b). However VAT fails for cases where (a) there are no natural clusters in the data and



(a) (b)

Figure 3.4: For social network of coauthors data set [42], after applying similarity function of k-ring neighbourhood for ($k = 2$) on the original matrix in (a), the transformed matrix on application of VAT seriation algorithm is as shown in (b). A cluster of nodes identified after transformation is highlighted in blue in both representations.

(b) clusters cannot be identified if the cluster types can not be identified using single linkage.

The original VAT works only on symmetric matrices. Hence, our inputs are either undirected graph or symmetric adjacency matrix. VAT ordering and display algorithm is as given in Algorithm 1.

3.4 Graph Transformations

Our work focuses on the display of the similarity matrix as opposed to the original adjacency matrix, as we believe that post processing of the network by the application of a similarity function reveals more information of the network than that portrayed by the adjacency matrix alone. We assume the original graph to correspond to the adjacency matrix and the transformed graph to the similarity matrix. We also focus on series of similarity matrices which can be assumed to be a transformation on the input graph. Summarizing, we will be discussing the following kinds of graph transformations in our work:

- Transformation using similarity functions on the adjacency matrix: Similarity function should be such that it links data point. The notion of similarity depends on the kind of data set we are using. Though for social network data k-ring neighborhood function is good similarity measure, for several other data sets it does give any useful information. The transformation function that we apply on the adjacency matrix is not necessarily reversible in nature, hence we retain a copy of the adjacency matrix. We can see from the application the

Algorithm 1: Visual Assessment of cluster Tendency (VAT) Algorithm

```
Input  : Dissimilarity matrix ($R$) of $n \times n$ where $n$ is the number of vertices.
Output: VAT ordering
$K = \{1, 2, \ldots, n\}$; I = J = $\emptyset$;
P[0] = (0,...,0)
double max = 0; int iForMax;
for i= 1 to n do
    for j =1 to i do
        if R[i][j] > max then
            max = $R$[i][j];
            iForMax = i;
P[0] = i;
I = {i};
J = K - {i};
minj= 0;
for r = 2 to n do
    for i belongs to each element of I do
        for j belongs to each element of J do
            find minimum $R_{ij}$ and store value j in minj;
    P(r) = j
    I← I∪ {j}
    J← J-{j}
```

original graph, transformation of graph and retrace where the element are in transformed graph.

- Transformations inherent in a series of similarity matrices: This kind of transformation can be found in time series data, application of various similarity functions or permutation functions on the same data, or different subspace clustering in multi-attribute data. For time series data, we can see temporal evolution of a network. For multi-attribute data, we can compute various subspace clusterings, i.e. selecting a subset of attributes and studying their relationships. Those attributes which do not comply with the general behavior of data set is removed as noise or exception, those attribute are known as *outliers*. The series of matrices computed using various similarity functions can show the robustness of these functions and also the conditions under which two objects are in a cluster; e.g., two objects which are very similar might not be affected by the change in similarity function and will feature in the same cluster for most similarity functions.

3.5 Discussions

Graph layouts are dependent on the properties of the data set. We briefly describe here how certain graph layouts are the best for chosen data. VAT algorithm can be treated as a single-linkage algorithm which is probably the reason for it to give clear clusters. We briefly discuss the relationship between VAT and single-linkage algorithms here.

3.5.1 Graph Layouts

Spherical layout is not very helpful for large data set, edge occlusion makes it difficult to see any of the network's structure. Fruchterman Reingold layout generally gives good insight of network's structure. Circular layout is one of the layouts that rely on good ordering as discussed by Muelder et al. in [43].

3.5.2 VAT and Single-Linkage Algorithms

When an algorithm uses the minimum distance $d_{min}(C_i,C_j)$ to measure the distance between clusters, it is called a *nearest-neighbour clustering algorithm.* If the clustering process is terminated when the distance between nearest clusters exceed an arbitrary threshold, it is called a *single-linkage algorithm.* If we view the data points as nodes of a graph with edges forming a path between the nodes in a cluster, then the merging of two cluster, C_i and C_j, corresponds to adding an edge between the nearest pair of nodes in C_i and C_j. As edges linking cluster always go between distinct clusters, the resulting graph will generate a tree. Thus, an agglomerative hierarchical clustering algorithm that uses the minimum distance measure is also a *minimal spanning tree algorithm.*

In [22], Havens et al. have discussed about VAT being similar to single-linkage algorithm owing to the requirement of construction of minimum spanning tree (MST) for clustering. VAT reorders the relational data with a modified Prim's algorithm that is used for constructing a MST. We have leveraged the relation of the MST to both VAT and single-linkage clustering to prove that single-linkage clustering always produces aligned clusters of the VAT-reordered data for the case of a unique MST. In cases of data that may have more than one MST, single-linkage will still, in practice, produce aligned clusters of VAT-reordered data. If both VAT and single-linkage are initialized so that they lead to the same MST, this analysis applies.

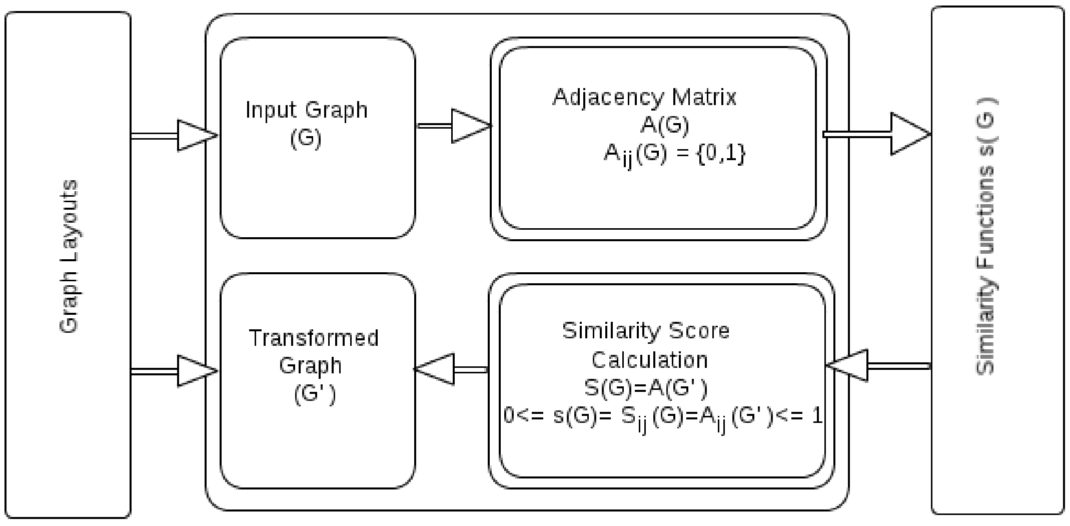
Chapter 4

Linked Views

Our work explores linked views of matrix and graph visualization of a given network. Our goal is to use a visualization tool with linked views to find patterns which will help us in understanding clustering structures in the data. Our tool has helped us in determining the effectiveness of various seriation algorithms.

4.1 System Architecture

We showcase how application of similarity matrix on a network can be considered to be a graph transformation. As shown in the schematic diagram of our system in 4.1, we apply a transformation on the input graph G represented by its adjacency matrix A(G). When a similarity function s is applied to A(G) which gives a similarity matrix S(G). S(G) is a two-way one-mode representation which can be treated as an adjacency matrix to a weighted graph, G' i.e., S(G) = A(G'). Thus, s can be considered to be a transformation on G, and hence we refer to G' as the transformed graph and A(G') as the transformed matrix. Our tool uses synchronized node-link



Graph Layouts
Input Graph (G)
Adjacency Matrix A(G) A_ij(G) = {0,1}
Similarity Functions s(G)
Transformed Graph (G')
Similarity Score Calculation S(G)=A(G') 0<= s(G)= S_ij (G)=A_ij (G')<= 1

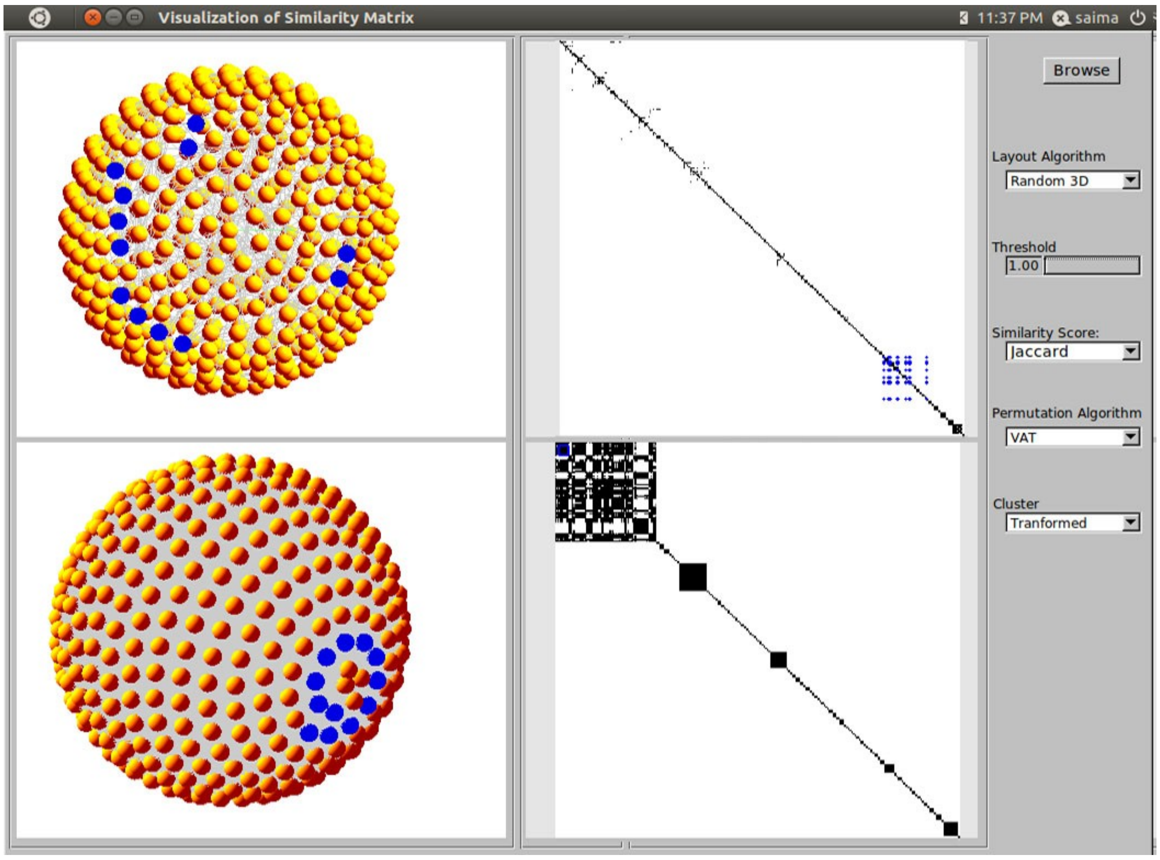
Figure 4.1: Schematic diagram of our visualization system. The input graph G and its adjacency matrix A(G) are used to find the similarity matrix S(G). A_{ij} are the element of matrix A(G). s(G) is the similarity function applied on the graph. We consider S(G) to be equivalent to an adjacency matrix of a transformed weighted graph G′, thus S(G) = A(G′). S_{ij} is an element of the similarity matrix S. The graph layouts are applied on G and G′, the transformation functions on A(G), and seriation or permutation algorithms on A(G) and A(G′) which is shown by a double box in the schematic diagram.

diagram and matrix visualization to view G and G'. When $\mathrm{A}(G)$ is not a linear function of $\mathrm{A}(G')$, one can see a lot of variations in the views of G and G', which can give a different perspective of the graph. Various graph layouts can be applied on G and G', and permutation algorithms can be applied on $\mathrm{A}(G)$ and $\mathrm{A}(G')$. To study the clustering capability of the similarity functions, we visually assess the clusters that are formed as blocks on the diagonal, as done in VAT.

4.2 User Interface

We have built a visualization tool complying with the architecture specified in Section 4.1. As shown in Figure 4.2, the user interface of our visualization tool consists of four display panels and a control panel. The top two display panels are for the input graph and the bottom two are for the transformed graph. The left panels show the graph layout and the right, the matrix visualizations. The control panel allows users to browse files to input data, choose graph layout, similarity and permutation functions, and clustering matrix i.e., adjacency or transformed matrix.

The user interface has a feature to hop from cluster to cluster and to select specific clusters or elements in the matrix. When a cluster is selected in any of the matrices, the four panels are synchronized to track the elements of the cluster. A cluster chosen in the adjacency matrix shows elements highlighted in green in all four panels, while one chosen in the transformed matrix uses blue highlight, as shown in Figures 4.5, 4.6, and 4.8. While all four panels are provided with zooming and translation controls, rotation is restricted to the graph layouts. The two highlight colors indicate where the cluster was located, i.e., in the adjacency matrix or the similarity matrix. Though only our tool enables only a single select-and-highlight operation across all four windows, we have provided the two highlights for the user to identify in a glance if the selected cluster was formed before or after transformation. This feature has helped us to find the location of the cluster in instances where nearly similar clusters occur before and after transformation.



Layout Algorithm: Random 3D

Threshold: 1.00

Similarity Score: Jaccard

Permutation Algorithm: VAT

Cluster: Tranformed

Figure 4.2: User interface of our tool shows four display panels and a control panel. The top panels show the graph layout and its adjacency matrix of the input graph (G) and bottom panels show the same for the transformed graph (G'). The control panel on the right,as shown in the inset, allows users to browse data sets and choose graph layout, seriation algorithm, and similarity function.

4.2.1 Locating Clusters

Since clustering is a process of our interest, we enable accessing clusters in both the adjacency and the transformed matrices. After seriation algorithm is applied on the matrix, we navigate through the clusters formed as blocks along the diagonal using Algorithm 2. Worst case complexity of this algorithm is $\theta(N^2)$ for N nodes in the graph.

4.3 Experiments and Results

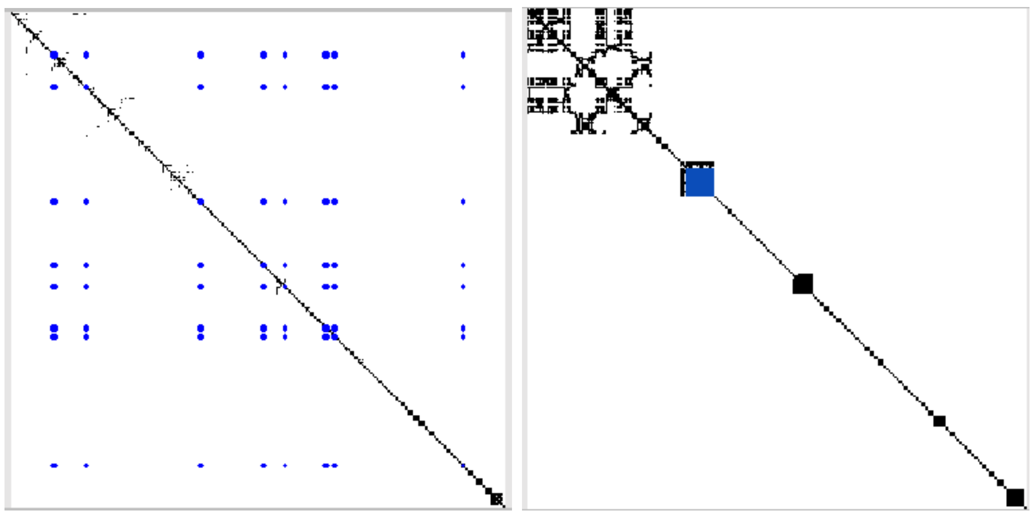
We have used our visualization tool to study the representation of the coauthorship network data [42] using various graph layouts, similarity functions, and permutation algorithms. The results shown in Figures 3.2, 3.3, and 4.3-4.6 show how useful our tool is for comparing various settings, namely, choice of seriation algorithm, similarity functions, and the graph layouts. These figures show the clustering effects of similarity functions for the social network of coauthors [42].

Figures 3.2 and 3.3 show how the matrix visualization looks before and after transformation when using various similarity functions.

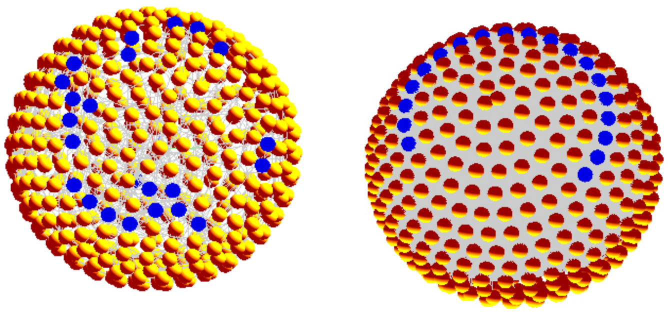
Using the similarity function to be k-ring neighbourhood function for $k = 2$, we have isolated a specific cluster and shown how the graph layouts look before and after the transformation in Figures 4.3 and 4.4. These figures show the adjacency and similarity matrices, where the latter are constructed using k-ring neighbourhood as similarity function and VAT as seriation algorithm. The nodes belonging to the selected cluster in similarity matrix are highlighted. We can see that nodes are together in different layouts corresponding to similarity matrix.

Algorithm 2: Algorithm for locating clusters

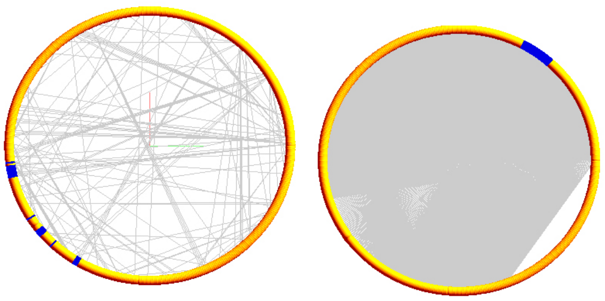
```
Input   : n×n matrix
Output: Total number of clusters with their starting and ending points
int startx = 0, starty =0, clustercount=0, check=0, x, y, cluster[n][2]
// set the start of first cluster[clustercount][0]=0
while ( last block (n, n) is not reached startx != n and starty != n ) do
    move to block below current block tempx=startx+1 and tempy=starty
    check = 0 ;
    while (x > y ) do
        // check if the block above the current block cannot be clustered
        if arrangedMatrix[x][y]-arrangedMatrix[x-1][y] ≤ allowedDifference
        then
            set check=1
            increment y by 1
    if (check == 0) then
        // set the ending point of the cluster
        cluster[clustercount][1] = x
        startx++
    else
        // set the ending point of the cluster
        cluster[clustercount][1] = x-1;
        startx++;
        // reset the starting point
        starty = startx;
        clustercount++;
```



(a)

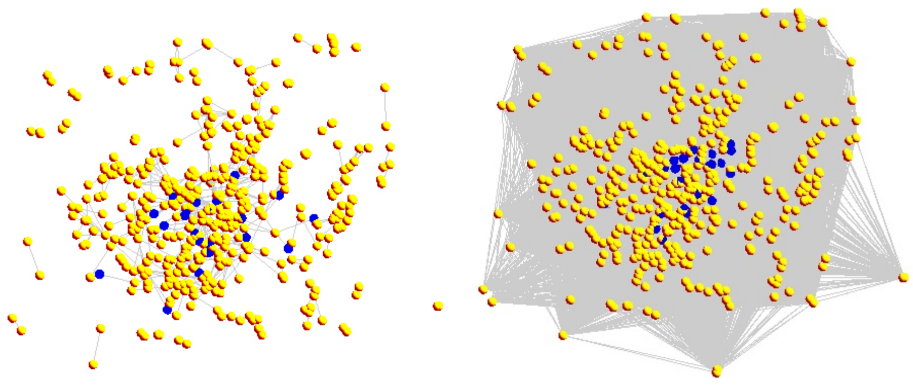


(b)

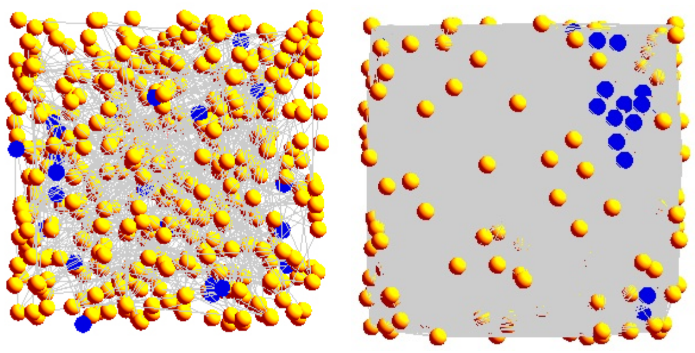


(c)

Figure 4.3: Using k-ring neighbourhood function, for ($k = 2$) for transformation of network of coauthors [42] and seriation using VAT, we see: (a) matrix visualization, and graph layout using (b) sphere, and (c) circular; where the left and right images show before (G) and after (G′) transformation, respectively. Blue blocks in the matrices and blue nodes in the graph layouts show how we can track the elements of a cluster identified after transformation, shown in the inset in (a).



(a)



(b)

Figure 4.4: Using k-ring neighbourhood function, for $(k = 2)$ for transformation of network of coauthors [42] and seriation using VAT, we see: (a) Fruchterman Reingold, and (b) random layouts; where the left and right images show before (G) and after (G′) transformation, respectively. Blue blocks in the matrices and blue nodes in the graph layouts show how we can track the elements of a cluster identified after transformation, shown in the inset in Figure 4.3(a).

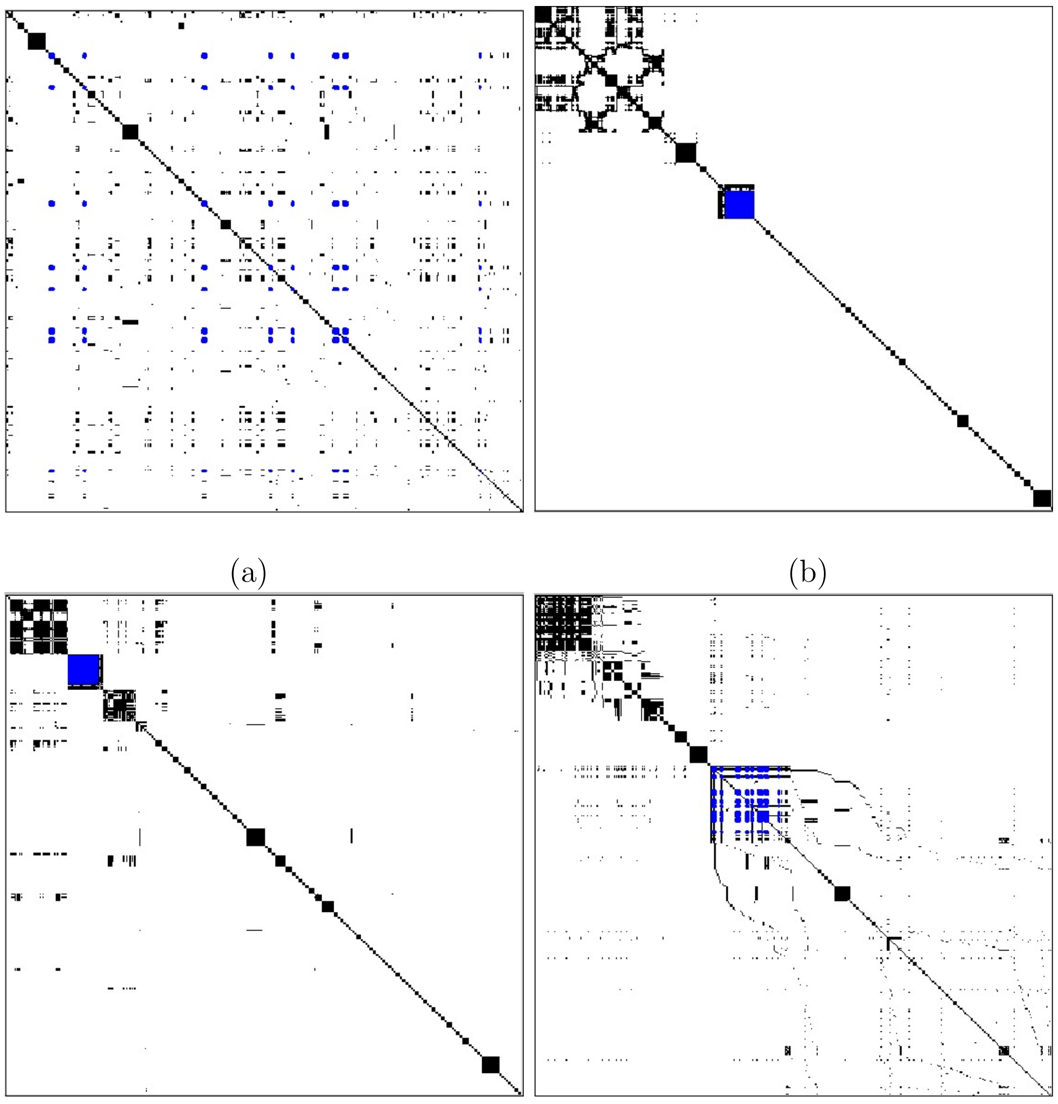

(a) (b) (c) (d)

Figure 4.5: For social network of coauthors data set [42], after applying similarity function of k-ring neighbourhood for ($k = 2$) on the original matrix in (a), the transformed matrices show differences in the visual representation of clusters on applying the following permutation algorithms: (b) VAT, (c) reVAT, and (d) BFS.

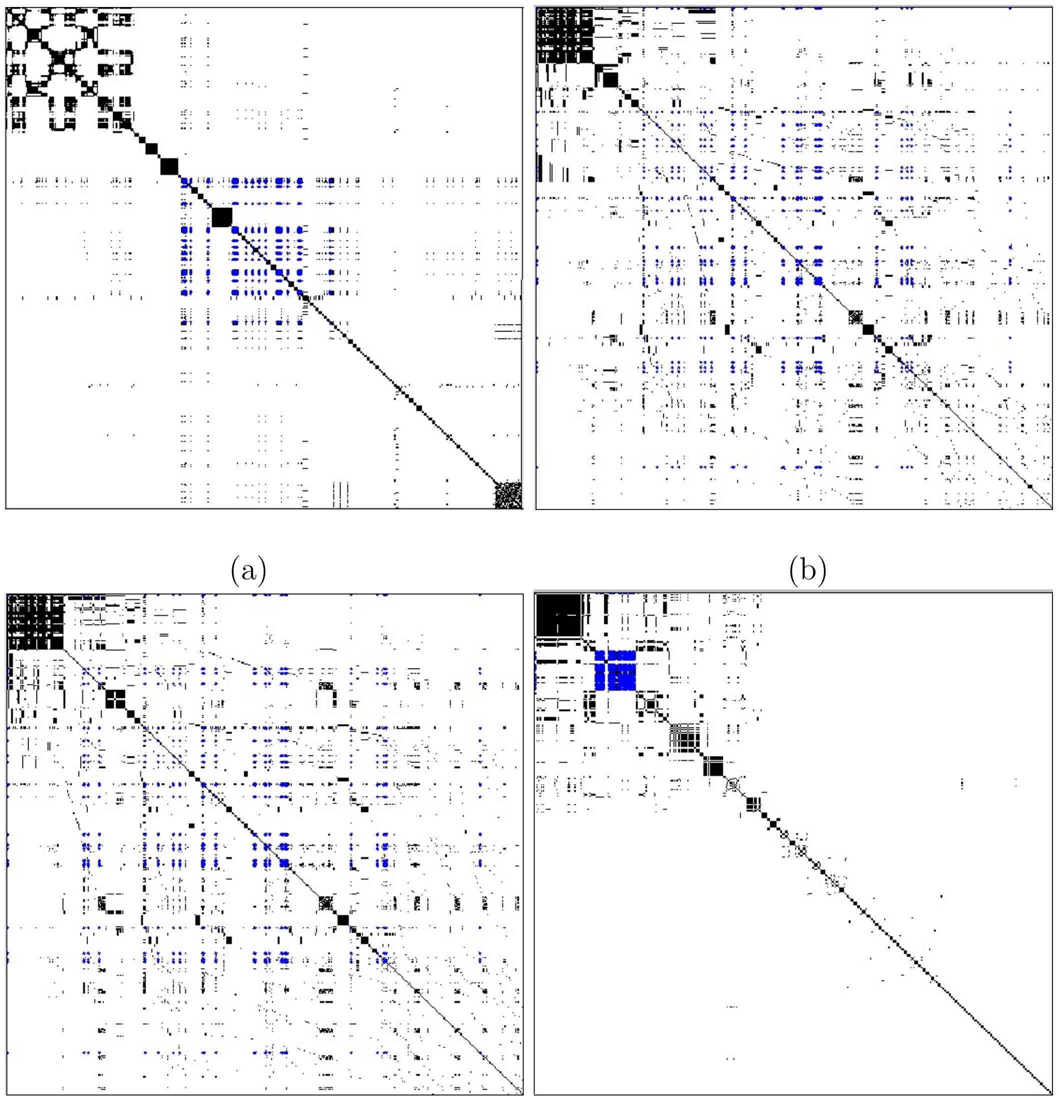


Figure 4.6: For social network of coauthors data set [42], after applying similarity function of k-ring neighbourhood for ($k = 2$) on the original matrix in Figure 4.5(a), the transformed matrices show differences in the visual representation of clusters on applying the following permutation algorithms: (a) DFS, (b) RCM, (c) Kings, and (d) MMD.

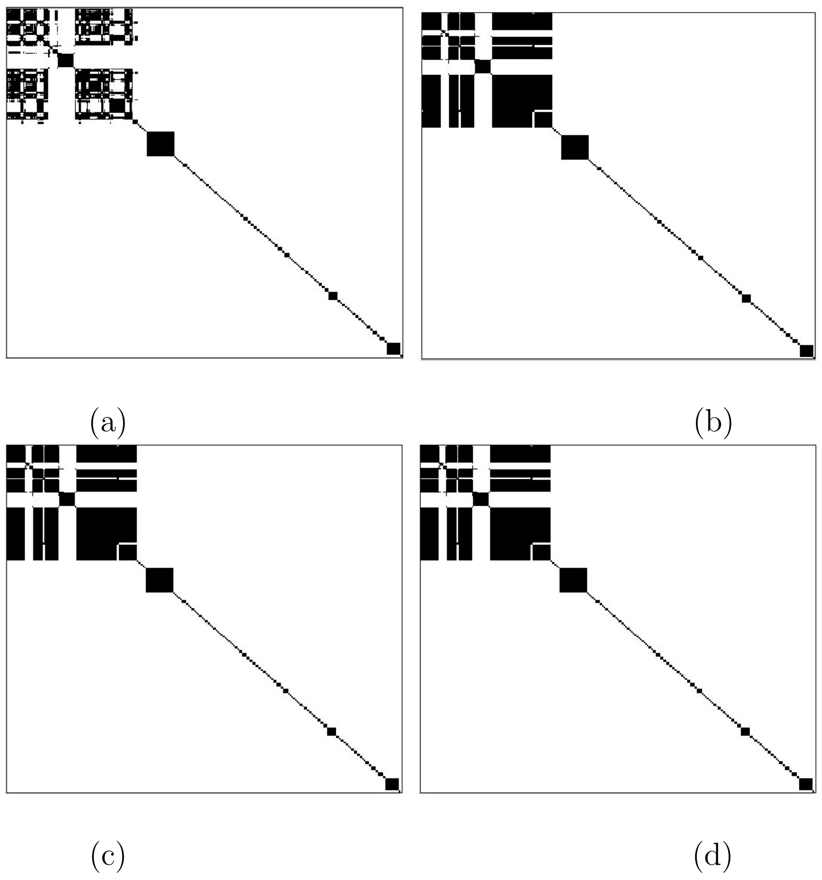
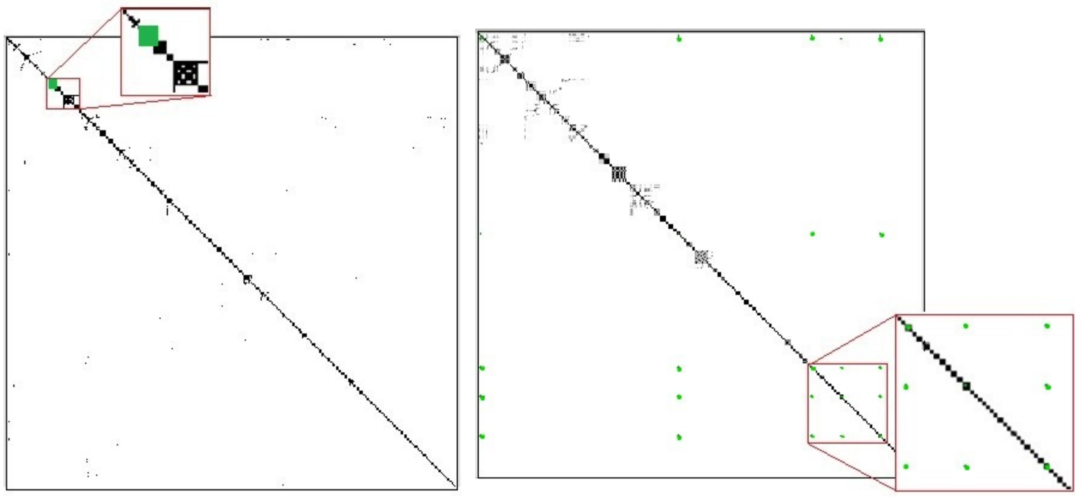

(a) (b) (c) (d)

Figure 4.7: For social network of coauthors data set [42], on applying k-ring neighbourhood function and seriation using VAT, the transformed matrices for value of $k =$ (a) 3, (b) 5, (c) 6, and (d) 10, show that clustering capability of the similarity function changes. No changes after $k = 5$ reflect the theory on six degrees of separation.



(a) (b)

Figure 4.8: For the peer-to-peer file sharing network data set [34], we can see here how a cluster seen in the original matrix, identified after using VAT in (a), moves after applying dice similarity function and subsequent seriation using VAT in (b).

Figures 4.5 and 4.6 show the effect of applying different permutation algorithm on the coauthorship network data set [42] using k-ring neighbourhood for ($k = 2$) as similarity function. In Figures 4.5 and 4.6, we have highlighted the nodes forming a cluster in 4.5(b). In Figure 4.5(a) which shows the adjacency matrix itself, we can see that the nodes are scattered while in Figure 4.5(c) they can be seen in physical proximity, forming a cluster. In Figures 4.6(d) and 4.5(d), though the highlighted nodes are not forming cluster still they are found to be close to each other after matrix reordering. Thus we can see from Figures 4.5 and 4.6 that VAT brings together all the nodes at the diagonal and maximizes size of the clusters. From the above examples, by visual assessment we can conclude that VAT performs better than the other seriation algorithms with respect to partitioning the matrix in terms of clusters.

Figure 4.7 shows how we can visually assess the clustering capability of the similarity functions. On applying the k-ring neighbourhood function on the data set of social network of coauthors [42], Figure 4.7 shows how varying k gives different clustering results. We have observed that increasing value of k after 6 does not show differences in the clustering pattern, which reflects the theory on six degrees of separation (which corresponds to the function when $k = 6$). Six degrees of separation is the theory that everyone and everything is six or fewer steps away, by way of introduction, from any other person in the world, so that a chain of "a friend of a friend" statements can be made to connect any two people in a maximum of six steps.

Just as we can track a cluster after transformation, we can also track how elements in a cluster seen before transformation move after transformation, as shown in Figure 4.8, for data set on peer-to-peer file sharing network [34].

4.4 Discussions

In this section we briefly discuss a form of validation for the clustering provided by seriation algorithms, comparison of seriation algorithms, on validation of graph layouts, and the effectiveness of our proposed tool.

4.4.1 Clustering Results from Seriation Algorithms

In the absence of a generic analytical metric to evaluate the correctness of the visual assessment of clustering after transformation in our tool, we have used a metric called "clustering coefficient" specifically for the application of the k-ring neighborhood similarity function, as shown in Table 4.1. A small world network can be quantified using a "clustering coefficient" $C(p)$ [4] for varying k in k-ring neighbourhood, as it measures the "cliqueness" in the network. If c_i is the number of neighbours of a vertex i, there are a priori $c_i\frac{(c_i-1)}{2}$ possible links between these neighbours. C_i the fraction of these links that are really present in the graph, $C(p)$ is the average of C_i over all vertices. Using varying k in k-ring neighborhood function to show different similarity functions, we have evaluated for the social network of coauthors [42], the trend in the number of clusters seen in the visualization of transformed matrix after permutation. We see that our visual assessment follows the same trend as the clustering coefficient.

k	2	3	4	5	6	10
C	4.07	3.42	3.28	3.16	3.09	2.97
n(VAT)	127	117	107	104	104	104
n(reVAT)	129	114	107	104	104	104
n(DFS)	132	118	107	104	104	104
n(BFS)	139	123	107	104	104	104
n(Kings)	194	176	160	157	157	157
n(RCM)	186	176	160	157	157	157
n(MMD)	193	149	141	140	140	140

Table 4.1: Using k-ring neighborhood function on social network of coauthors [42] to compute clustering coefficient C, and check its trend against the number of clusters ($n(P)$) automatically counted assessed on using various permutation algorithms (P) on the transformed matrix.

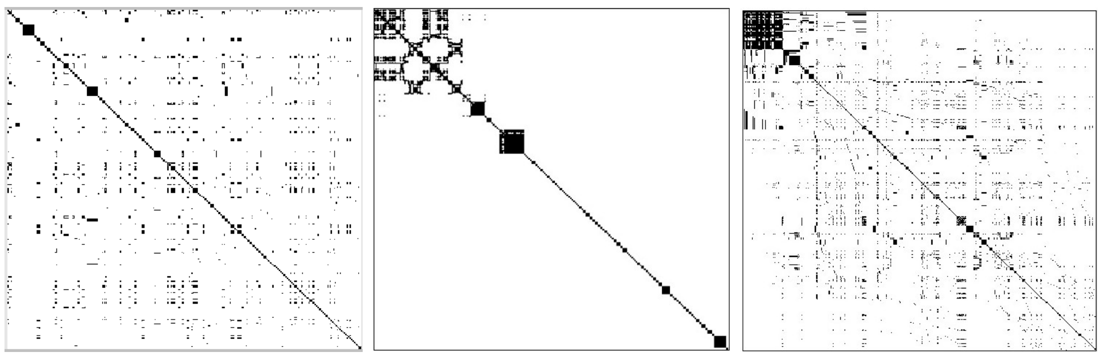
4.4.2 Comparison of Seriation Algorithms

We have compared the performance of several standard techniques used in visualization community with the VAT algorithm which is widely used in the pattern recognition community. We have found by visual assessment that VAT performs very well in terms of identifying clusters for our experiments in comparison to other permutation algorithms we have included in our tool. Hence in the remainder of our work, we will be using VAT as the default seriation algorithm and used it for further analysis.

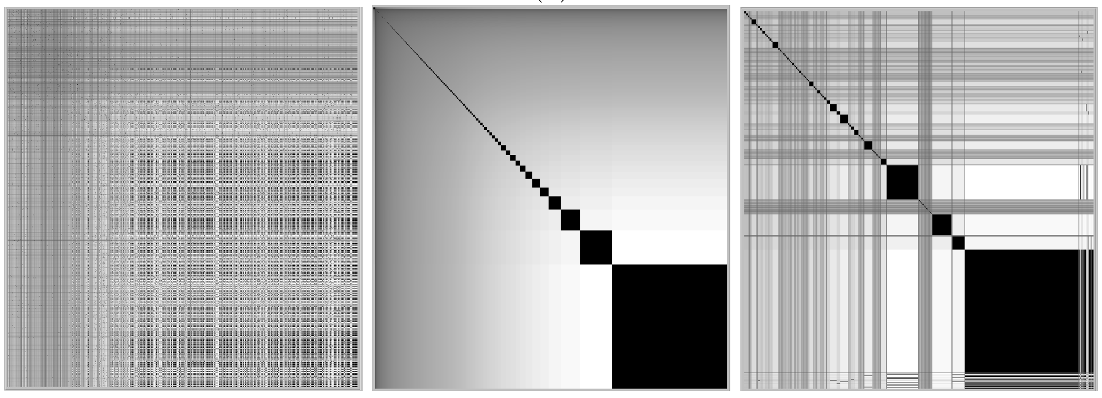
To show an example of VAT's effectiveness, Figure 4.9 shows clear clusters when using VAT over RCM seriation algorithms for various datasets: social network of coauthorship [42], Wikipedia's who-votes-on-whom data set [37] and collaboration network of jazz bands [17].

4.4.2.1 Comparison of VAT with nearest neighbor traveling salesman problem ordering

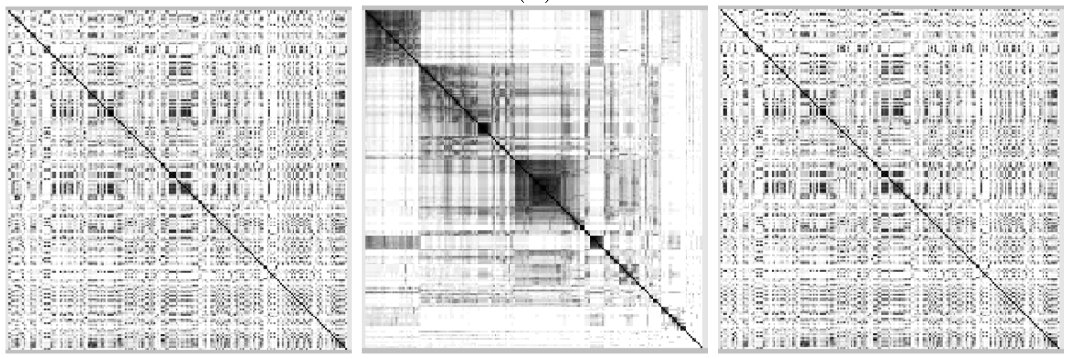
Figure 4.10 shows the comparison of VAT with one of the most popular reordering/seriation algorithm used in the information visualization community currently, i.e., the one based on the nearest neighbor traveling salesman problem (NNTSP), as shown in the case of ZAME [11]. On applying the k-ring neighborhood similarity function, for ($k = 2$), to the social network of coauthors data set [42], we can see from Figure 4.10 that VAT performs better than NNTSP in partitioning the matrix with respect to identifying clusters.



(a)

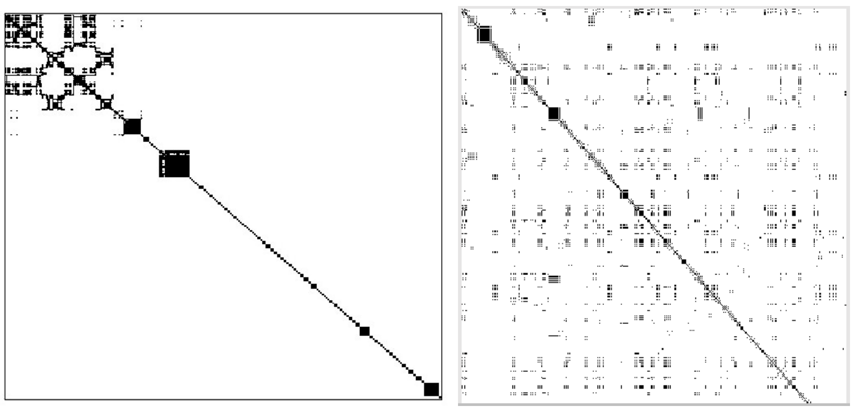


(b)



(c)

Figure 4.9: Original similarity matrix, with VAT and RCM reorderings in the left, middle, and right columns, respectively for: (a) social network coaurthor data set [42] using 2-ring neighborhood similarity function; (b) Wikipedia's who-votes-on-whom network [37] using 6-ring neighborhood and Jaccard similarity functions; (c) collaboration network between jazz bands [17] using dice similarity function.



(a) (b)

Figure 4.10: For social network of coauthors data set [42], after applying similarity function of k-ring neighbourhood for $(k = 2)$ on the original matrix in (a) VAT, and (c) NNTSP.

4.4.3 On Graph Layouts

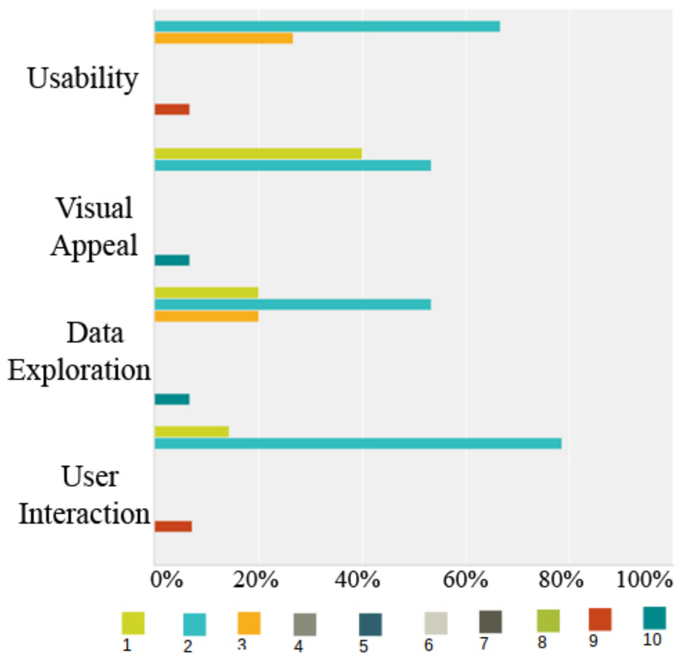
As discussed in Section 3.5.1, we have found for the coauthorship network data [42] that the spherical layout gives the most coherent representation for ordered data in comparison to the other layouts we have considered.

4.4.4 Effectiveness of Proposed Tool

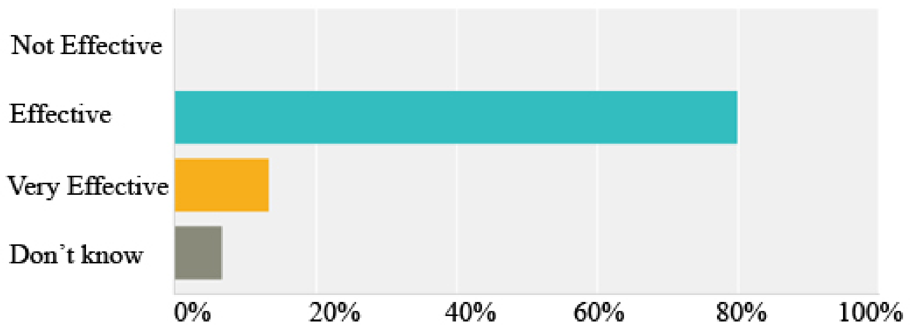
In terms of layout, our tool may seem equivalent to two instances of an existing application of linked views of matrix representation and node-link diagrams applied to the adjacency and the transformed graphs of a network. However our tool provides additional value in the synchronization of the two linked views. e.g., our tool can navigate through clusters in the transformed matrix or adjacency matrix and simultaneously locate the entities in the cluster before and after transformation in the node-link diagrams as well as in the matrix views. The synchronization of all the four display panels help in tracking elements within or outside a cluster which makes our tool useful for visual analytics.

We have performed an informal user study on our tool and collected responses from fifteen users. The participants of the user study were students of masters' degree in Information Technology at IIIT-Bangalore. Before the questionnaire was given to them, they were given a brief introductory talk on clustering and the different clustering techniques discussed in the thesis. The users were then provided with the software and were asked to perform certain operations and their observations were recorded. The operations included opening a file, choosing specific similarity function, permutation functions and graph layouts to visualize the data, and track specific clusters. The users were given 30 minutes to use the tool before answering the questionnaire.

The users have confirmed that matrix based visualization is a better representation technique when data set is large. Figure 4.11 shows a summary of the user study which includes the users' perception about the effectiveness of our tool in visually assessing clustering capability of similarity functions.



(a)



(b)

Figure 4.11: Survey analysis showing: (a) users' rating of our tool on the basis of usability, visual appeal, data exploration and user interaction (b) how effective our tool is in visually assessing clustering capability of similarity functions.

Chapter 5

Improving Performance

We have compared the performance of several standard techniques used in visualization community with the VAT algorithm [5], which is widely used in the pattern recognition community. As shown in Chapter 4, we have found that VAT performs very well in terms of identifying clusters for our experiments. However its runtime complexity of O(N^2) for a matrix with N nodes makes it unsuitable for large networks. We propose parallelized VAT (pVAT), i.e., using CUDA for parallelizing VAT.

5.1 pVAT

Since VAT has a runtime complexity of $O(N^2)$ for N nodes in the graph, it is not scalable for large data sets. Though bigVAT [28] and sVAT [21] are scalable, they rely on good choices of representative data to downsample the given graph. Hence we propose the parallel implementation of VAT (pVAT).

VAT is based on Prim's algorithm for finding the permuted order of elements in the similarity matrix. We propose using Borůvka's algorithm for finding the minimum spanning tree (MST) as implemented by Vineet et al. [53]. In the serial implementation of VAT, Prim's algorithm is used for constructing MST however using Borůvka, we exclusively order the nodes without constructing the actual MST or evaluating the feasibility of constructing one. In a disconnected graph, a MST cannot be constructed but we can still get an ordering and construct the ordered image in VAT, which is an advantage of our approach. Thus Prim's and Borůvka's algorithms will give us different orderings in disconnected graphs. Borůvka's approach is generally favored in parallel implementation owing to its recursive nature. The workflow for pVAT is as shown in Figure 5.1. Our proposed algorithm for pVAT is as given in Algorithm 3.

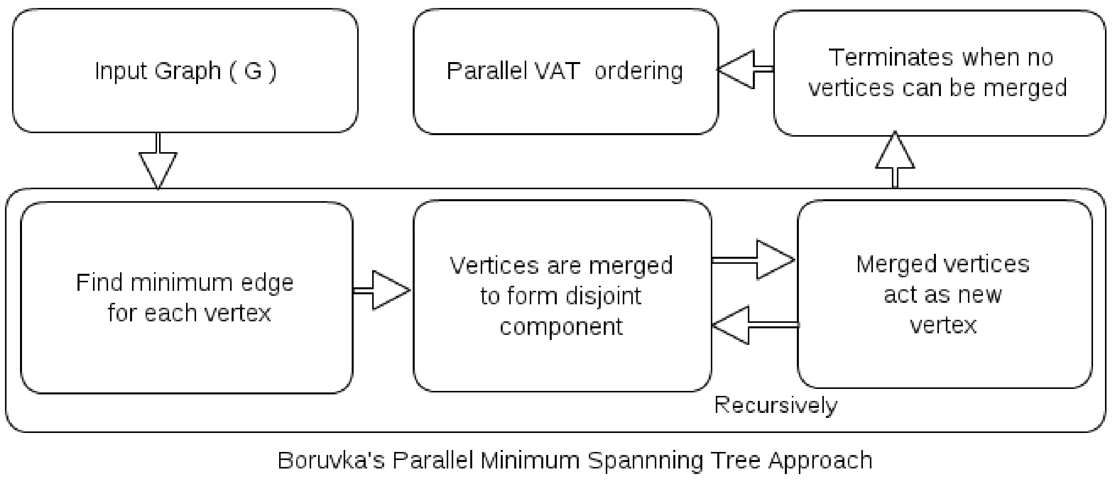

Input Graph (G)
Parallel VAT ordering
Terminates when no vertices can be merged
Find minimum edge for each vertex
Vertices are merged to form disjoint component
Merged vertices act as new vertex
Recursively
Boruvka's Parallel Minimum Spannning Tree Approach

Figure 5.1: Schematic diagram of parallelized VAT (pVAT) algorithm.

5.2 Experiments and Results

We have used the following data sets: coauthorship network [42], snapshots of peer-to-peer file sharing network [34, 35], who-votes-on-whom network data [37], and coc-

Algorithm 3: pVAT: Parallel implementation of VAT

```
Input  : Graph (G) with $n$ nodes in set $V$ and $e$ edges
Output: Parallel VAT ordering
for each vertex belong to $V$ do
    find the minimum edge
while No more vertices can be merged do
    vertices are merged to form disjoint component
    merged vertices act as new vertices
get the ordering of graph
```

itation network [36]. Implementation of pVAT algorithm requires CUDA [49] and CUDPP [20] libraries. We have performed the experiments using Nvidia GeForce GTX 480 and 4GB RAM.

Performance measurements comparing pVAT to the serial implementation of VAT on different data sets are as given in Table 5.1, which shows as expected that the former is far more efficient than the latter.

5.3 Discussions

In the remainder of our work, we have used the serial or the parallel implementation of VAT as deemed efficient for the size of the concerned data set.

Data set	#Nodes	#Edges	Serial VAT (msec)	pVAT (msec)
Coauthorship network [42]	475	1250	1x10^3	2.42
Peer-to-peer file sharing network [34]	6301	20777	1.8x10^6	4.55
Who-votes-on-whom network [37]	7115	103689	2.4x10^6	5.21
Peer-to-peer file sharing network [35]	8114	26013	3.8x10^6	4.07
Cocitation network [36]	9877	51971	6.6x10^6	3.92

Table 5.1: Performance measurements of serial and parallel implementations of VAT for various network data sets.

Chapter 6

Data Simplification

Data simplification implies elimination of unnecessary and redundant data. We have used clustering of data into groups for simplifying data. Hierarchical clustering groups objects like a tree data structure, where the root is a superset and the leaf nodes are the minimal subset i.e., data objects. Hierarchical clustering can be further categorized as agglomerative or divisive.

6.1 Multilevel Clustering

In our work, we have used agglomerative hierarchical clustering for achieving multiple levels of detail. It works in bottom-up manner, initiliazed by all data objects as leaf nodes and each single data object node belonging to a cluster node. For implementing multilevel clustering, we merge all the nodes in a cluster into a single node when moving from a finer to a coarser level of detail and recursively apply clustering algorithm on the new set of nodes. Suppose we have n nodes $(x_1, x_2, \ldots, x_n)$ and after applying similarity function and permutation algorithm, we get k clusters

$(c_1, c_2, \ldots, c_k)$ where $k < n$. In general, the condition $k \neq n$ should be held because $k = n$ implies that every cluster contains a single node, which in turn implies that there are no inherent clusters in the data set. This condition helps us to ensure that every cluster has a subset of nodes which will be merged in the next level of detail to form a single node. Suppose clusters c_1 and c_2 have the following constitution in level 1: $c_1 = \{x_1, x_5, x_7\}, c_2 = \{x_2, x_3, x_4, x_9\}$, in level 2, the nodes get merged and c_1 and c_2 become the new nodes.

When moving to a coarser level of detail, the merged nodes replace the constituent nodes. For sake of simplicity, we will assume clusters to contain one or more nodes. Thus all the nodes in the subsequent coarser level of detail can be assumed to be merged nodes. When moving to a coarser level of detail, the adjacency matrix changes, leading to the requirement of recomputation of the similarity matrix. In our work, we have not derived an appropriate formulation for the attributes of a merged node which has to be derived from an appropriate aggregation of the attributes of the constituent nodes from the finer level. Hence we use a simple similarity function of finding the maximum weighted edge between the clusters. This function reduces a $n \times n$ matrix to a $k \times k$ matrix, where n nodes have reduced to k clusters across one level of detail. Owing to the absence of an appropriate aggregation of the attributes, for multilevel clustering we will not be using the similarity functions offered in our tool.

We terminate the multilevel clustering algorithm when none of the nodes can be further merged or there are no more clusters. At the coarsest level of detail, the similarity matrix is the same as the adjacency matrix, i.e., an identity transformation of the adjacency matrix.

Algorithm 4: Multilevel Clustering Algorithm

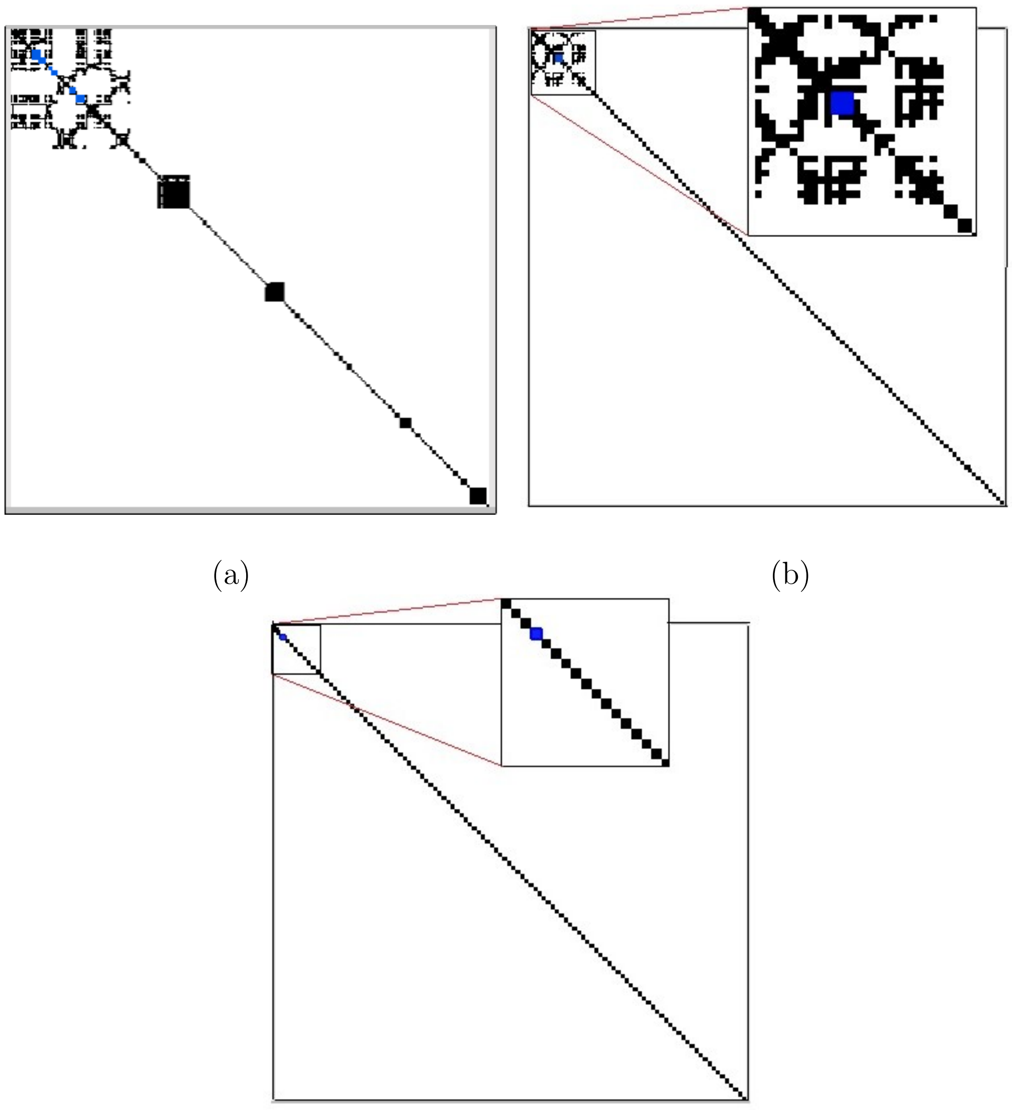
```
Input  : clusterSize number of cluster in after applying similarity function,
         VAT and clustering it.
         arrangedMatrix[n][n] is permuted matrix
         clusterx first element of every cluster
         clustery last element of every cluster
Output: arrangedMatrix[c][c] where c is the number of node in matrix after k
         levels of clustering
void multilevel()
for i=0 to clusterSize do
    for j= 0 to clusterSize do
        if i==j then
            groupedMat[i][j]=1;
        else
            groupedMat[i][j]= findSimilarity(i,j);
int findSimilarity(int cluster1 , int cluster2)
int max=0;
for i=clusterx[cluster1] to clustery[cluster1] do
    for j=clusterx[cluster2] to clustery[cluster2] do
        if arrangedMatrix[i][j] >max then
            max=arrangedMatrix[i][j];
return max;
if group button is pressed then
    VAT(groupedMat);
    assign groupedMat to arrangedMatrix
    multilevel();
```

6.2 Experiments and Results

We have used the following data sets: coauthorship network [42, 47], who-votes-on-whom network data [37], taro exchange network [19], and karate club network [61]. We have used the k-ring neighbourhood function for ($k = 2$) for computing similarity matrices and VAT as the seriation algorithm. At every transition in the level of detail, we perform the following steps: (a) compute the new similarity matrix, (b) merge nodes in cluster to form new nodes, and (c) apply VAT on the new similarity matrix. Table 6.1 summarizes the number of nodes in transitions between levels of detail when implementing multilevel clustering. Applying multilevel clustering for social network of coauthors data set [42], using recursive application of similarity function of k-ring neighbourhood for ($k = 2$) on the similarity matrix and VAT for seriation, we can see from Figure 6.1 that the number of nodes reduce from 475 to 133 to 115 across three levels finally giving 109 clusters. Figure 6.1 shows a how the nodes in a selected cluster in the coarsest level, shown by blue highlight, were placed in the finer levels.

6.3 Discussions

When using VAT as the seriation algorithm, any one of the minimum spanning tree amongst all possible ones is used which makes the results of our multilevel clustering combinatorial. The number of clusters for a particular combination of data set, similarity function and seriation algorithm may vary for all transitions to the coarser levels except the last one. The last transition gives the same result as multilevel


(a)

(b)

(c)

Figure 6.1: For social network of coauthors data set [42] of 475 nodes, recursively applying similarity function of k-ring neighbourhood for ($k = 2$) on the original matrix and using VAT for seriation gives (a) similarity matrix at finest level with 133 clusters. Transitioning by a level of detail, the similarity matrix has (b) 115 clusters, and (c) 109 clusters, which is the coarsest level. A cluster in the coarsest level of detail is tracked across the levels of detail using blue highlight.

Data set	#Nodes	#Nodes in Level Transitions	#Levels
[61]	22	22 →10→ 5→4	3
[19]	34	34 →7→2	2
[42]	475	475 →133 → 115 → 109	3
[47]	1589	1589 → 615 →238 →122 →113	5
[37]	7115	7115 →3216 →1983 →927 → 725 → 635	6

Table 6.1: Simplification of network data sets using multilevel clustering based on VAT algorithm for seriation and cluster identification after recursively applying 2-ring neighborhood similarity function.

clustering terminates when no further merges can occur, i.e., the similarity matrix in the coarsest level is an identity matrix of size u where u is the minimum number of nodes to which data set can be simplified. For example, in the coauthorship network data set [42], when using k-ring neighbourhood for ($k = 2$) as similarity function and VAT as seriation algorithm, more than one minimum spanning trees are possible as shown in Figure 6.2, showing two possible matrix visualizations for the same combination of data set [42], k-ring neighbourhood for ($k = 6$) as similarity function and VAT as seriation algorithm. Since the choice of MST determines the ordering, two different MSTs gives different number of clusters. Irrespective of the path of reduction of nodes in the multilevel clustering, we have observed from our experiments that the minimum number of nodes a data set can be reduced to in the coarsest level of detail is specific for a data set.

One application of multilevel clustering would be to efficiently distribute infor-

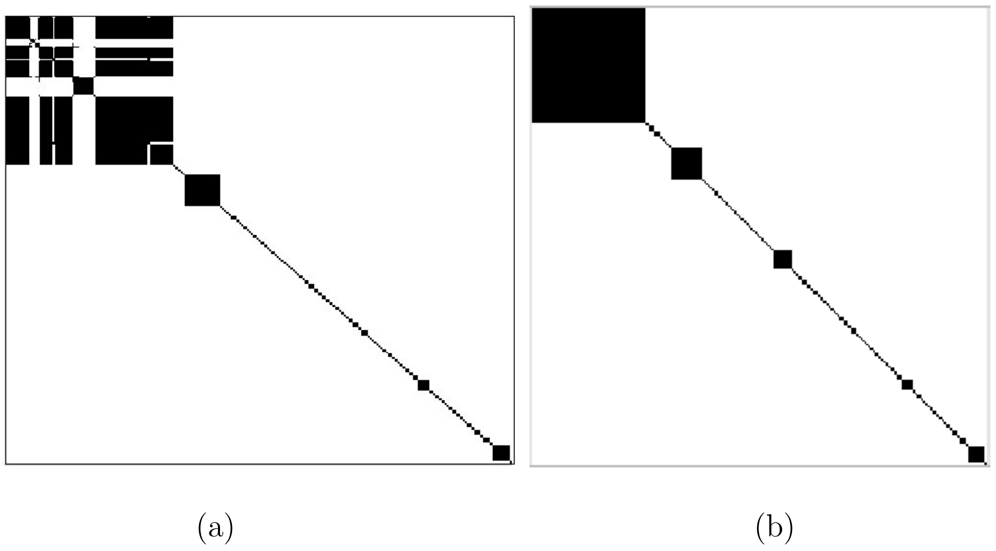


Figure 6.2: Two possible results after applying VAT as seriation algorithm on social network of coauthors data set [42] containing 475 nodes, using k-ring neighbourhood for $(k = 6)$ as similarity function on the original matrix. However, irrespective of the choice of minimum spanning trees during the transitions of levels of detail, the coarsest level of detail of this network always contains 109 nodes when using our approach.

mation or resources efficiently to all the nodes in a network. One can distribute the resource to all the nodes in the coarsest level, where each node can further pass it on to the cluster of nodes it represents at the subsequent finer level of detail. Thus the new nodes created by merging nodes in a cluster at a level of detail can be considered to be a good representative node for the entire set of nodes. For example, in the coauthorship network [42], we want to update the number of publications of each author in the network by a certain number, say, k, the operation can be performed top-down from the coarsest level to the finest level in a broadcasting fashion. Thus instead of distributing the function to all 475 nodes, a single distribution will probably have to communicate with at most 109 nodes, which is the number of nodes in the coarsest level of detail.

Chapter 7

Visualization of Similarity Matrix Series

Similarity matrix series refers to multiple similarity matrices obtained for a set of data objects or nodes, either in the form of time series, or by application of various similarity functions or permutation functions, or by generating different subspace clusterings. Membership of objects in clusters in each of these similarity matrices is a salient aspect of the series as well as of the entire data set. Tracking the cluster-membership of objects will enable us in identifying trends and patterns in the data. We propose parallel sets-like representation for tracking cluster-membership of objects across similarity matrices.

Figure 7.1 shows the parallel sets-like repesentation for the social network of coauthorship data set [42], where series of similarity matrices are obtained by applying different similarity function. The matrix representation for the series is shown in Figures 3.2 and 3.3.

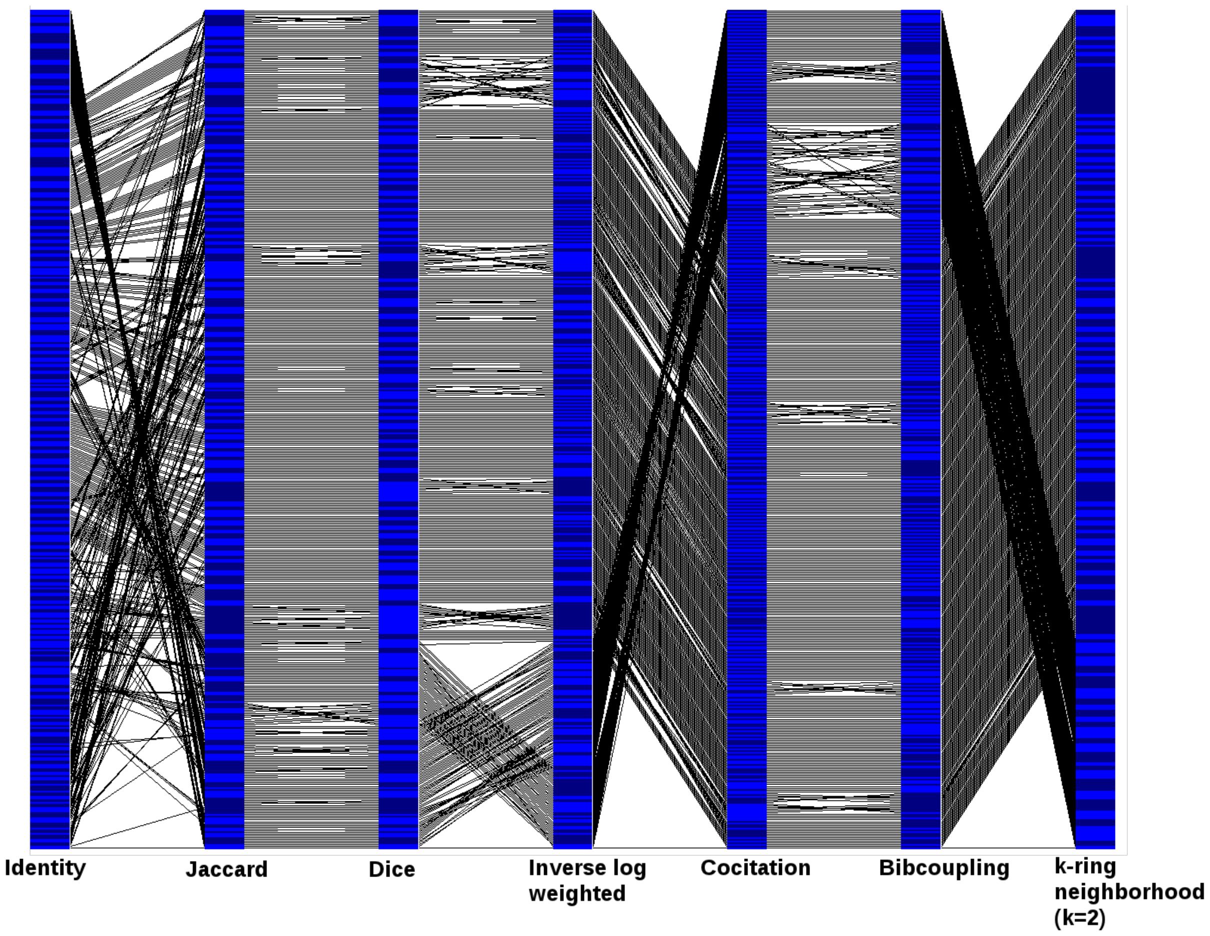


Figure 7.1: Parallel sets-like representation of similarity matrix series obtained by varying the similarity function on the social network of coauthorship data set [42] and using VAT for seriation.

7.1 Parallel Sets-like Representation

Visual exploration of multidimensional data is of great interest to the information visualization community, however visualizing it in a single visual space is a tedious process. Visual exploration of multidimensional data enables us to analyze trends, patterns, outliers and relationships among variables. Some of the constraints of the process are: limited screen space, not a single process to show all attributes simultaneously and insufficiency in the display of intuitive structures.

Combining cluster analysis and parallel sets-like representation makes the visualization system scalable. We display data objects in clusters in a single dimension on a line instead of data points. Objects are ordered in each attribute or timestamps using VAT to form clear clusters. While looking at the multiple similarity matrices in gray scale representation is tedious, we can look at a single snapshot of the series using a parallel sets-like representation for an overall analysis.

Kosara et al. have proposed parallel sets representation [33] to represent categorical data. Parallel sets has the following features: (a) it shares the parallel coordinate layout, treating all dimensions to be independent of each other, and (b) instead of using a line connecting values of coordinates to represent a data point, boxes are used to display categories and parallelograms between the axes to show the relations between categories. In our representation, the axes indicate the permuted order of objects in the seriated similarity matrix. We use blocks of axes to indicate clusters and lines between axes to indicate location of an object in the permuted order in the similarity matrix, similar to boxes and parallelograms in the parallel sets representation. Hence we refer to our technique as “parallel sets-like.”

In the parallel sets-like representation, the horizontal axis provides no specific information. Each vertical axis corresponds to a $n \times n$ similarity matrix in the series and displays the ordered set of n data objects or nodes. Each node has a label which can be tracked. Nodes are assigned color on the basis of the cluster to which the node belongs, and the colors are assigned alternately along the vertical axis to indicate start and end of clusters. A node is displayed using a line that connect the same data object in the orderings across the vertical axes. Tracking these lines enables us to see the evolution of the data object in terms of the changes in its membership in clusters across the series. There are some attributes for which trend is completely different or they do not comply with general behavior of the other attributes of data sets they are know as outliers. Outliers are the noise or exceptions in data set which are generally detected using statistical tests.

Algorithms 5 and 6 show how to obtain the coordinates for the data objects in the parallel set-like representation and how to display them using a line, respectively.

7.2 Experiments and Results

We have implemented our visualization technique on three different types of data sets: univariate data [42], multiattribute data [6], and timeseries data [14].

We have generated similarity matrix series from univariate data of the social network of coauthorship [42] in two different ways: by varying (a) similarity functions using VAT as the seriation algorithm, and (b) permutation algorithms using 2-ring neighborhood function as similarity function. Figures 7.1 and 7.2 shows the parallel sets-like representation for the two series of similarity matrices, respec-

Algorithm 5: Parallel Sets-like Representation

```
Input  : Input file has $N$ number of vertices and $M$ number of attributes and
         adjacency matrix for each attribute,
         Similarity function choice,
         Permutation algorithm choice.
Output: Parallel set-like Representation
int PemrutedOrder[$M$][$N$];
std::vector<int> locationInAttributes;
double distanceBetAttributes = 30.0/$M$ double lenNode = 30.0/$N$;
for each attribute belong to $M$ do
    for each adjacency matrix $N \times N$ do
        Apply similarity function( similarityFunctionChoice)
        Apply Permutation algorithm( algorithmChoice)
        Save the PermutedOrder
for $i$ =0 to $N$ do
    for $j$ = 0 to $M$ do
        for $k$ =0 to $N$ do
            // if vertex corresponding to vertex number i is found
            if PermutedOrder[j][k]==i then
                save k in an vector locationInAttributes
    For vertex i the location in every attribute is found
    drawLine(locationInAttributes)
```

Algorithm 6: Algorithm for drawing a line to represent a data object across similarity matrices

```
void drawLine(std::vector <int> lineBetweenAttributes)
for  $i = 0$ to $M$ do
    draw line between these two vertices
    Vertex(leftmargin+(j*distanceBetAttributes),
    (topmargin+(k[j]*lenNode))+(lenNode/2))
    to
    glVertex(leftmargin+((j+1)*distanceBetAttributes),
    (topmargin+(k[j+1]*lenNode))+(lenNode/2))
```

tively. We can compare these against the matrix representation of the similarity matrices in Figures 4.5 and 4.6. We have highlighted cluster 37 in the matrix representation using the blue highlight, we highlight the same cluster in the permutation functions-based similarity matrix series in red in Figure 7.2. Cluster 37 comprises of 26 nodes or authors, namely, BarrettA.Lee, BarryWellman, BeateVolker, BelaJanky, BenedettaBelli-McQueen, BennyJose, BeomJunKim, BernardGrofman, BihchiiLauraYang, BlydenB.Potts, BonnieH.Erickson, BrentSimpson, BrianL.Foster, BrianMullen, BridgetA.Browning, BrunoMorra, CraigWilson, CullenGoldblatt, J.DavidJohnson, J.J.Hox, J.JillSuitor, J.M.Hyman, KatherineFaust, KatherineL.Woodard, KathleenM.Carley, and LenoardWaverman. Five of these 26 nodes, namely BrentSimpson, BrianL.Foster, BrianMullen, BridgetA.Browning and BrunoMorra, shown by yellow highlight in Figure 7.2. We can see that these authors form a cluster irrespective of the permutation algorithm. We can also see that in the original similarity matrix without applying reordering, all the five nodes except BrentSimpson lie in the same cluster.

The employee interaction data set [6] is an observational data on 14 Western Electric (Hawthorne Plant) employees from the bank wiring room. The employees worked in a single room and include two inspectors (I1 and I3), three solderers (S1, S2 and S4), and nine wiremen or assemblers (W1 to W9). The interaction categories include: RDGAM, participation in horseplay; RDCON, participation in arguments about open windows; RDPOS, friendship; RDNEG, antagonistic(negative) behavior; and RDHLP, helping others with work. From Figure 7.3, we can see (W1, S4) and (W9, S1) participate in horseplay and are friends who participate in argument about open window, do not behave negative with each other and help each other in their work. They are similar in all attributes and hence they lie in same cluster for each of the attributes. We can see from Figure 7.4 that most of the employees are friends and are ready to help each other in their work and a small group of people participate in horseplay, show negative behavior and argue about open windows. We can conclude that there are no outliers amongst these attributes as all are showing behavior that is expected from a group of employees working together.

Since we could not obtain a timeseries data for a social network, we have applied our technique on timeseries of an univariate data set to show proof of concept. From the monthly record of industrial production indices data in U.S.A. [14] data set from 1947-1993, we use ten years data from year 1947-1956. For every month, following eight attributes are given: industrial Production(IP) index, manufacturing, durable, nondurable, mining, utilities, products total, materials. These attributes define the indices for each month that mean they are the feature vector of months. Similarity

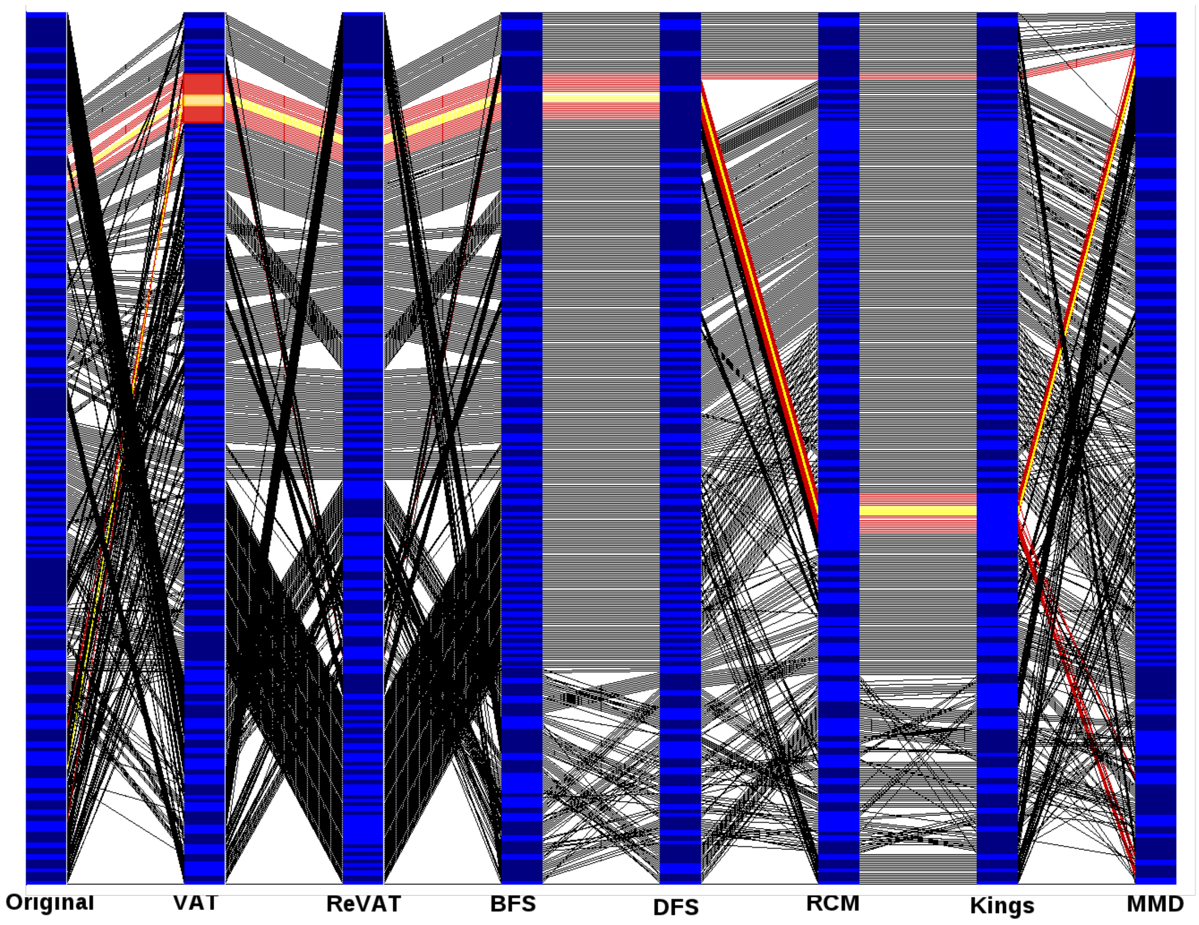


Figure 7.2: Parallel sets-like representation of a series of similarity matrices generated using k-ring neighbourhood for $(k = 2)$ as similarity function on social network of coauthorship data set [42] and varying the seriation algorithm. Red highlight shows all the 26 nodes or authors in cluster 37 highlighted in blue in the matrix representation of the similarity matrices in Figures 4.5 and 4.6. Yellow highlight traces the membership of five specific authors who are in a cluster irrespective of the permutation algorithm.

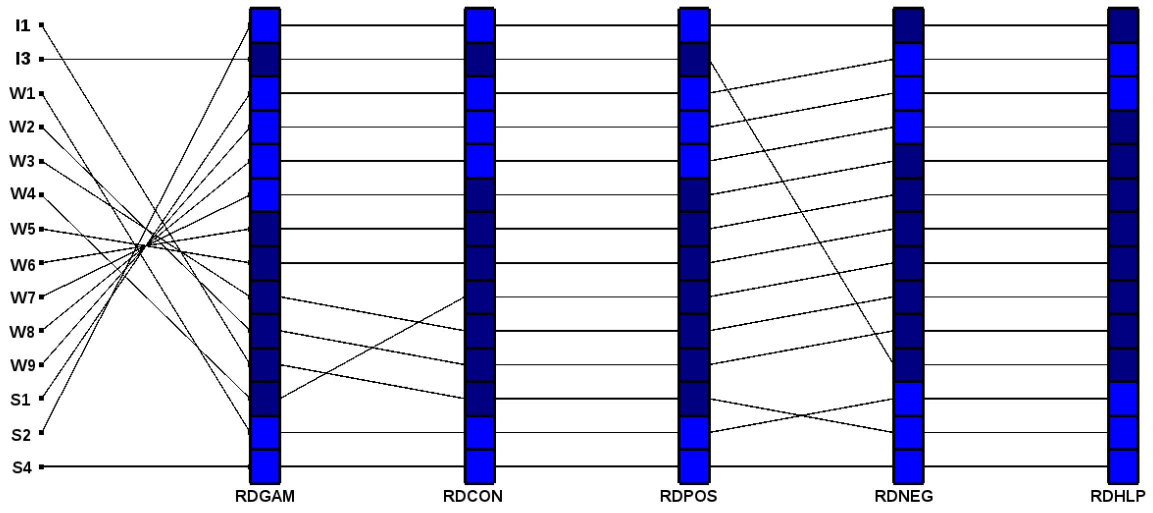
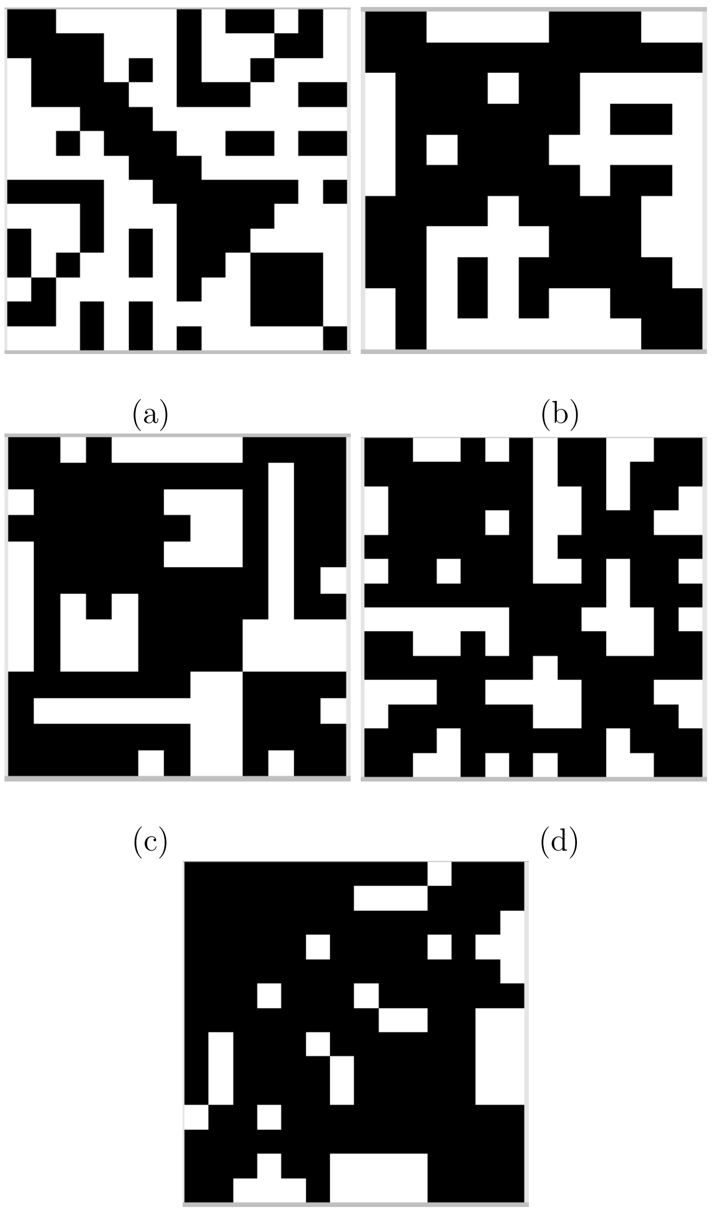


Figure 7.3: Parallel sets-like representation using k-ring neighbourhood function and VAT as seriation algorithm, for $(k = 2)$ for transformation of employee interaction data [6] for the following attributes: RDGAM, participation in horseplay; RDCON, participation in arguments about open windows; RDPOS, friendship; RDNEG, antagonistic (negative) behavior; and RDHLP, helping others with work.



(e)

Figure 7.4: Matrix representation of the similarity matrices generated using k-ring neighbourhood function, for ($k = 2$) for transformation of employee interaction data set [6] for five attributes and seriation using VAT, for the following attributes: (a) RDGAM, participation in horseplay (b) RDCON, participation in arguments about open windows (c) RDPOS, friendship (d) RDNEG, antagonistic(negative) behavior; and (e) RDHLP, helping others with work

score is computed between two months using cosine similarity function, which gives us ten similarity matrices of 12×12. This series of similarity matrices shows how the indices vary every year and which the groups of months that have similar indices are.

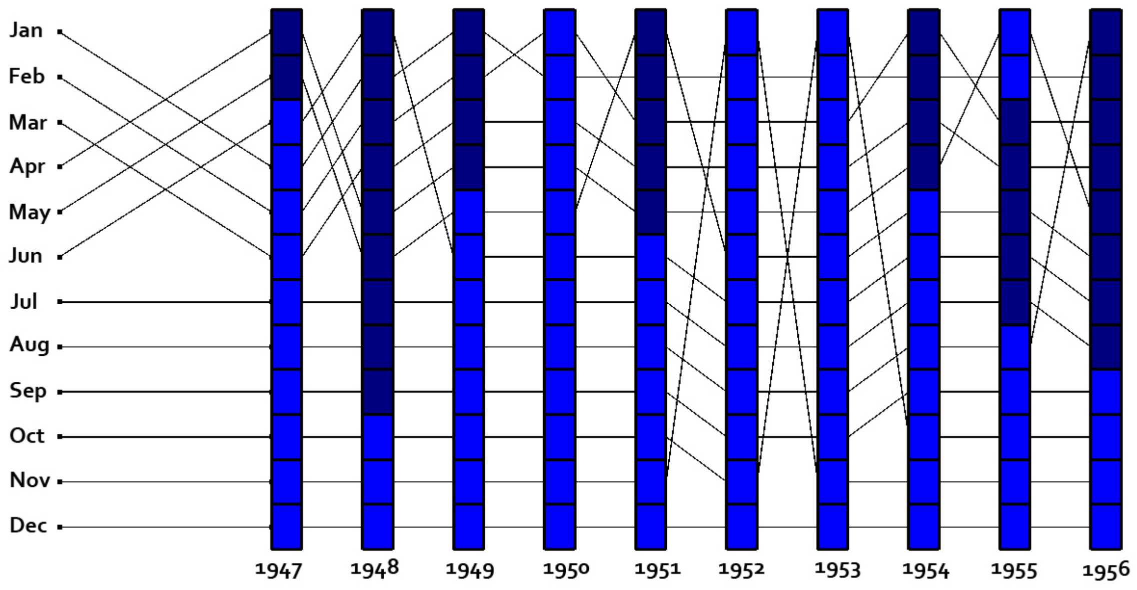


Figure 7.5: Parallel sets-like representation of industrial production data set [14] after applying cosine similarity as similarity function and VAT as seriation algorithm.

We can see from the parallel sets-like representation in Figure 7.5 that October, November and December lie in same cluster for every year although their order change for year 1952 and 1953 but they are back at the same location in 1954. This is not apparent from the matrix representations in Figure 7.6. We can also see from Figure 7.5 that there is not much change in the data in the months of October, November and December every year.

Summarizing, our proposed parallel sets-like representation of a series of similarity matrices is a minimalistic representation of the clustering results in the similarity

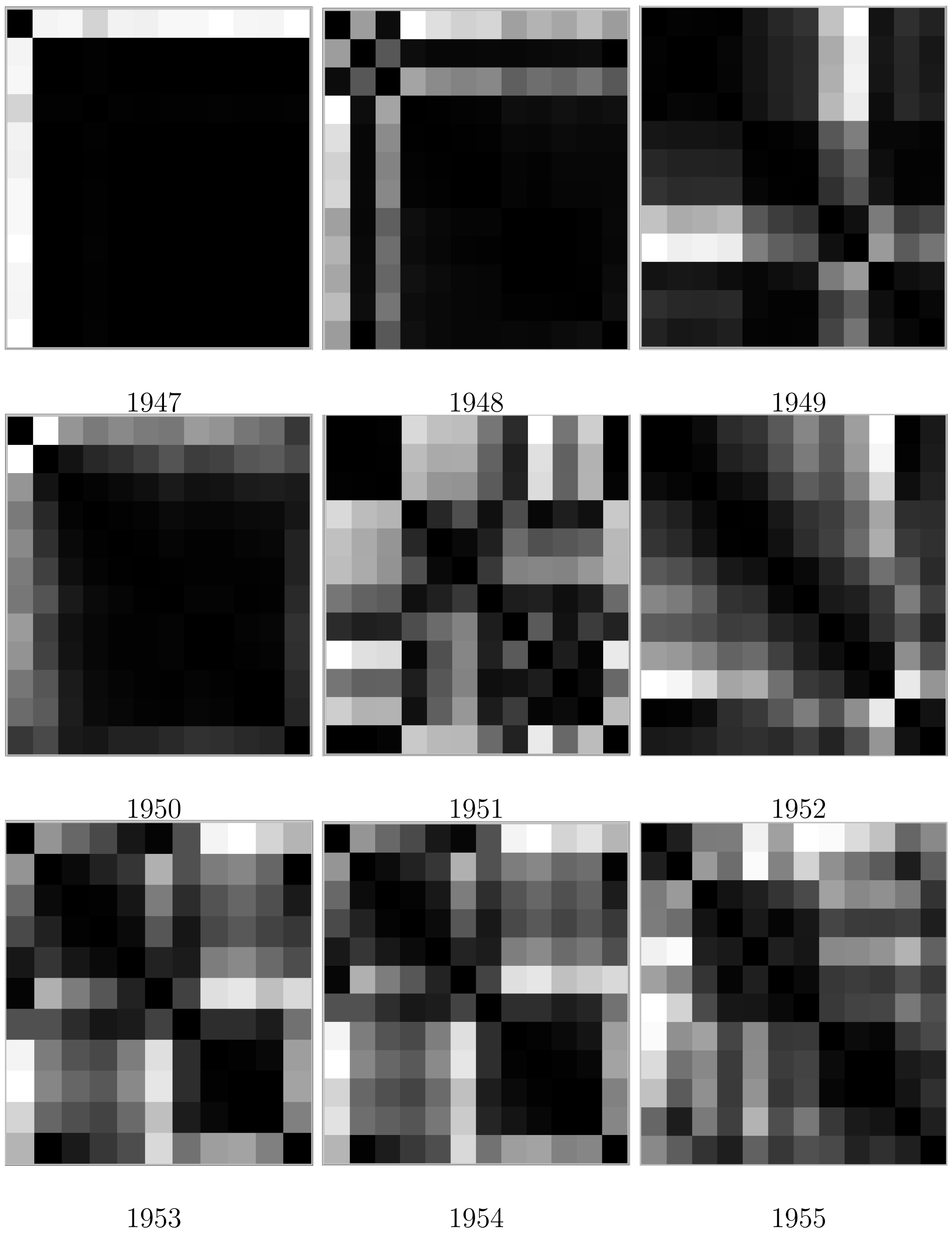


Figure 7.6: Representation of series of similarity matrix constructed by applying cosine similarity function and using VAT as seriation algorithm on industrial production data set [14] for the years 1947-1955.

matrices, showcasing how changes in the similarity matrix can perturb the clustering tendency in the network. It is a snapshot of the entire series, as opposed to viewing individual matrices to track the clustering behavior of the objects/nodes in the network.

7.3 Discussions

Multivariate visualization is an important subfield of data visualization that focuses on multivariate data sets. In [60], Wong and Bergeron have referred to multivariate data as “multidimensional multivariate data.” They have stated that multivariate data can be defined as a set of observations X, where the ith element x_i consists of a vector with m variables, $x_i = (xi_1 , . . . , xi_m)$. Each variable may be independent or dependent on other variables, dependent variables are referred to multidimensional variables and dependent variables are referred to multivariate.

For time series data, every similarity matrix is independent of each other, which can be treated similar to multidimensional data.

For multivariate datasets the order of attributes or vertical lines can increase and decrease the visual appeal of the parallel sets-like representation. In our work we are not experimenting with changing the order of attributes in the input.

Chapter 8

Conclusions

We have presented a novel ensemble for a visualization tool to visually assess clustering techniques in a small world network. Our tool compares the effectiveness of different similarity functions, layouts and permutation/seriation algorithms using synchronized matrix representation and node-link diagram of a network. We have shown that amongst our chosen list of seriation algorithms, VAT shows better clustering tendency assessment.

There have been several improvements to VAT since its inception to handle scalability, its scalable versions bigVAT [28] and sVAT [21] downsample data which could lead to inadvertent loss of information. We have proposed a parallel implementation of VAT, pVAT, using CUDA which will enable preserving all data. As expected, performance of pVAT is more efficient compared to the serial implementation VAT which makes our tool scalable when using VAT as a seriation algorithm.

We have proposed using a parallel sets-like representation to visually explore multidimensional data which can be posed as a series of similarity matrices. This representation enables us to track the membership of data objects in clusters across different similarity matrices which correspond for different time stamps, dimensions or attributes, characteristic function, such as similarity function or seriation algorithm. It has indirectly helped in visually detecting the outliers in the data set.

Since our work heavily depends on VAT, it inherits the limitations of VAT as well. Our future work will address the following shortcomings: (a) representation of dense graphs and for network data without any inherent clusters, and (b) theoretical evaluation for our assessment of the clustering capability of the similarity functions. For highly dense networks with large number of nodes, node-link diagram suffers from occlusion. Highly dense graphs will inherently have fewer number of clusters which will cause multilevel clustering to fail. Though matrix representation is limited by a spatial complexity of $O(N^2)$ is not scalable with data size, it can be resolved using pixel-level displays. VAT will have to be substituted by a suitable permutation algorithm for such graphs.

Bibliography

[1] J. Abello and F. van Ham. Matrix Zoom: A Visual Interface to Semi-External Graphs. In Ward and Munzner [57], pages 183–190.

[2] D. Auber. *Tulip : A huge graph visualisation framework.* P. Mutzel and M. Junger, 2003.

[3] D. Auber, Y. Chiricota, F. Jourdan, and G. Melançon. Multiscale Visualization of Small World Networks. In *INFOVIS*. IEEE Computer Society, 2003.

[4] A. Barrat and M. Weigt. On the Properties of Small-world Network Models. *The European Physical Journal B- Condensed Matter and Complex Systems*, 13(3):547–560, Jan. 2000.

[5] J. C. Bezdek and R. J. Hathaway. VAT: A Tool for Visual Assessment of (Cluster) Tendency. In *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, volume 3, pages 2225–2230, Piscataway, NJ, 2002. IEEE Press.

[6] R. Breiger, S. Boorman, and P. Arabie. Data set of social interactions in a bank wiring room. `http://moreno.ss.uci.edu/wiring.dat`, 1939.

[7] W. Chan and A. George. A linear time implementation of the reverse cuthill-mckee algorithm. *BIT Numerical Mathematics*, 20(1):8–14, 1980.

[8] C. D. Correa, T. Crnovrsanin, and K.-L. Ma. Visual Reasoning about Social Networks Using Centrality Sensitivity. *IEEE Transactions on Visualization and Computer Graphics*, 18(1):106–120, 2012.

[9] A. Dasgupta, R. Kosara, and L. Gosink. Meta parallel coordinates for visualizing features in large, high-dimensional, time-varying data. In *Large Data Analysis and Visualization (LDAV), 2012 IEEE Symposium on*, pages 85–89, 2012.

[10] J. Díaz, J. Petit, and M. J. Serna. A Survey of Graph Layout Problems. *ACM Comput. Surv.*, 34(3):313–356, 2002.

[11] N. Elmqvist, T.-N. Do, H. Goodell, N. Henry, and J.-D. Fekete. ZAME: Interactive Large-Scale Graph Visualization. In I. Press, editor, *IEEE Pacific Visualization Symposium 2008*, pages 215–222, Kyoto, Japan, 2008. IEEE.

[12] M. Erdélyi and J. Abonyi. *Node Similarity-based Graph Clustering and Visualization*, page 483494. Citeseer, 2006.

[13] E. Estrada and D. J. Higham. Network Properties Revealed through Matrix Functions. *SIAM Rev.*, 52(4):696–714, Nov. 2010.

[14] Federal Reserve Statistical Release G.17. Data set of monthly record of US industrial production indices from 1947-1993. `http://www.stat.duke.edu/~mw/data-sets/ts_data/industrial_production`, 1993.

[15] M. Ghoniem, J.-D. Fekete, and P. Castagliola. A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 17–24, 0-0 2004.

[16] 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*, 4(2):114–135, 2005.

[17] P. M. Gleiser and L. Danon. Data set of collaboration in jazz. `http://moreno.ss.uci.edu/jazz.dat`, 2003.

[18] J. Guare. *Six Degrees of Separation: A Play.* Vintage Books, 1990.

[19] P. Hage and F. Harary. Data set of Taro exchange. `http://moreno.ss.uci.edu/taro.dat`, 1983.

[20] M. Harris, S. Sengupta, J. Owens, Y. Zhang, A. Davidson, and N. Satish. CUDPP. `http://gpgpu.org/developer/cudpp-1-0a`, 2007.

[21] R. J. Hathaway, J. C. Bezdek, and J. M. Huband. Scalable Visual Assessment of Cluster Tendency for Large Data Sets. *Pattern Recogn.*, 39(7):1315–1324, July 2006.

[22] T. C. Havens, J. C. Bezdek, J. M. Keller, M. Popescu, and J. M. Huband. Is vat really single linkage in disguise? *Annals of Mathematics and Artificial Intelligence*, 55(3-4):237–251, Apr. 2009.

[23] N. Henry and J.-D. Fekete. MatrixExplorer: a Dual-Representation System to Explore Social Networks. *IEEE Trans. Vis. Comput. Graph.*, 12(5):677–684, 2006.

[24] N. Henry and J.-D. Fekete. MatLink: Enhanced Matrix Visualization for Analyzing Social Networks. In *Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction - Volume Part II*, INTERACT'07, pages 288–302, Berlin, Heidelberg, 2007. Springer-Verlag.

[25] I. Herman, G. Melançon, and M. S. Marshall. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Trans. Vis. Comput. Graph.*, 6(1):24–43, 2000.

[26] S.-H. Hong and K.-L. Ma, editors. *APVIS 2007, 6th International Asia-Pacific Symposium on Visualization 2007, Sydney, Australia, 5-7 February 2007*. IEEE, 2007.

[27] J. M. Huband, J. Bezdek, and R. Hathaway. Revised Visual Assessment of (Cluster) Tendency (reVAT). In *North American Fuzzy Information Processing Society (NAFIPS)*, pages 101–104, Banff, Canada, 2004. IEEE Press.

[28] J. M. Huband, J. C. Bezdek, and R. J. Hathaway. bigVAT: Visual Assessment of Cluster Tendency for Large Data Sets. *Pattern Recogn.*, 38(11):1875–1886, Nov. 2005.

[29] A. Inselberg and B. Dimsdale. Parallel coordinates: a tool for visualizing multidimensional geometry. In *Proceedings of the 1st conference on Visualization '90*,

VIS '90, pages 361–378, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[30] R. Jin, V. E. Lee, and H. Hong. Axiomatic Ranking of Network Role Similarity. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 922–930, New York, NY, USA, 2011. ACM.

[31] R. Jin, V. E. Lee, and H. Hong. Axiomatic Ranking of Network Role Similarity. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD 11, pages 922–930, San Diego, California, USA, 2011. ACM.

[32] Y. Koren and D. Harel. A Two-Way Visualization Method for Clustered Data. In L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, editors, *KDD*, pages 589–594. ACM, 2003.

[33] R. Kosara, F. Bendix, and H. Hauser. Parallel sets: Interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):558–568, July 2006.

[34] J. Leskovec. Gnutella peer-to-peer file sharing network, August 08, 2002. `http://snap.stanford.edu/data/p2p-Gnutella08.html`, 2002.

[35] J. Leskovec. Gnutella peer-to-peer file sharing network, August 09, 2002. `http://snap.stanford.edu/data/p2p-Gnutella09.html`, 2002.

[36] J. Leskovec. High-energy physics citation network, January 1993-April 2003. `http://snap.stanford.edu/data/cit-HepPh.html`, 2003.

[37] J. Leskovec. Wikipedia vote network. `http://snap.stanford.edu/data/wiki-Vote.html`, 2008.

[38] I. Liiv. Seriation and Matrix Reordering Methods: An Historical Overview. *Stat. Anal. Data Min.*, 3(2):70–91, Apr. 2010.

[39] I. Liiv, R. Opik, J. Ubi, and J. Stasko. Visual Matrix Explorer for Collaborative Seriation. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(1):85–97, 2012.

[40] H. Liu, J. He, D. Zhu, C. X. Ling, and X. Du. Measuring Similarity Based on Link Information: A Comparative Study. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints):1, 2012.

[41] J. W. H. Liu. Modification of the minimum-degree algorithm by multiple elimination. *ACM Trans. Math. Softw.*, 11(2):141–153, June 1985.

[42] C. McCarty. Data set of network of coauthorships in the Social Networks journal in 2008. `http://moreno.ss.uci.edu/data.html#auth`, 2008.

[43] C. Muelder and K.-L. Ma. Rapid graph layout using space filling curves. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1301–1308, Nov. 2008.

[44] C. Mueller. Sparse matrix reordering algorithms for cluster identification, 2004.

[45] C. Mueller, B. Martin, and A. Lumsdaine. A Comparison of Vertex Ordering Algorithms for Large Graph Visualization. In Hong and Ma [26], pages 141–148.

[46] C. Mueller, B. Martin, and A. Lumsdaine. Interpreting Large Visual Similarity Matrices. In Hong and Ma [26], pages 149–152.

[47] M. E. J. Newman. Data set of coauthorship network in network theory and science. `http://moreno.ss.uci.edu/netsci.dat`, 2006.

[48] A. Noack. Energy-based Clustering of Graphs with Nonuniform Degrees. In *Proceedings of the 13th international conference on Graph Drawing*, GD'05, pages 309–320, Berlin, Heidelberg, 2006. Springer-Verlag.

[49] Nvidia. CUDA. `https://developer.nvidia.com/category/zone/cuda-zone`, 2013.

[50] H. C. Purchase. Effective information visualisation: a study of graph drawing aesthetics and algorithms. *Interacting with Computers*, 13(2):147–162, 2000.

[51] D. A. Spielman and S.-H. Teng. Spectral Partitioning Works: Planar Graphs and Finite Element Meshes. In *FOCS*, pages 96–105. IEEE Computer Society, 1996.

[52] F. van Ham and J. J. van Wijk. Interactive Visualization of Small World Graphs. In Ward and Munzner [57], pages 199–206.

[53] V. Vineet, P. Harish, S. Patidar, and P. J. Narayanan. Fast Minimum Spanning Tree for Large Graphs on the GPU. In *Proceedings of the Conference on High*

Performance Graphics 2009, HPG '09, pages 167–171, New York, NY, USA, 2009. ACM.

[54] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. W. Fellner. Visual Analysis of Large Graphs. pages 37–60, Norrköping, Sweden, 2010. Eurographics Association.

[55] J. Wang, B. Yu, and L. Gasser. Classification Visualization with Shaded Similarity Matrix. Technical report, GSLIS, University of Illinois at Urbana-Champaign, 2002. 9 pages.

[56] J. Wang, B. Yu, and L. Gasser. Concept Tree Based Clustering Visualization with Shaded Similarity Matrices. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, ICDM '02, pages 697–701, Washington, DC, USA, 2002. IEEE Computer Society.

[57] M. O. Ward and T. Munzner, editors. *10th IEEE Symposium on Information Visualization (InfoVis 2004), 10-12 October 2004, Austin, TX, USA*. IEEE Computer Society, 2004.

[58] D. J. Watts and S. H. Strogatz. Collective Dynamics of 'Small-world' Networks. *Nature*, 393(6684):440–442, June 1998.

[59] D. Wishart. ClustanGraphics3: Interactive graphics for cluster analysis. *Classification in the Information Age, Springer-Verlag*, pages 268–275, 1999.

[60] P. C. Wong and R. D. Bergeron. 30 years of multidimensional multivariate visualization. In *Scientific Visualization, Overviews, Methodologies, and Techniques*, pages 3–33, Washington, DC, USA, 1997. IEEE Computer Society.

[61] W. Zachary. Data set of a university karate club. `http://moreno.ss.uci.edu/zachary.dat`, 1977.

[62] X. Zhao and A. Kaufman. Structure revealing techniques based on parallel coordinates plot. *The Visual Computer*, 28(6-8):541–551, 2012.

[63] H. Zhou, W. Cui, H. Qu, Y. Wu, X. Yuan, and W. Zhuo. Splatting the lines in parallel coordinates. In *Proceedings of the 11th Eurographics / IEEE - VGTC conference on Visualization*, EuroVis'09, pages 759–766, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association.

[64] H. Zhou and D. Woodruff. Clustering via Matrix Powering. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '04, pages 136–142, New York, NY, USA, 2004. ACM.

[65] H. Zhou, X. Yuan, H. Qu, W. Cui, and B. Chen. B.: Visual clustering in parallel coordinates. *Computer Graphics Forum*, 2008.