Effectiveness of Unique Grouping Techniques for Network Nodes in Serving Various Application Domains

Chen, Alan

UNIVERSITY OF CALGARY

Effectiveness of Unique Grouping Techniques for Network Nodes in Serving Various Application Domains

by

Alan Chia-Lung Chen

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

APRIL, 2013

Abstract

A network is an abstract representation of entities, which can be objects or concepts. Entities are generally represented by nodes, and connected to other entities in the model by links based on their relationship or interaction with the other entities. Networks are a simple but powerful tool for modeling and analyzing relationships between entities, which have become an important technique in many different fields of study. The semantics of the nodes and the links are determined based on the specific domain of study. Nodes in a network could be classified into groups. A group in a network is a subset of the nodes in the network that is being considered together for certain functions. Grouping network nodes refers to a technique of assigning labels to the nodes; grouping techniques are important for building an understanding of the network, and can be used in solving many problems in various domains.

Various techniques have been explored to group network nodes together, such that nodes in each group are highly connected, and nodes between groups have fewer connections. General grouping techniques will discover these high density groups in a wide variety of networks for further examination in numerous fields. The problem with general grouping techniques is that they are multipurpose tools, thus they produce groups of nodes with some characteristics that are commonly sought. Nevertheless, there may be situations that call for discovering groups that have an unusual characteristic. In these problems, a unique grouping technique that is designed specifically to address that particular problem would be a much more effective means to solve the problem. Accordingly, a general framework is proposed in this thesis to help guide the design of unique grouping techniques. This thesis will demonstrate the effectiveness and significance of unique grouping techniques through the development, and application of unique grouping techniques in four distinctive cases. This thesis will show that unique grouping techniques should be a serious consideration alongside general grouping techniques for research work dealing with networks.

Acknowledgements

I want to thank my supervisor Professor Reda Alhajj for his guidance throughout my program, and especially for his counseling in completing this thesis. I also want to thank Shang Gao, another member of the research group whom has given me direction for many of the projects I've worked on. I would also like to thank NSERC (The Natural Sciences and Engineering Research Council of Canada) and Alberta Innovates, whose financial support was a major factor for me deciding to complete this program. And I want to thank my mother and sister for their support.

I simply want to thank everyone that have contributed to my success during my Master's program, and everything else that has made this all possible.

Table of Contents

List of Tables

List of Figures and Illustrations

Chapter 1

INTRODUCTION

1.1 Background

1.1.1 General Networks

A network is an abstract representation of entities, which can be objects or concepts; these entities are generally represented by circles (nodes), and connected to other entities in the model by lines (links) based on their relationship or interaction with the other entities (see Figure 1.1). There are many different types of networks with different features. For example, the links of some networks are directed to indicate a one-sided connection or interaction. A number of other networks have weighted links to indicate strong or weak relationships, and in some cases these weights are signed to indicate positive or negative relationships. Networks are a simple but powerful tool for modeling and analyzing relationships between entities, which have become an important technique in many different fields of study [55]. Network analysis is an emerging field that is heavily based on graph theory, statistics, mathematics, and more recently computer science techniques, which have influenced the scalability of modeling and analysis. The power of network modeling is the interpretation of relationships or connections between entities, and the ease of visualization with graphical models.

1.1.2 Social Networks

Social networks are valuable structures that involve individuals or groups of individuals, and their relationship with the other individuals or groups in the network. The nodes in a social network represent individuals or groups, such as organizations, residents of particular areas, families, etc. The links in a social network indicate the relationship between the individuals or groups, such as organization position, communication, kinship, etc. Links in the social

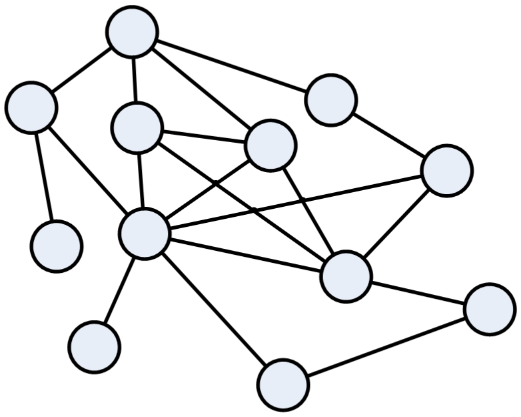


Figure 1.1: Example of a Network

network can often be directed, e.g., to indicate which individuals supervise which other individuals; in addition, these links can also be weighted, e.g., to indicate how strong of a friendship two individuals have. However, different types of nodes can also be present within social networks which represent other entities that have some connection to individuals or groups in the network, such as events or locations which the individuals or groups were present at.

Social networks are a model of the connections of individuals or groups within a society, and thus social network analysis is vital in the study of sociology. They provide a method to study the interactions of a large number of individuals to find special entities, patterns, or make predictive models. For instance, in order to find out who plays the most central role in an organization, we can construct a social network with individuals being the nodes and the link between two individuals representing certain social connections, like working on the same project. Once the social network has been constructed, it could be analyzed using knowledge discovery techniques, which look for how well connected nodes are to determine the most influential individuals. In this context I look specifically at criminal networks, which are a type of social networks where the individuals involved are criminals, related to crimes, or connected to illegal activities.

1.1.3 Neural Networks

Neural networks are models that are based on the biological neural network of the brain, and how the neurons transmit information [27]. They are a model of information processing in the brain, which can be used to model complex relationships between inputs and outputs, and to find patterns in data [22]. The nodes in a neural network represent neurons, which are cells in the brain that manage the processing and transmission of information. The links in a neural network indicate the structure of the interconnected neurons, and thus how information passes through the neural network.

A learning algorithm is used to adapt the structure of the neural network to uncover complex patterns in a training data set. The goal is to use the simple processing of neural networks to emulate a complex system. Neural networks are used in a variety of fields to solve problems, such as speech recognition, image recognition, artificial intelligence in video games, robotics, and other machine learning problems [20, 1, 52, 19].

1.2 Grouping

A group in a network is a subset of the nodes in the network that is being considered together for certain functions; these sets may contain any number of nodes which are related in some way. There are many different names for groups in networks, such as clusters, communities, modules, and partitions. Grouping network nodes refers to a process of assigning labels to the nodes within the network under consideration, where the labels assigned may or may not be distinct. These labels determine which groups the nodes in the network belong to.

Grouping techniques may produce any number of groups from a network, where the groups may overlap with other groups meaning nodes can belong to several groups. Grouping techniques are important for building an understanding of the network, and can be used to help solve many problems in different domains. Grouping techniques can be used in a direct matter to solve problems, where the grouping results provide the required information. In other cases, grouping techniques can be used in an indirect matter to tackle problems by generating informative groups of nodes, which go on to be further processed to achieve an ideal solution.

There are many researched grouping techniques for network nodes, such as clique based techniques, clustering techniques, modularity optimization techniques, etc. However, these are general techniques that typically focus on finding groups of nodes with a high density of links within the group. These are general grouping techniques that have been developed for common situations, which have the advantages of being well researched, and documented for good reliability and easy implementation. Additionally, general grouping techniques are broadly applicable to many different types of networks for various situations.

The problem with general grouping techniques is that they are multipurpose tools, thus they produce groups of nodes with some characteristics that are commonly sought. Nevertheless, there may be situations that call for discovering groups that have an unusual characteristic, such as finding groups of nodes with no links to each other. In these problems, a unique grouping technique that is designed specifically to address that particular problem would be a much more effective means to solve the problem. While unique grouping techniques are developed for a singular problem, these grouping techniques can prove valuable to the research of unrelated work. For these reasons, a framework for developing unique grouping techniques is proposed and discussed in this thesis. This thesis will also demonstrate the effectiveness and significance of unique grouping techniques through the development, and application of unique grouping techniques in four distinctive cases.

1.3 Thesis Overview

This chapter (Chapter 1) provides a background on networks and an overview of the thesis. Chapter 2 will describe other related research on general and unique grouping techniques to give a sense of the current pool of research knowledge, utilizing grouping techniques on network nodes. Chapter 3 explains the general framework this thesis proposes for developing and applying unique grouping techniques in practice. Chapters 4, 5, 6, and 7 are case studies that show the effectiveness of using unique grouping techniques for networks to solve a wide range of problems.

The problem of finding and tracking the customer shopping behaviors in different market segments, with no impositions on the customers is detailed in Chapter 4. The social networks of criminals are a valuable source of information for investigation, but naturally criminals will seek to conceal their connections or their presence in the criminal network to impeded investigators. It follows that exposing hidden links and hidden nodes in criminal networks is a critical problem. Chapters 5, and 6 present a solution to each of these two problems, respectively. The last case study (Chapter 7) illustrates a real world example of developing, and applying a unique grouping technique to support the computation of the results for surveys in a charity and donor matching service. The final chapter (Chapter 8) summarizes the work of this thesis, and gives the conclusion on the importance of developing and applying unique grouping techniques. In addition, the final chapter discusses some future research directions for this thesis.

Chapter 2

RELATED WORK

2.1 General Grouping Techniques

Various techniques have been explored to group network nodes together, such that nodes in each group are highly connected, and nodes between groups have fewer connections. This means that groups of nodes have a high density of links, whereas there are sparser links between the groups of nodes. General grouping techniques will discover these high density groups in a wide variety of networks for further examination in numerous fields. In extreme cases where the group of nodes has the highest density possible, it is known as a clique.

A clique is a set of nodes such that every node in the set is connected in the network to every other node in the set, which means there cannot be anymore links included between nodes of the clique. This is because every possible link already exists, thus making cliques the groups of nodes with the highest density of links.

One type of general grouping techniques are clique based techniques. These are methods that locate the maximal cliques or cliques of a specified size; maximal cliques are essentially the biggest cliques possible, such that no other clique can be found that contain every node of the maximal clique. There are many clique based techniques, one example of which is the Bron-Kerbosch algorithm [30]. This is a recursive algorithm that finds all the maximal cliques in a network by considering three sets of nodes, R, P, and X. The algorithm starts with two empty sets R and X, with the set P containing all the nodes of the network, then considers one node at a time from P, makes a recursive call by moving the node into R, and reducing P and X down to the neighbors of the node. The final step after the recursive call will add the node in consideration into the set X. This recursion continues until the set P is empty, and if the set X is empty as well, it signifies that R is a maximal clique.

Another type of general grouping techniques use clustering methods, which incorporates different types of similarity measures to assess how alike two nodes are. A variety of clustering methods will group similar nodes together based on the similarity measure chosen, and keep dissimilar nodes in different groups. For example, the Jaccard index can be used as the similarity measure with agglomerative hierarchical clustering. The Jaccard index calculates the similarity between two sets by dividing their intersection by the union of the two sets (see Equation 2.1). The sets in networks can be established by the links in the row representation of the nodes in the adjacency matrix. Agglomerative hierarchical clustering begins by considering every node in the network as a cluster, then finds the clusters with the closest similarity between each other's centers and merges the two clusters into one cluster [33]. The merging of two clusters with the closest similarity repeats until there is only one cluster remaining, or the minimum cluster similarity or quantity is reached to stop merging the clusters [64].

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.1}$$

Modularity optimization techniques are another type of general grouping techniques that is prevalent in many research works. A modularity score is utilized in these methods to aid in the arrangement of the groupings of nodes by searching for groupings that maximize the modularity score. The modularity score provides feedback on the strength of the groupings produced based on the structure of the network, considering the density of links within and between the groups of nodes (see Equation 2.2). Hence groupings of nodes with many links inside the groups, but very few links outside the groups will have a high modularity score.

In Equation 2.2, A_{ij} represents the weight of the link between nodes i and j, $k_i = \sum_j A_{ij}$, c_i denotes the group to which node i is assigned, $\delta(a, b)$ is 1 if $a = b$ and 0 otherwise, and $m = \frac{1}{2}\sum_{ij} A_{ij}$.

The Louvain method is an example of a modularity optimization method that uses a greedy algorithm [10]. In the Louvain method's first stage, all nodes of the network start off by being considered in its own group, then the method moves nodes into their neighboring (linked to) groups that give the most increase in modularity. The second stage assembles a new network based on the original network by aggregating the nodes of the groups into a node representation, combining link weights, and creating or updating self loops for links between nodes in the groupings. These two stages are repeated until the modularity score stops increasing, thus a maximum modularity will be obtained.

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{2.2}$$

2.2 Unique Grouping Techniques

There have been a number of unique grouping techniques created in diverse areas of disciplines that generate groups of nodes with distinctive properties to address different problems. These unique grouping techniques are difficult to identify, because these techniques are typically not labeled as grouping techniques, but just methods that produce a solution to their problem at hand. This is a problem, because even though each unique grouping technique is not broadly applicable to many situations, there can still exist some substantially different subject matter which can benefit from the application of the unique grouping technique.

Christley et al.'s research work on simulating the spread of infections in social networks to identify high risk individuals gives an example of a unique grouping technique [14]. In this study, several networks were randomly generated to be used as the social network for the simulation model. The simulation model assumed that every node in the network was susceptible to infection, and began by randomly selecting a node from the network to become infected. The model simulates time steps iteratively until the infection has disappeared from the network, and each time step gives the infection a chance to spread to a infected nodes' neighbors based on an infection probability, and the number of infected neighbors a node is connected to. After a set amount of time steps, infected nodes in the simulation model will recover from the infection, thus no longer spreading the infection and also becoming immune to future infections. The study ran three sets of 500 simulations of the model for each network, with different infection probabilities for each set, and recorded the number of times each node was infected within a set of 500 simulations. Figure 2.1 illustrates the results of one network, where the shape of the nodes correspond to the number of times the nodes were infected throughout the 500 simulations. In this figure, nodes shaped in a cross had the least amount of infections, while nodes shaped in a circle had the most amount of infections during the 500 simulations. The shape of the nodes essentially groups the nodes of the network into several groupings based on their likelihood of being infected. These groupings supply infectious disease control agencies with groups of individuals with varying risk factor levels for becoming infected, which can provide a guideline for what groups of people to monitor for detecting possible outbreaks of an infectious disease.

A technique like this can be informative for businesses promoting a new product. For example, word of mouth is considered by many to be the most effective advertising. Accordingly, the infection simulation model can be used to forecast the effects of word of mouth advertising for a product. In the model, an infection can be seen in this context as a customer purchasing and enjoying the product, and would thus likely spread positive word of mouth advertising to influence other customers to buy the product. The recovery from an infection can be seen in this context as the hype of the new product dying down. Consequently, the customer would likely stop talking about the product, and would probably not buy the same product again in most cases. In this scenario, businesses can use information from popular social networking websites to build their social networks, and determine the groups of individuals who are not expected to buy the product, so future promotions can be developed to target these groups.

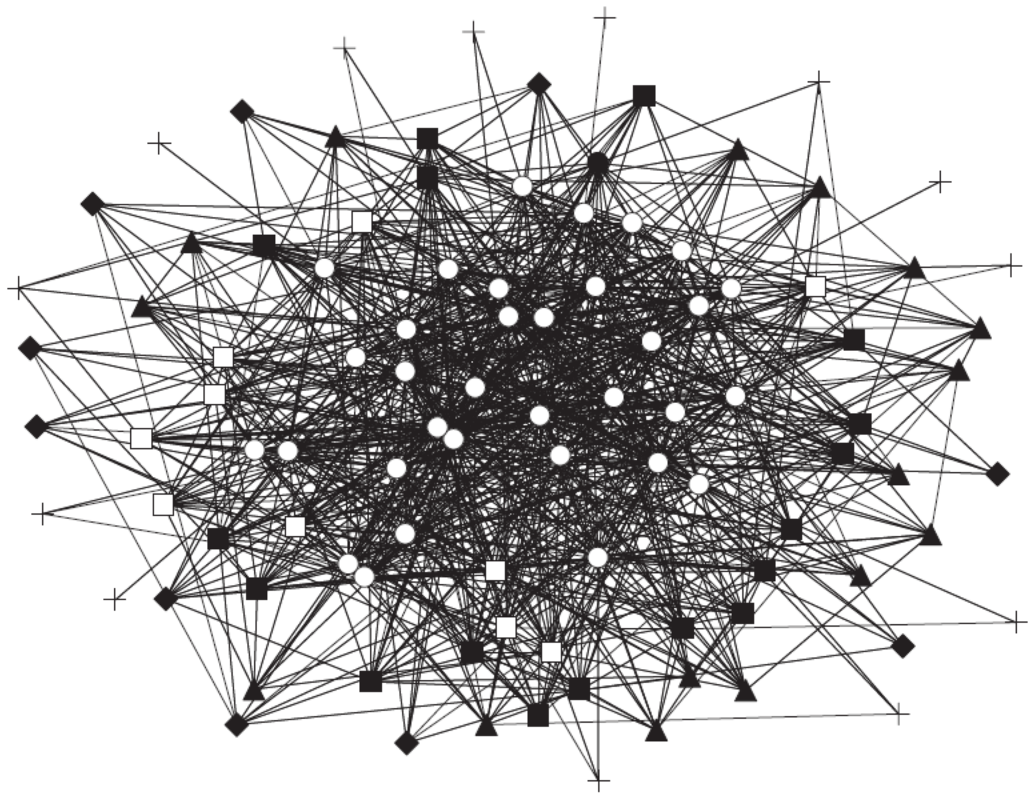


Figure 2.1: Infection Simulation Network [14]

Chapter 3

IMPLEMENTING UNIQUE GROUPING TECHNIQUES IN PRACTICE

3.1 Grouping Techniques

3.1.1 Grouping Nodes

There are many challenges where the grouping of nodes in networks is an essential technique in establishing a solution, because the groupings of nodes can provide valuable information for solving the problem. For many businesses, social networks are an important source of data, which need to be examined to better plan business strategies. Specific business promotions may want to find groups of close friends, so they can target the promotion to one or a few of the people in the grouping, who are most likely to spread word of mouth advertising to the rest of their friends in the group. This is especially vital for businesses that offer products or services that engage more than one person, because the groups of close friends provide a set of potential customers who can make full use of the product or service with their friends, and split the cost.

Market researchers may want to study groups of people in the social network who have similar interests or characteristics, to determine how to best tailor their products, services, or advertisements to attract people with similar attributes to the group into purchasing from them. Businesses may also want to find many different groupings of people with similar characteristics, to evaluate how well their products are doing in the market according to each segment.

Some other examples of challenges that require the grouping of nodes can still be found in social networks, where researchers may want to find the groups of people who are highly connected to determine how a highly infectious disease will spread from area to area [14]. Security experts might want to study networks of criminals to determine how best to operate resources for preventing criminal activities [57]. In bioinformatics, networks of genes may be studied to find groups of genes that have a lot of interaction with each other, which may lead to medical breakthroughs on causes or treatments of diseases [9]. For financial analysts, they may want to find tightly knit groupings of stocks modeled using networks of nodes that are correlated to help predict share prices [34]. These are some examples which demonstrate how the grouping of nodes in networks is an important set of techniques that provide a pivotal role in answering many challenges.

3.1.2 Input and Output

Unique grouping techniques will often need to be developed to solve specific problems for networks, because general grouping techniques do not separate the network nodes in a matter that is required for solving the problem. Or if general grouping techniques do prove useful in providing the needed partitions of nodes, then the development of unique grouping techniques might be pursued to address specific issues with much more effective results than general grouping techniques. The main input of these grouping techniques will be the network of nodes that need to be processed to produce the main output, which would be the groupings of the network nodes.

Both the input network of nodes and output groupings of network nodes may or may not be flexible, which will have important implications for the development of the grouping technique. If there is flexibility for the input to the grouping technique, then this is a positive for the development of the grouping technique. The reason is because this allows for more options with regards to the network to be used as input when designing the grouping technique, since vital information can be collected from the input network if the most appropriate network is used. On the other hand, if there is no flexibility for the input to the grouping technique, then this is a negative for the development of the grouping technique.

This will impose a constraint on the design of the grouping technique to consider a specific configuration of the input network, which can limit the information available and thus limit the options available when designing the grouping technique.

In the case that there is flexibility for the output of the grouping technique, then this is beneficial for the development of the grouping technique. This is advantageous because the design of the grouping technique has the opportunity to produce an intermediary result for the groupings of nodes, after that this can be further processed and fine tuned to find a solution to the problem. When there is no flexibility for the output of the grouping technique, then it will be more problematic for the development of the grouping technique. When the output of the grouping technique is constrained, this means the grouping technique by itself will need to produce a solution to the problem, because there will not be any other procedures to further improve the results, and will thus limit the design options for the grouping technique.

3.1.3 Arrangement

The grouping technique implemented to solve the problem may be the only process required to solve the problem, or it may be a process in a series of procedures carried out to solve the problem (see Figure 3.1). Regardless, the grouping technique is an essential process, which should be looked at carefully. If the grouping technique is the only step required, then it is likely that the input network itself is not preset and thus flexible. This is because there is no process that occurs before the grouping technique to produce a particular network to be used as input to the grouping technique. On the other hand, the outcome from the grouping technique in this situation is not flexible, because there is no process afterward to further revise the output of the grouping technique to provide a better solution.

The grouping technique may also be a part of a chain of procedures, where it can occupy an initial, intermediate, or final step in the chain. If the grouping technique is the initial procedure to execute, then it is likely that there is no fixed input network for the grouping

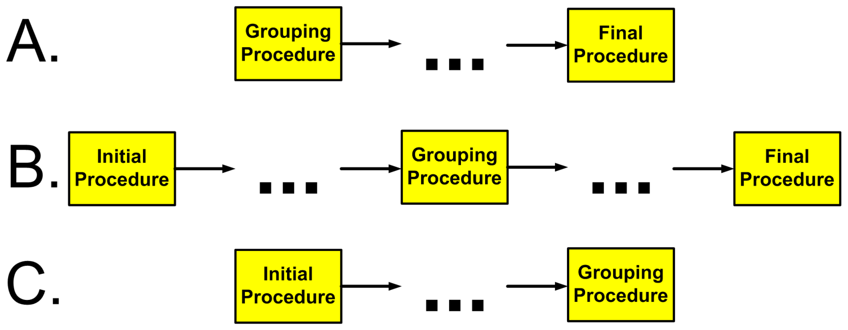


Figure 3.1: Positions of Grouping Techniques

technique; therefore there would be flexibility for the input. The reason for this is the same as the reason for the situation where the grouping technique is the only step. Since there is no process that precedes the grouping technique, it is likely that no specific network has been created to be utilized in the grouping technique. In this case, the output is also likely to be flexible, as processes can be developed to improve the results of the grouping technique.

One example of this is mentioned initially in the Grouping Nodes section, where a business would want to find groupings of close friends for their promotions, to target the people in the group most likely to spread word of mouth advertising to the rest of their friends in the group. In this case, the grouping technique to find groupings of close friends will likely be the initial step, because the social network data would likely be readily available, either from online sources, or the business data warehouses. This means there is no need for any processing of the social network to prepare it for the grouping technique. After the grouping technique produces the groupings of close friends, then the final step would proceed to identify the people in the grouping with the highest chance of spreading word of mouth advertising to the rest of their friends in the group.

Another example of a grouping technique which is part of the initial step in the chain of procedures can be found in the example of Finding Hidden Nodes in Criminal Networks (see Chapter 6). In this example, there are two correlated criminal networks and one network is being used to infer if there are hidden nodes in the other criminal network, and what the characteristics of those nodes are. As the networks are already given, the first step partitions the nodes of both networks to help find and match similar partitions between the two networks. These groups provide information which is used in the final step to determine which criminal nodes from one network match the criminal nodes of the other network, and the criminal nodes which have no matches could potentially indicate a hidden criminal node in the other network.

For grouping techniques that are an intermediate step in a chain of tasks, it is likely that the input is not flexible. The task that finishes before the grouping technique will probably generate or have some hand in creating a particular network for input into the grouping technique. In contrast, the output of the grouping technique in this instance is likely flexible, for the same reason as when the grouping technique is the initial step. Since there are additional procedures which execute after the grouping technique, tasks can be designed to touch up the groupings of the nodes produced.

An example of this is mentioned previously in the Grouping Nodes section in the field of bioinformatics. Researchers in bioinformatics might want find groups of genes with a high amount of interaction in gene networks to discover causes or treatments for numerous diseases. For this task, the first step would most likely be to have a procedure to create the network of genes, as this network is not likely to be present in the data set collected from experiments and would need to be inferred from the data set instead. The grouping technique to find genes with a high degree of interaction could then proceed after the network of genes is created. The groupings of genes produced by the grouping technique would then have to be further investigated in the final step. This could help determine if a specific gene is a major source for contributing to a disease and how this gene affects the disease, which may lead to medical treatments.

One more example of a grouping technique that is an intermediate procedure can be found in the example of Finding Hidden Links in Criminal Networks (see Chapter 5). In this example, a criminal network is being examined to determine if there are any hidden links within the network and reveal where these hidden links may be. Although the network is already available in this situation, the first step in the solution needs to further process the network and divide the network up into smaller sub-networks. Subsequently, the grouping technique will find potential hidden groupings of nodes in each of the criminal sub-networks who may all be connected. The final step cross examines the potential hidden groupings found to indicate prospective hidden links, and the involved nodes in the criminal network, which should be examined by law enforcement agencies.

When the grouping technique is the final step in the series of processes, the input network for the grouping technique is likely not flexible. This is because of the same reason as for when the grouping technique is in the intermediate step, as the previous step will probably influence the composition of the network given to the grouping technique. In addition, the output of the grouping technique is also likely not flexible; because just as in the case where the grouping technique is the only step, this grouping technique needs to produce the correct groupings of nodes to solve the problem, as it is the last step.

There is an example of this mentioned in the Grouping Nodes section with regards to financial analysts, who may be interested in modeling stocks in networks of nodes to uncover strongly correlated or dependent groupings of stocks, to forecast when share prices might rise or fall, depending on the performance of the other shares in the same group. Here, the financial analysts will probably have access to a large collection of data on the stock market, but there is most likely no network of stocks available within this data. This indicates that the first step is to create a network of stocks which are connected based on how dependent their share prices are. The creation of the network will then feed into the grouping technique, allowing it to function as the final step to discover groups of stocks which are well correlated, and then the financial analyst can watch for fluctuations of shares within the groups to anticipate the fluctuations of other shares in the group.

An alternative example of a grouping technique which is part of the final step in the chain of procedures can be found in the example of Post Processing of Association Rules into Segments (see Chapter 4). This example deals with using a data set consisting of transactions in a supermarket, and the buying behaviors of the customers, to group the buying behaviors of the customers into different market segments to better serve the business operation. The first step in the solution takes the data of customer buying behaviors and the transactions of the supermarket, and models this information in a network of nodes. The final step in the solution runs a grouping technique multiple times over the created network to generate groupings of customer buying behaviors, and further refines these into several groups that represent the buying behaviors of different market segments. These groups provide valuable information to business mangers on what the buying behaviors are for specific market segments, which can be monitored for how they change over time to better plan and adapt business strategies.

It is best if the grouping technique is not required to be the only step, and intermediate step, or the final step. This will generally allow for the greatest flexibility when designing the grouping technique, given that the input network of nodes and output groupings of nodes would not be constrained. This indicates that the design of the grouping technique can consider different types of input networks, and post processing to better refine the groupings of nodes generated by the grouping technique.

3.1.4 Aspects

In designing a unique grouping technique, there are two significant benefits; and one is the understanding of how and why the grouping technique works. This is harder to comprehend with the implementation of grouping techniques created by others, because it is work which may need further investigation to understand all the nuances of the technique. When designing a unique grouping technique, all the subtle intricacies of the grouping technique are apparent to the designer, due to it being conceived in their minds. Another significant benefit is that unique grouping techniques are likely more effective in solving the problems they are designed for, when compared to general grouping techniques. This is because general grouping techniques are broad-spectrum and thus not problem specific.

When designing a unique grouping technique, there are many important factors to take into consideration. Some of these considerations include the time and resources that are needed, as well as the feasibility and risks of the project. While timelines and resources can be set, it is very difficult to forecast exactly what assets and how much time will be needed. One of the main reasons is because there is an element of creativity needed during the development of unique grouping techniques, which is not a science. The feasibility and risks are other important factors to consider, because sometimes the solution requirements for the problem may be too difficult to solve, or it may be unachievable.

It can be difficult to develop a unique grouping technique, and although the grouping technique is unique, it does not have to be entirely original. The unique grouping technique can be based on or inspired by existing grouping techniques if they can help solve the problem (see Chapter 4). Furthermore, the uniqueness of the grouping technique does not mean that the unique grouping technique cannot be reused to solve other problems (see Chapter 5). The uniqueness of the grouping technique just makes it not as applicable to as many problems as general grouping techniques. Once the unique grouping technique is designed, the comprehensive understanding of how the unique grouping technique works may help make it less challenging to apply to other problems. The unique grouping technique developed also does not need to be complicated, and can be kept simple as long as it provides satisfactory groupings to solve the problem (see Chapter 6). Furthermore, unique grouping techniques are not just techniques that can only solve theoretical problems, but are also applicable to real world problems (see Chapter 7).

3.2 The Framework

3.2.1 Overview

A general framework is proposed to help guide the design of unique grouping techniques (see Figure 3.2). There are two main phases of this framework, which are the analysis phase and the design phase. In the analysis phase, the main tasks are to create the network and conduct research on existing grouping techniques, and other research works related to the problem. The design phase is the difficult part, because the main tasks are to develop a method to separate the nodes into the required groupings, and then test that method to ensure that it does what it is suppose to do. Developing the method is the most difficult part, because there is some degree of ingenuity needed to formulate the solution. This would be unique to each situation, and it is difficult to describe the exact steps needed in sequence for the development of the grouping method. It is also difficult to recycle the same steps used to develop a grouping method, because different situations will require different approaches. Some people may need to look at the problem from a different perspective, and some problems might be better tackled with other methodologies.

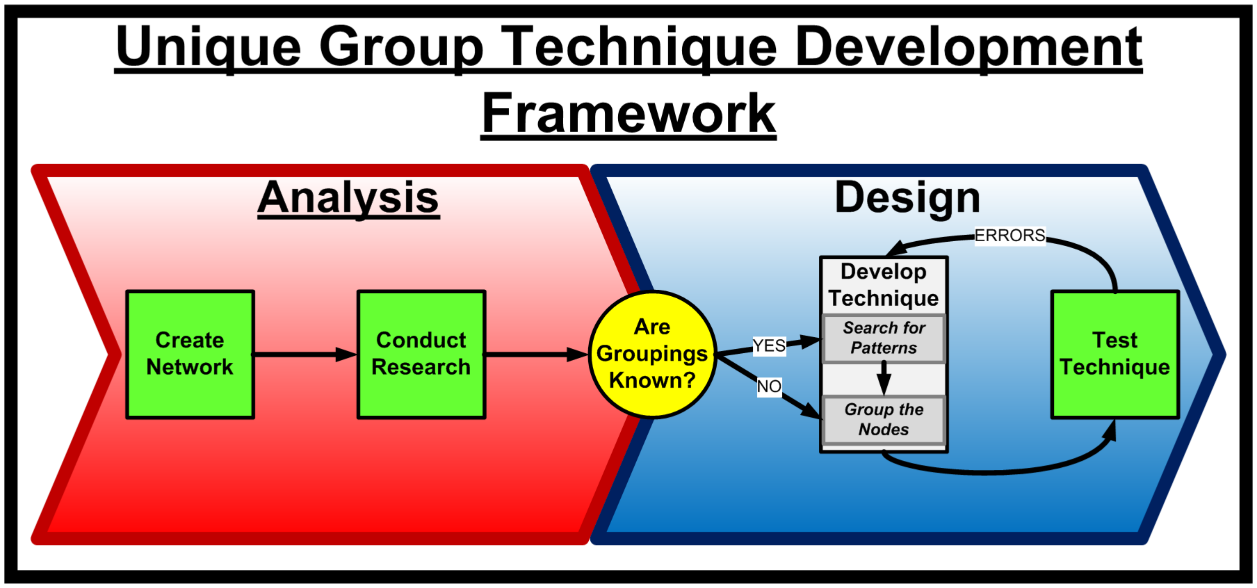


Figure 3.2: Framework for Developing Unique Group Techniques

The framework proposed is a general system to guide the development of unique grouping techniques for various problems. This framework may not be applicable in some situations, or maybe some practices in the framework will not be appropriate for solving the problem depending on the nature of the case. Nevertheless, this framework can serve as a reference point, and provide direction for the development of unique groupings techniques that cannot utilize this framework.

3.2.2 Analysis

The first step in the analysis phase is to determine the network of nodes that is needed to solve the problem. This includes determining what the nodes and links in the network will represent, and the source of data that is needed to create this network.

Depending on the specifics of the problem, sometimes the network of nodes may essentially be predetermined, as there cannot be any other network that can be used to solve the problem. The network settled on in this phase does not have to be permanent; this network can be adjusted in later stages if a more suitable network is discovered. The purpose for conceiving the network initially is to get a sense of the network model for solving the problem, as well as to employ the model of the network for the next step.

The last step in the analysis phase is to conduct research on existing grouping techniques. This research will help determine if there are existing general or unique grouping techniques that can be utilized to successfully solve the problem. It is a good idea to use existing grouping techniques when possible, assuming that the grouping technique has adequate performance in terms of efficiency and results. Existing grouping techniques will have documentation regarding the technique, which will help in their implementation for solving the problem. Furthermore, existing grouping techniques will have already been tested and have most if not all of the faults resolved. Therefore, using existing grouping techniques will likely be less time consuming, and less uncertain than developing a new unique grouping technique. The research will also help brainstorm ideas, which can aid in the development of the unique grouping technique, by revealing mechanisms from existing grouping techniques that can be applicable to the development of the unique grouping technique.

3.2.3 Design

After the analysis phase, the main and most difficult component is the design of the unique grouping technique. The major step in the design phase is the development of the unique grouping method. This is a step that requires innovation and can be complex or simple, extensive or short, successful or disastrous, and so forth. This step is a huge variable that can be drastically different depending on the situation. There are two suggested steps to take in developing the unique grouping technique, depending on whether there is sufficient knowledge on the contents of the groupings of nodes needed for the solution.

Knowledge on what nodes from the network will be in which groupings could possibly be deduced from the details of the problem, the identities of the nodes, and the type of connections in the network. There are many other ways to infer this knowledge, such as examining other data that relates to the source of information used to create the network, or examining the groupings of nodes in other research with similar problems. With this knowledge, sample groupings of nodes from the network can be composed to contribute to solving the problem. These groupings may contain a pattern or model that can be significant to the development of the unique grouping technique. The unique grouping technique might be able to replicate this pattern or model found in the sample groupings to solve the problem.

In the case where this knowledge cannot be obtained, or when there is no pattern or model discovered in the sample groupings compiled, then the next suggestion is to search for ways to separate the nodes into groupings that will solve the problem. Consideration of the nodes' different elements may lead to logical approaches for breaking up the nodes into groups to solve the problem. This approach can then be used in the unique grouping technique to produce the groupings of nodes that are desired.

Designing the unique grouping method can be a daunting task. However there are practices that can provide some assistance during this designing process. One practice is to look at existing grouping techniques for inspiration, or the opportunity to adapt the technique to fit the problem at hand. Existing grouping techniques can provide a starting point, and possibly reduce the amount of work needed for developing a unique grouping technique. Another good practice is to try applying simple solutions first, which keeps the technique less complex, therefore it will be easier to understand, test, and implement. Taking into account different networks if that selection is available can provide new opportunities when designing the unique grouping technique, and is a good practice when there is no progression made in the development.

Once the unique grouping technique has been developed, then the next step is to test the technique. This is a critical step to find the faults of the developed unique grouping technique. The testing should aim to ensure that the outcome of the grouping technique is what is expected and required for solving the problem. Additionally, the testing should also examine all the extreme conditions of the network, as well as any unique circumstances where the network might be configured in a way that can possibly produce errors. When the testing completes, then adjustments will need to be designed for the unique grouping technique to fix the errors found. Therefore, the development process in the previous step should be returned to for the development of the renovations to the unique grouping technique. The testing and repairing of the unique grouping technique should continue indefinitely in a loop until there are no more errors found. Subsequently, the unique grouping technique will be complete and ready for implementation.

Chapter 4

CASE STUDY ONE: Post Processing of Association Rules into Segments

4.1 Introduction

This case study demonstrates the effectiveness of a unique grouping technique in segmenting association rules, which are mined data that can represent customer behaviors. In this case study, the developed unique grouping technique is the final step in the series of processes. The unique grouping technique encompasses phase two of the solution, where the groups are created, and phase three of the solution, where the groups are merged and reduced. The unique grouping technique in this case study is based on a graph problem known as graph coloring, which shows that unique grouping techniques can be inspired by existing techniques to help with the creativity part of developing unique grouping techniques. The developed solution in this case study is an important technique in data mining, especially for businesses as they have a target market. The partitioning of rules into non-contradictory market segment rules will assist businesses by grouping the rules into non-contradictory groups, so it will be less likely that the business considers contradictory rules from conflicting customer segments.

4.1.1 Background

Association rule mining is a useful technique for discovering relationships between items in transactional databases. The relations are provided in the form of implication rules, such as P $\rightarrow$ Q, which in market analysis means that if a customer buys item P, the customer will likely buy item Q. In transactional databases, there are finite sets of transactions, and items, as well as a binary matrix describing what items a transactions contains.

A transaction from a transactional database supports a rule, if that transaction contains both the antecedent (items before the implication) and the consequent (items after the implication) of the rule. Similarly, a transaction from a transactional database contradicts a rule, if that transaction contains the antecedent but not the consequent. This is based on formal logic, as an implication rule is only false when the antecedent is true, but the consequent is false. However, the current trend of increasing data collection poses a problem to using association rule mining in areas such as market analysis. The reason is that more and more association rules will likely be generated with the addition of more information; although the pruning thresholds can be increased to reduce the number of rules, this strategy is not always best as it can lead to eliminating some relevant information. There is also the problem of having little information about the association rules in relation to each other, and this problem increases exponentially as more and more association rules are generated. This is an issue known as information overload, which leads to difficulty in understanding the outcome and hence making decisions using the information generated, because of many different factors, such as the individual being given too much information or contradicting information. Post processing on the association rules is essential to group and prune the rules into non-contradictory sets, to better understand and utilize the association rules without the problem of information overload.

4.1.2 Problem and Solution

Association rule mining has been shown to be useful for various applications, one of which is market analysis. The large number of items and transactions present in a supermarket contributes to this information overload problem for market analysis. In 2008, the average number of items in a supermarket in USA was 46,852 [29], thus the number of possible combinations for items in association rules is enormous. The large number of items along with the large number of transactions could potentially lead to the generation of a vast quantity of association rules, which would be overwhelming to examine.

Another problem is the issue of contradictory association rules. These rules can be generated in a market database due to different customer segments. For example, a mother might always buy milk and cereal, but not bread; and a senior might always buy milk and bread, but not cereal. Association rule mining may generate the two rules milk $\rightarrow$ cereal and milk $\rightarrow$ bread, but these two rules should not be considered together. The reason is because each rule is associated with a different customer segment, and when one rule takes affect, the other does not.

Customer segments are an important factor for business strategies, because a business has a target market that they are catering their products and services to, and it would be more beneficial for businesses to focus on the association rules of their target market. It is difficult to distinguish which generated rules apply to which customer segments, because businesses do not always have the data needed to segment customer transactions, and there are also privacy concerns with collecting and using such information. Furthermore, the potentially large number of rules also presents an obstacle to finding the solution for this problem. A method is needed to partition rules based on the buying behaviors of customers without raising privacy concerns.

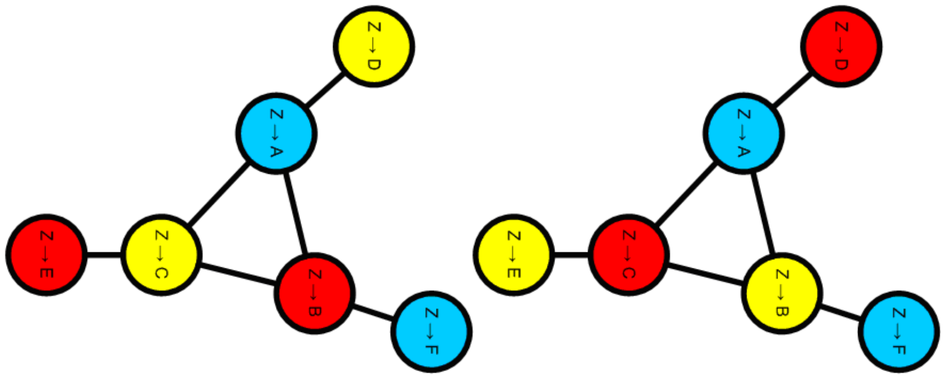


Figure 4.1: Different Graph Colorings (Red and Yellow Sets Swapped)

The solution developed incorporates a fundamental graph coloring like technique; graph coloring refers to coloring the nodes of a graph so that no two connected nodes share the same color [12]. In graph coloring, the problem is finding the minimum number of colors needed to color a graph, or finding the number of different ways in which a graph can be colored with a specified number of colors. This problem originates from the coloring of maps, where the minimum number of colors were sought such that no territories sharing a common border were colored with the same color. Algorithms for this problem can also be used to partition the network into non-contradictory rule sets. The problem with using graph coloring algorithms that color nodes such that no adjacent nodes have the same color for k colors, is that the user does not know the number of contradicting rule sets, so it is hard to decide on how many colors to use. The problem with using graph coloring algorithms that find all the possible sets of colors assignments for k colors, is that the different assignments of colors are not always important as they result in the same set of rules in most cases (see Figure 4.1). There is also the problem that not all the possible rule sets can be found with one number of colors. For example in Figure 4.2, three and four colors are needed to find all the possible combinations of non-contradictory rules such that these sets are maximal, because the set of the three outer nodes cannot be part of only three colorings. Another problem is that all nodes are assigned a color, but this problem is only interested in finding sets of nodes that are not connected. Therefore the graph coloring algorithm is redundant when it assigns some rule sets, as some rule sets are just subsets of a larger non-contradictory rule set. For example, in Figure 4.3 the three inner nodes are part of three sets of one, but these are subsets of the colorings found in Figure 4.1. This is the reason a custom technique is designed to find non-contradictory rule sets.

This solution proposed is a procedure to segment rules into different sets with no internal conflicts. The method establishes all the possible non-contradicting rule sets, then further reduces and combines the different sets to find the final sets of rules that represent the different customer segments. First, a network of association rules is generated, and a graph coloring like algorithm is applied to find all the non-contradicting sets of association rules.

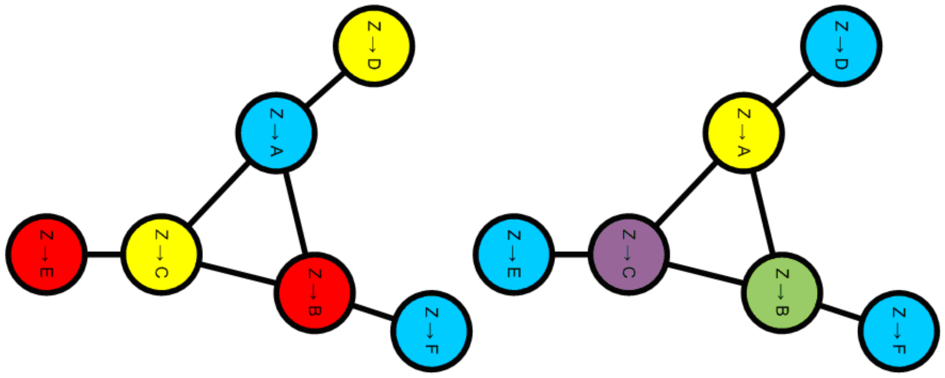


Figure 4.2: Graph Coloring with Different Number of Colors (Three and Four Colors)

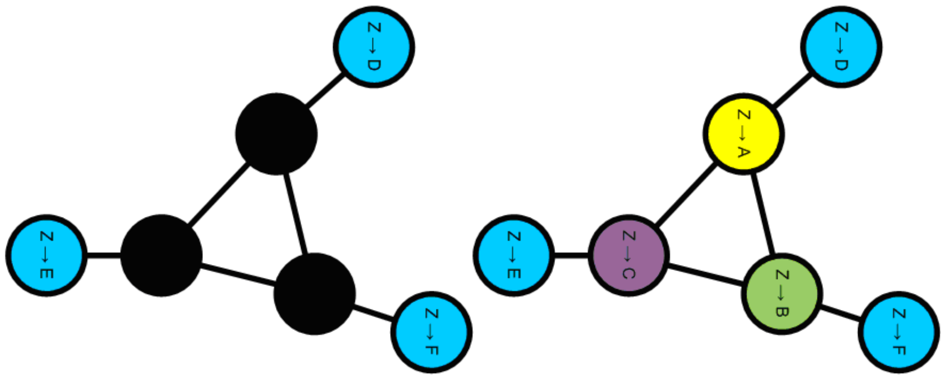


Figure 4.3: Graph Coloring Comparison (Colorings of Inner Three Nodes Unimportant)

The rules and rule sets are mapped to nodes in a network connected based on co-occurrences to determine which sets to prune and combine. The final sets of rules can then be used as classification rules, or further inspected by comparing the current result to previous results to determine changes in customer behaviors.

4.2 Related Work

Researchers have already realized the need for post processing of the reported set of rules. For instance, Baesens, et al. [5] reports about the need for, and importance of post processing techniques for association rule mining, which include reducing the amount of association rules, summarizing the association rules, grouping the association rules, and visualizing the association rules. Liu, et al.'s post processing of associative classification rules using closed sets [43] is an example of a post processing method to reduce the amount of available information by pruning rules. Domingues and Rezendes post processing of association rules using taxonomies [21] is an example of a post processing method that involves summarizing the information. Reducing the amount of available information and summarizing the information has been the main focus of many research projects, but there has been less work on grouping and visualizing the information [5].

Some research on grouping association rules involve clustering the association rules using distances based on features such as support, confidence, lift or the bit-vector representation [59]. There are several algorithms developed for grouping association rules, such as Won et al.'s variation of hierarchical clustering algorithm [62], Gupta, et al.'s agglomerative clustering algorithm [25], and Lent, et al.'s BitOp algorithm [40]. An et al. also provides an objective and subjective algorithm to group association rules, which are similar to clustering algorithms [2]. The clustering algorithms will group the association rules into groups of similar rules, but customer segments will likely have a broad range of rules, therefore the grouping of similar rules is not an effective method to find the rules of different customer segments.

Sung et al. presents a method to predict association rules in different situations by grouping association rules together based on similar characteristics of the customers, and then uses those groupings to sample the correct proportion relative to the new situation [58]. The problem with grouping rules based on characteristics of customers is that not all the information on the customers may be available to separate them correctly, such as lifestyle information which is generally not collected, and difficult to collect as well. Even if collected, it is hard to keep such information up to date because people tend to change their life style based on several factors, including income and the community they live in. This also has privacy issues as many people value personal information, and are reluctant to give it away for fear of uncontrolled usage.

4.3 Proposed Solution

4.3.1 Overview

In general the solution has three phases; in the first phase, a network of association rules is built to find association rules occurring together in the same segments. The main idea is to start off by building a network of association rules, connected based on the logic contradictions in the raw data. This network can than be visualized to give the user a general idea of the status on contradictions between the association rules; such as which rules contradict which other rules. General graph and network properties can also be derived from the network, to give the user more information regarding the contradictions between the association rules.

In the second phase, all the non-contradicting sets of association rules are found by partitioning the network to group rules into contradictory rule sets (a set of rule sets), where the rule sets (a set of rules which are not contradictory to each other) of a contradictory rule set are contradictory to each other, and the user should not consider any of the rule sets together. There can also be multiple contradictory rule sets, and this would mean that rule sets from one contradictory rule set is independent of another contradictory rule set, therefore rule sets from different contradictory rule sets can be considered together.

Afterwards in the third phase, a network of the non-contradicting sets of association rules is built using the links of a generated co-occurrence network to eliminate, and combine the non-contradicting sets of association rules to produce the final sets of rules. The final sets of rules can be checked and matched against all the previously found sets of rules, to determine which sets is the new representation of the old set, so the discrepancies between the newer and older sets can be reported to track the behavioral changes in market segments.

4.3.2 Phase One: Network Construction

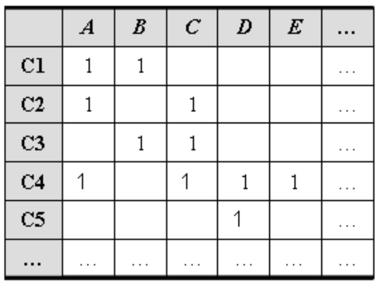
An algorithm is shown to produce a network of the association rules, which are connected

Algorithm 1 Building Contradiction Network of Association Rules

```
INPUT: Set R of rules, Database D, Threshold t0
OUTPUT: Network of association rules G0
for every rule in R do
  Make the a node for the rule
end for
for each transaction in D do
  Find all the rules that transaction supports SR
  Find all the rules that transaction contradicts CR
  for every pair of rules in between SR and CR do
    if there is a link between the pair of rules then
      Increase the weight by 1
    else
      Create a edge of weight 1
    end if
  end for
end for
Decrease all edges by t0
for every edge do
  if weight < 1 then
    Remove edge
  end if
end for
```

based on contradictions in the transactional database (see Algorithm 1). The links in the network hold information on which rules have a contradiction with which other rules, and are used to ensure that contradicting rules are not grouped into the same set. This algorithm is illustrated in the following example; assume there are rules A $\rightarrow$ B, A $\rightarrow$ C, B $\rightarrow$ C, and D $\rightarrow$ E.

The first step is to make each rule a node (see Figure 4.4), afterwards the connections are made by processing the transactions. For instance, transaction C1 supports A $\rightarrow$ B, and contradicts A $\rightarrow$ C and B $\rightarrow$ C, therefore a link is created between A $\rightarrow$ B and A $\rightarrow$ C, as well as a between A $\rightarrow$ B and B $\rightarrow$ C with a weight of one for both connections. For transaction C2, the transaction supports the rule A $\rightarrow$ C, and contradicts A $\rightarrow$ B, as a result the weight of the link between A $\rightarrow$ B and A $\rightarrow$ C is increased to a weight of two. For transaction C3, B $\rightarrow$ C is supported, while the transaction contradicts no rules; accordingly no modification to



	A	*B*	*C*	*D*	*E*	...
C1	1	1				…
C2	1		1			…
C3		1	1			…
C4	1		1	1	1	…
C5				1		…
...	…	…	…	…	…	…

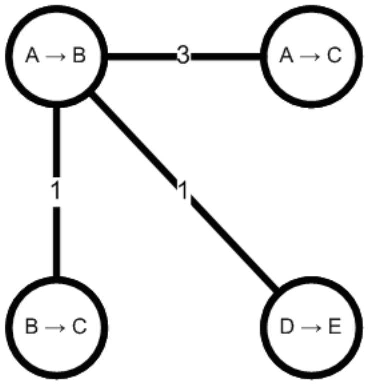


Figure 4.4: Example of a Transactional Database and the Accompanying Network

the network is performed as this transaction gives no indication of contradicting rules. For transaction C4, it supports A → C and D → E, and contradicts A → B, consequently the weight of the link between A → B and A → C is increased to a weight of three; in addition, a link between A → B and D → E is created with a weight of one. For transaction C5, no rules are supported, and even though it contradicts D → E, no changes to the network is needed as this transaction also gives no indication of contradicting rules.

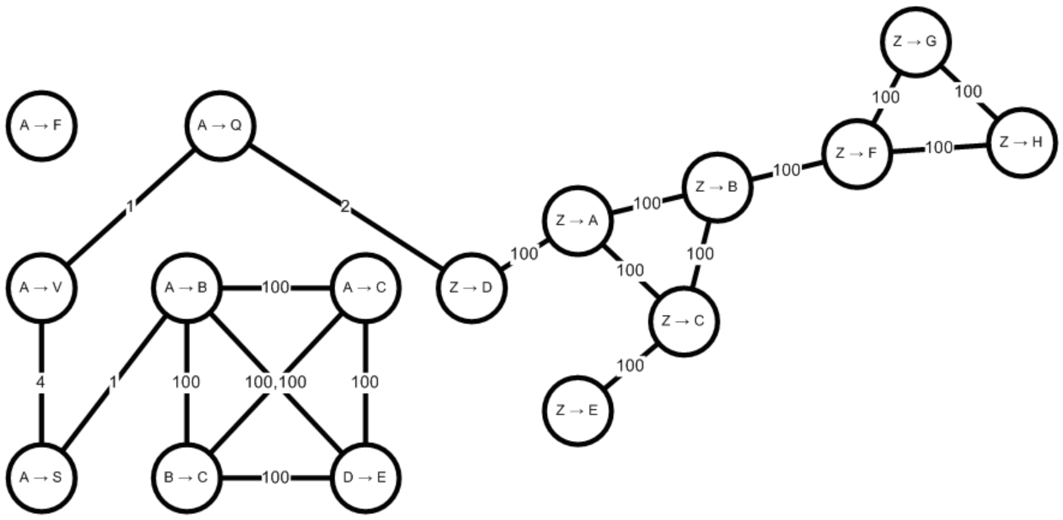


Figure 4.5: Example of a Created Network

After the completion of the above process with all the transactions and rules, a potentially complex network of rules can be generated (see Figure 4.5). The final step is to decrease the weight of every link based on the user specified threshold. This step will eliminate the outlier data to more accurately partition the network into contradictory rule sets. A threshold of four is applied to the network in Figure 4.5, and will change the initial network by deleting four links and isolating several components of the network (see Figure 4.6). The threshold of four was chosen because a few links had a weight of less than four, which were significantly different from the majority of the weights that were at one hundred.

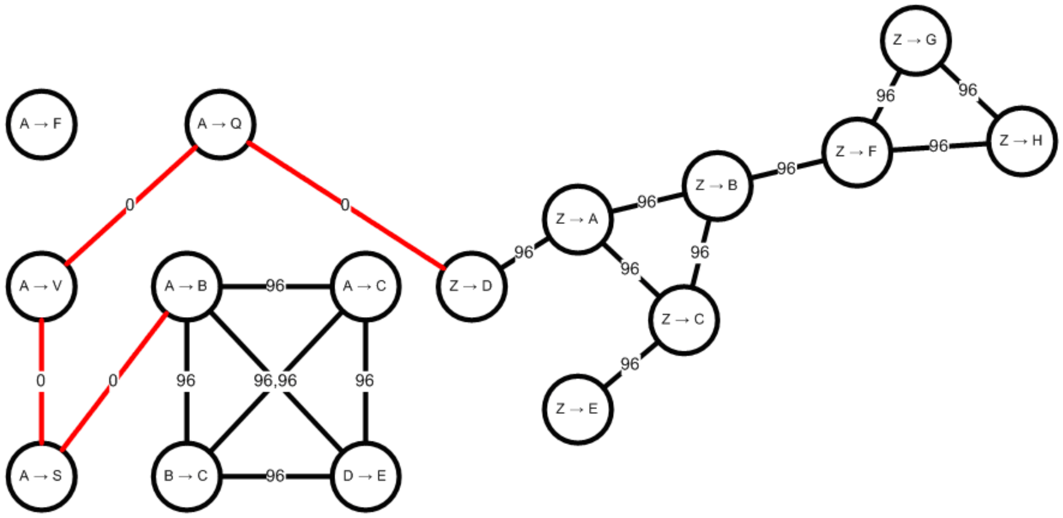


Figure 4.6: Example of a Threshold Applied to a Network

The generated network can provide crucial information visually in extreme or near extreme cases, such as where there are no contradictions found among the rules, in the cases when the network created is totally disconnected, meaning there are no connections between any nodes. This means there are no contradictions among the rules, and all the rules can be considered together. This gives the user the freedom to consider any rules together, but the result is similar to one large group of rules, and does not give the user much more information than was previously present. The other extreme case is one where every rule contradicts every other rule. This occurs when the nodes in the network are linked to every other node, meaning all possible connections between nodes exist. This means that since every rule contradicts each other, then no rules should be considered together. This allows for elimination of many rules from consideration, because the user should only consider one group, thus removing from consideration all other rules. Furthermore, the properties of the network can be computed for further knowledge on the rules and their relationship with one another.

4.3.3 Phase Two: Node Partition

Algorithm 2 Finding All Non-Contradictory Rule Sets

```
INPUT: Network of association rules G0
OUTPUT: All non-contradictory rule sets
for every component in G0 do
    Find the node with the lowest degree and add it to the set N
    for every node in N do
        if node is not assigned a rule set # then
            Assign node the next available rule set #
            Assign neighbors the same rule set # is “not available” (unavailable for assignment)
            Assign non-neighbors the same rule set #
        else
            for each rule set # assigned to the node do
                Assign all neighbors the rule set # is “not available” (unavailable for assignment)
            end for
        end if
        Add all the nodes neighbors to N
    end for
end for
Build # of rule sets based on max rule set #
for each node in N do
    for every rule set # assigned do
        if rule set # is not assigned “not available” then
            Add the nodes rule to the rule set #
        end if
    end for
end for
for each rule set # do
    Create a rule subset
    Consider a sub graph of the nodes where the rule set # is assigned and assigned “not
    available”
    Apply this algorithm to this sub graph
end for
Traverse rule set heirarchy and merge rule sets
```

The rules of the generated network are then partitioned into contradictory rule sets (a set of rule sets). The rule sets are a set of rules with no contradictions with one another, but the rules sets contradict the other rule sets in the contradictory rule set (see Algorithm 2). Rule sets from one contradictory rule set is independent of other rule sets from another contradictory rule set, thus all the combinations of rule sets are found to form all the combinations of non-contradictory rule sets. The algorithm will be demonstrated with a running example; suppose the graphical structure of a network is as shown in Figure 4.7; for partitioning the network of entities, start at the node Z $\rightarrow$ E, because this node has the lowest degree, a degree of one. It is also possible to start at node Z $\rightarrow$ D to end up with the same hidden sets, because Z $\rightarrow$ D also has a degree of one. The algorithm needs to start at the node with the lowest degree, because it is a recursive algorithm that calls itself on the subgraphs. Starting at the node with the lowest degree, allows for omission of the nodes with the least links in the subgraphs, and makes sure that all possible sets are generated.

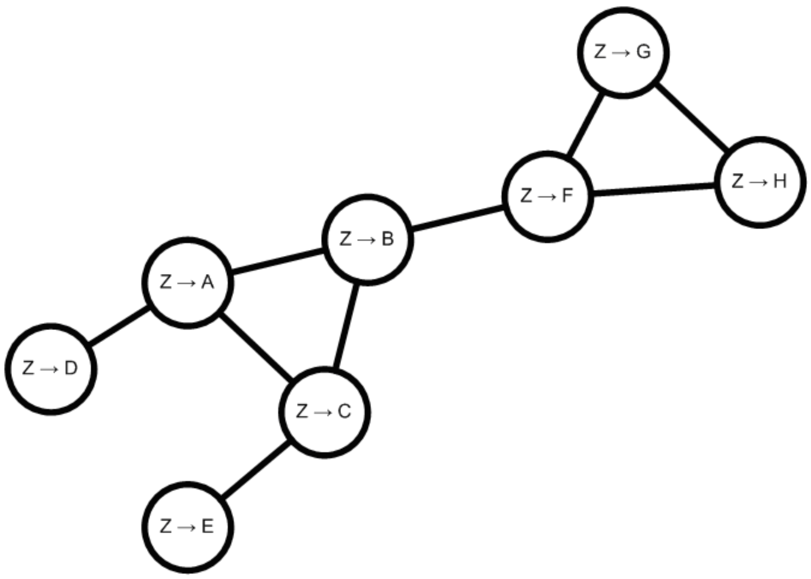


Figure 4.7: Partitioning Network Example

The node Z $\rightarrow$ E does not have a set number assigned to it, so the next step is to assign the node a set number that has not been marked as unavailable for assignment, the first available set number is one. Then assign all of its neighbors the same set number is "not available" (denoted with negative numbers in the figures), which in this case is the assignment of the set number one is "not available" to its neighbor Z $\rightarrow$ C. The assignment of the set number being unavailable to its neighbors will make sure that no connected nodes will appear in the same set. After, assign all of its non-neighbors the same set number one. Assigning the same set number to all of the non-neighbors will make sure that every possible set is generated, because this will add the non-neighbor nodes to the set, or make sure the non-neighbor nodes are considered in the subgraphs for other iterations.

Next, move onto a neighboring node Z $\rightarrow$ C, and since it also does not have a set number assigned to it, assign it the set number two, because the set number one was marked as "not available" for assignment. Then assign its neighbors Z $\rightarrow$ E, Z $\rightarrow$ A, and Z $\rightarrow$ B, the set number two is "not available". After, assign all of its non-neighbors the same set number two.

Look to the next neighboring node Z $\rightarrow$ A, or it is also possible to move onto the node Z $\rightarrow$ B to end up with the same sets; now because Z $\rightarrow$ A already has a set number assigned to it, the only step needed is to assign its neighbors Z $\rightarrow$ D, Z $\rightarrow$ B, and Z $\rightarrow$ C, the set number one is "not available", because this node has been assigned the set number one. This assignment of the assigned set number is "not available" to the node's neighbors in this case will make sure that all the possible subsets, where there are no connected nodes are considered. For the neighboring node Z $\rightarrow$ D, since it already has a set number assigned, its neighbor Z $\rightarrow$ A will be assigned the set number one is "not available", because this node is assigned the set number one.

Now that there are no neighbors for Z $\rightarrow$ D to move onto, we will move onto a previous neighboring node that was skipped, which was Z $\rightarrow$ B. Z $\rightarrow$ B already has a set number assigned to it, so its neighbors Z $\rightarrow$ A, Z $\rightarrow$ C, and Z $\rightarrow$ F are assigned the set number one is "not available", because this node is assigned the set number one. Moving onto a neighboring node Z $\rightarrow$ F, a set number is already assigned to it, therefore its neighbors Z $\rightarrow$ G, Z $\rightarrow$ B, and Z $\rightarrow$ H need to have the set number one and two assigned as "not available", because this node is assigned the set number one and two. Next, move onto a neighboring node Z → G, and since it already has a set number assigned to it, assign its neighbors Z → F, and Z → H, the set numbers one and two are “not available”, because this node is assigned the set number one and two. Repeating again for the next neighboring node Z → H, which already has a set number assigned to it, assign its neighbors Z → F, and Z → G, the set numbers one and two are “not available”, because this node is assigned the set numbers one and two.

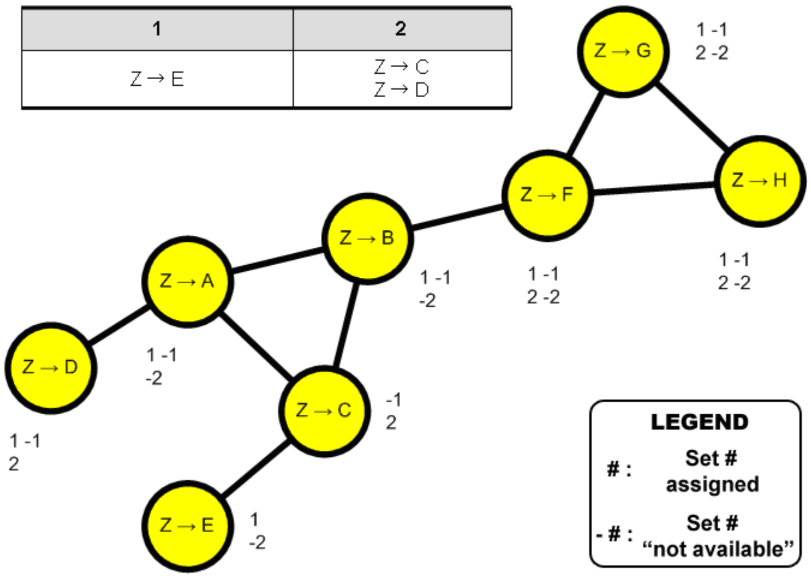


Figure 4.8: First Iteration of Partitioning Network Example

With all the nodes having been traversed, the first iteration is finished, and the algorithm will start building the sets (see Figure 4.8). In the figure, the negative sign is used to denote that a set number is assigned as “not available”. Continuing onward the next step is to build sets for each set number, and assign the entities with nodes that are only assigned one set number to that set number. The entity Z → E is assigned to set one, and the entities Z → C and Z → D are assigned to set two (see Figure 4.8). The other entities also have nodes that are assigned set numbers, but in addition, they are also assigned the same set number is “not available”. They will not be assigned to a set at this stage, because they will be assigned to a subset in following iterations of the algorithm on subgraphs.

At this point, the next step is to consider the subsets for set one by using the same

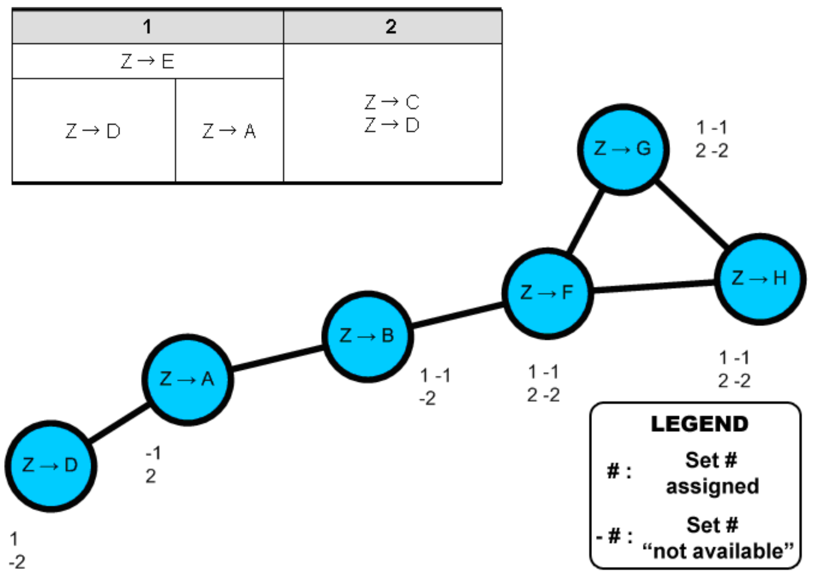


Figure 4.9: Second Iteration of Partitioning Network Example

procedure on the subgraph of nodes that were assigned set number one, and assigned set number one is "not available" (see Figure 4.9). In the case of the second iteration, all the nodes except for Z → E and Z → C are omitted in the subgraph, because we know from the set assignment, whether or not Z → E and Z → C will appear in all the subsets of set number one. The sets generated from the subgraph will be subsets of the original set one (see Figure 4.9). This recursion will continue until the subgraph is empty, and then the process moves to the other sets to begin the same procedure. At the end of the process, a set hierarchy is generated as shown in Figure 4.10. From this we can see that there are nine possible rule sets, which are contradictory to each other. They can be built from the bottom up from Figure 4.10:

1. Z → G, Z → B, Z → D, Z → E
2. Z → H, Z → B, Z → D, Z → E
3. Z → F, Z → D, Z → E
4. Z → G, Z → A, Z → E

5. Z → F, Z → A, Z → E

6. Z → H, Z → A, Z → E

7. Z → G, Z → C, Z → D

8. Z → F, Z → C, Z → D

9. Z → H, Z → C, Z → D

The rule sets have no links between the nodes in each individual set, and therefore no contradictions. Any other possible sets of rules with no contradictions will be a subset of one of the nine rule sets. This makes the contradictory rule sets maximal, and this is important, because it contains the rule sets that hold the most rules with no contradictions. This means there is no possible set of non-contradictory rule combinations that is not covered by one of the rule sets, so the user will not have any missing groups.

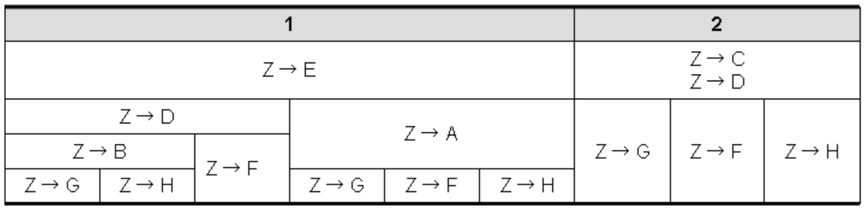


Figure 4.10: Contradictory Rule Set Hierarchy

4.3.4 Phase Three: Partition Refinement

The third phase is split into two parts, the algorithm for the first part starts by creating a node for each rule, and then checking every transaction in the database to link all the rules that occur in the same transaction and updating their respective links (see Algorithm 3). The links are then removed based on the threshold to eliminate links that do not represent rules that occur frequently together in the same transactions. The threshold in this algorithm is an integer input specified by the user. This threshold should be specified such that it eliminates any co-occurrences of rules from random chance; this is similar to the minimum

Algorithm 3 Building Support Network of Association Rules

```
INPUT: Set R of rules, Database D, Threshold t1
OUTPUT: Network of association rules G
for every rule in R do
  Make the a node for the rule
end for
for each transaction in D do
  Find all the rules that transaction supports SR
  for every pair of rules in SR do
    if there is a link between the pair of rules then
      Increase the weight by 1
    else
      Create a edge of weight 1
    end if
  end for
end for
Decrease all edges by t1
for every edge do
  if weight $< 1$ then
    Remove edge
  end if
end for
```

support threshold used for frequent pattern mining. The resulting network is a network where connected nodes represent rules that frequently occur in the same transactions.

The algorithm for the second part utilizes all the non-contradictory rule sets using found in phase two, and creates a node for each of these rule sets (see Algorithm 4). The rule sets are linked together if they contain rules that occur in the same transaction frequently, and this can be checked by looking at the links in the network generated in the first part of phase three. The links are then removed based on which rule sets contradict with each other, and this information is obtained from the contradicting rules sets generated in phase two. The links are also removed based on the threshold to eliminate links that do not represent rule sets that contain rules occurring frequently together in the same transactions. The threshold in this algorithm is an integer input specified by the user. This threshold should be specified such that rule sets with rules that do not occur frequently together are eliminated, and rule sets that do not share rules that frequently occurs together do not share a link. The resulting

Algorithm 4 Finding Rule Sets of Different Customer Segments

```
INPUT: Set R of rules, Database D, Network of association rules G, Threshold t2, Threshold t3
OUTPUT: Set of rule sets representing different customer segments S
Find all non-contradictory rule sets NCR with R and D, and t3
for every rule set in NCR do
  Make the a node for the rule set
end for
for each edge(A,B) in G do
  RSA = all rule sets that contain rule A
  RSB = all rule sets that contain rule B
  for every pair of rule sets between RSA and RSB do
    if there is a link between the pair of rulesets then
      Increase the weight by 1
    else
      Create a edge of weight 1
    end if
  end for
end for
Eliminate all edges between contradictory rule sets
Decrease all edges by t2
for every edge do
  if weight < 1 then
    Remove edge
  end if
end for
for every maximal clique in the network do
  Build a rule set by merging every rule set in the clique and add it to S
end for
```

network is a network where connected nodes represent non-contradicting rule sets that have rules which frequently occur in the same transactions. The last step is to find every maximal clique in the network, and build the final rule sets by merging the rule sets in the maximal cliques, because each maximal clique in the network represents a set of rule sets that share rules which occur frequently together.

The rules sets generated can then be examined and labeled by market research experts with the customer segments they represent. These market segment behaviors can then be used for a variety of purposes, such as using the information to plan business strategies with

Algorithm 5 Finding the Rule Set Representations

```
INPUT: Set of rule sets representing different customer segments S, Set of previous rule
sets representing different customer segments SO
OUTPUT: Discrepancies between rules for current and previous rule sets of different cus-
tomer segments
for every rule set A in S do
    for every rule set B in SO do
        $Similarity(A, B) = |A \cap B| \,/\, |A \cup B|$
    end for
end for
Find the best matching using the Hungarian algorithm
Find the discrepancies between matched rule sets
```

market promotions, classifying customers into the market segments and tailoring services based on their behaviors, or using this data to track changes in market segment behaviors. For example, tracking the changing behviors of market segments can be accomplished by calculating a similarity measure, based on the number of common rules between each rule set from the current set of rule sets found, and the previous set of rule sets found during the last execution of this proposed method (see Algorithm 5). An combination optimization algorithm, such as the Hungarian algorithm can then be used to find the rule sets that are the current representation of the previously found rule sets [37], and this is done by finding the best matching for the rule sets' similarity measures, because the majority of rules for customer segments are likely to remain the same over time with only gradual changes. Any rule sets that have no matches would indicate a new or dissolved customer segment, and new customer segments can be checked against previous customer segments to determine if they are new variations of already discovered customer segments. Discrepancies between the matched sets can then be searched for to inform businesses of the changes in customer behaviors.

4.4 Results

4.4.1 Experimental Method

To validate the proposed method, a simulation has been generated to allow for checking the various aspects considered by the proposed approach. The generated databases and details of the conducted experiments are described next in this section. Groups of customers with contradicting association rules were created, and were assigned a percentage of the population to demonstrate how the proposed method works with a range of different population sizes from majority to minority. In addition, a group with subgroups who all share one same association rule, but also have their own contradicting association rules was created to demonstrate how the proposed method can group these rules together as well.

- Married mothers (30%)
  - milk $\rightarrow$ cereal, pop $\rightarrow$ chips, vegetables $\rightarrow$ fruits
- Single young adults (30%)
  - milk $\rightarrow$ bread, pop $\rightarrow$ vegetables

- Health cautious adults
  - water $\rightarrow$ sport drinks
  - Regularly active (10%)
    - vitamins $\rightarrow$ cereal
  - Athletes (10%)
    - vitamins $\rightarrow$ fruits
  - Professional athletes (10%)
    - vitamins $\rightarrow$ protein shake
- Seniors (10%)
  - water $\rightarrow$ vitamins

Transactional databases with 1000 transactions using the pseudo code in Algorithm 6 along with the customer groups above were created, in an attempt to simulate the buying behavior of the customer groups according to the association rules, while taking into consideration the randomness of customer buying behaviors. Several transactional databases were created with minor changes to the customer groups' behaviors for each transactional database. The first database is created with no changes to the customer segments, and tested with phase one and two. The rule sets generated in the final phase are used as the old rule sets for the other simulation experiments. Four other databases were created for testing by removing a rule from a customer segment, adding a rule to a customer segment, removing an entire customer segment, and adding an entire customer segment, to simulate the buying behavior of the customer segments changing over time. The method for creating the transactional databases that were used in this experiment is detailed in Algorithm 6.

Algorithm 6 Simulating Transactional Databases

```
INPUT: Customer Segments Populations and their Rules
OUTPUT: Transactional Database DS
for 1000 transactions do
  Select a customer segment based on the populations
  Create a transaction
  for every item in the database do
    Add the item to the transaction with a 5% chance
  end for
  for every customer segment's rule's antecedent item do
    Add the item to the transaction with a 50% chance
  end for
  for every customer segment's rule's consequent item do
    if the rule's antecedent is in the transaction then
      Add the item to the transaction with a 80% chance
    else
      Add the item to the transaction with a 50% chance
    end if
  end for
end for
```

4.4.2 Experimental Results

The proposed method was applied on the simulated transactional databases with thresholds of 10 for t1, 1 for t2, and 80 for t3. All the customer segments and their respective rules were discovered. The simulation experiments were successful for all four of the simulated transactional databases. For the modified transactional database with a rule from a customer segment removed, the proposed method was able to detect that the customer segment had lost the rule. For the modified transactional database with a new rule added to a customer segment, the proposed method was able to detect that the customer segment had gained the rule. For the modified transactional database with a customer segment removed, the proposed method was able to detect that the customer segment had disappeared. For the modified transactional database with a new customer segment added, the proposed method was able to detect that the new customer segment had emerged. In all these simulation experiments, the proposed method also discovered all the rules that remained in the customer segments. This demonstrates that the proposed method can produce accurate rule sets, which can be used to discover changes in the customer segment behaviors, or the customer segments in general over time. The simulation experiment is a simple experiment that illustrates the effectiveness of the method when the changes to customer segments are applied.

4.5 Summary

This case study establishes an effective method to reduce the difficulty for businesses to review the association rules of different customer segments, and track the behaviors of market segments based on their buying behaviors. The method established in this case study has the advantage of not needing customer information, thus removing the need for businesses to obtain customer information, and removing the threat of intrusions into customer privacy. The method also generates the rule sets based on conflicting rules, and dividing rules based on customer behaviors is more accurate than customer characteristics, because the behavior is the focus and customers with similar characteristics such as age, income, and education can still have widely different behaviors.

“Data mining applications have proved highly effective in addressing many important business problems” [3]. This has great importance for businesses, because it can help businesses focus on their target market by reviewing the rule set representing the target market. This method can also inform businesses the changing behaviors of their customers, thus allowing for businesses to adjust their strategy to meet the current market conditions. Furthermore, the rule sets generated from this method can be used for classification purposes; accordingly, this method can then be used to update the classification rules over time to ensure accuracy does not drop due to changing behavior.

Chapter 5

CASE STUDY TWO: Finding Hidden Links in Criminal Networks

5.1 Introduction

This case study demonstrates the effectiveness of a unique grouping technique in finding hidden links in criminal networks, which can provide a more accurate picture for investigators to analyze. In this case study, the developed unique grouping technique is an intermediate step in the series of processes. The unique grouping technique is part of phase two of the solution, where the groups are created to determine hidden sets based on the sub-networks created. The unique grouping technique in this case study is based on part of the unique grouping technique in the previous case study (see Chapter 4), which shows that unique grouping techniques can be reused for other problems despite the fact that unique grouping techniques are specifically designed to solve one particular problem. The developed solution in this case study is an important technique in social network analysis, especially for law enforcement agencies as they need to have correct information. Finding hidden links in criminal networks will assist investigators by helping reduce the errors in the current data, so it will be less likely that the law enforcement agencies will make poor decisions.

5.1.1 Background

Social network analysis is an emerging and valuable approach to model and investigate connections between social entities [55]. Social network analysis can be applied to criminal networks, and more specifically this case study will be focused on terrorist networks; though the two are different as each have a different type of target, the same analysis applies to both networks. There has been much research work on criminal networks, and more recently the focus has been on terrorist networks. Social networks can help to elaborate on good strategies to prosecute or prevent criminal activities [6, 36, 39, 48, 50, 51]; for example, the works described by Baumes and Xu constructed social networks to analyze the organizational structure of terrorists groups [8, 63].

Social networks provide a way for law enforcement agencies to analyze criminal activities [42, 16], but here the agencies face the problem of missing links in the network [39]. This problem is known as incompleteness, which Sparrow identified along with two other main problems for using social network analysis for criminal activities [57]. Incompleteness in social network analysis for criminal activities is defined as having missing nodes and/or links in the network. This is due to the law enforcement agencies' inability to uncover every relevant node and link. The problem can arise because criminals may attempt to hide their ties to other criminals or events, in order to minimize their connection with criminal activities [36]. For example, one simple method for criminals to conceal their connection with another person or event is to have a middleman. In this case, there would be a link between the criminals and the middleman, as well as a link between the middleman and the other person or event, but there will be no direct link between the criminals and the other person or event. Furthermore, the criminals can have several middlemen to reduce the weights of the links by spreading their connection with all the middlemen, thus making it less obvious that there is any connection between the criminals to the other person or event. In addition, the criminals can place several middlemen in between themselves and the other person or event, thus further distancing themselves, and making it more difficult for law enforcement agencies to make the connection. These are simple methods which can be achieved by the criminals, but they result in effectively eliminating the link in the social network for analysis by law enforcement agencies. The goal of this case study is to provide a method to help identify hidden links between nodes in a network with the current information available to investigators.

5.1.2 Problem and Solution

Sparrow [57] identified having missing nodes and links in the network as one of the three main problems for using social network analysis to analyze criminal activities. The investigators are just not able to discover all the nodes and links due to criminals attempting to hide their ties to each other, in order to minimize any compromises to the network. An example of this is the network created by Krebs of the 19 dead hijackers of the events during September 11th, 2001 [36]. The network was very sparse, and members on the same team were not directly linked to all other members of the same team. For social network analysis to be more effective for criminal networks, where there are likely missing links, the amount of missing links needs to be reduced.

Although criminals will try to minimize information about the criminal organization to avoid detection, our solution uses the information recorded about the individual, such as where they work, where they have studied, who their friends are, who their family are, what events they participated in, etc. The method will find the number of times two individuals are indirectly connected (connected through a chain of links and nodes) in different relationships. For example, if there is no connection between person A, and person B, but they have a chain of friends in common, attended the same schools, been to the same cities, are members of the same club, etc. This cannot be considered coincidental that entities A and B are not linked together if the number of indirect links is very high. This would mean that it is likely there's a hidden link between A and B, because each indirect connection implies there's a possibility of a hidden link, and a large number will indicate a higher likelihood. It is like each indirect connection is flipping a coin to determine whether there is a hidden link. The more indirect connections, the more coin flips to determine if there is a hidden link, thus this is why there's a higher likelihood of a hidden link if there's more indirect connections.

In this case study, the goal is to provide a method to help identify hidden links between nodes in a network with the current information available to investigators. The solution to this problem is for law enforcement agencies to look at the unconnected entities of a social network, and find the number of different indirect links in the various social networks between the unconnected entities. A higher number of indirect links between two unconnected entities than the average would indicate a potential hidden link, and thus should be further investigated by the law enforcement agency. This however can be a daunting task when the social networks are large, as there can be too many factors to manually keep track of. In this case, an automated solution becomes a very appealing method.

5.2 Related Work

Work on hidden links in networks, such as the work by Baumes and Magdon-Ismail [8, 46], deal with communication networks, but do not take into consideration other types of data to be used for identifying hidden links. Another piece of work published [53], provides an alternative solution to infer missing links, which uses a sampling technique on the network along with Bayes' theorem. Although similar to our solution, the difference is that our solution breaks down the network into many different sub-networks based on relationships, rather than take samples of the network; also, their approach predicts links based on Bayes' theorem, whereas our approach predicts links based on the number of times nodes are indirectly connected in the different sub-networks. In terrorist networks, hidden groups or structures can be mined [60, 45] and social network measures can be used to analyze the structure [16, 18]. Criminals and terrorists will try to keep minimal information from being revealed in order to avoid being traced and dissolved.

The problem of finding these hidden links can be considered similar to the problem of predicting links for the future state of the social network [42]. The main difference is the context of the uncovered link, whether or not the link exists in the current timeframe. Although there are methods that uncover links, which represent either links likely to exist in a later timeframe or hidden links in the current timeframe such as Backstrom and Leskovec's work [4]; this research work focuses only on uncovering hidden links within the timeframe of the network data. Leskovec also has done similar work in predicting positive and negative links [41], and although signed networks are not explored here, the method can still be applied to uncover positive or negative links. This can be accomplish by splitting the network into two with the positive and negative links occurring only in one network, thus when the method is applied to each network, it will uncover hidden positive links in the positive network and hidden negative links in the negative network. Some other research work on link prediction uses a proximity measure to determine the similarity of two nodes in the network to predict links. Research on this technique has involved efficient algorithms to approximate these proximity measures to handle large data sets [56], and also consideration of weights for proximity measures to improve the effectiveness of the prediction method [49]. Classifiers can also be used to predict links by using a set of features extracted from the social network for training in a binary model [26]. Regardless of the methodology, once the hidden links are found and connected, the uncovered network will be ready for use with various analysis techniques [51, 48, 50].

5.3 Proposed Solution

5.3.1 Overview

The general idea for the solution is split into three phases, where the first phase is to use various sources of data to create social networks based on the type of relationships between the entities. This is done by building a main network of all the entities, connected based on whether there is any type of relationship between the entities. The entities are the nodes, and they can be anything, such as people, places, companies, accounts, etc. The relationships are the links, and they can be any type of association, such as friend of, been to, works for, deposited in, etc. In addition, sub-networks will be built with all the entities, connected based on the same relationship types. It is from these social networks that we generate more networks in the second phase that represent all the possible hidden links, and have the links between the nodes represent the number of times the two entities are indirectly connected in different relationship types.

The relationship type of networks can be anything that links two entities together, such as friend of, family of, coworker of, been to, studied at, eats at, participated in, plays in, and member of. The network partitioning algorithm discussed in Chapter 4 is used on the main network to find all the possible hidden groups in the network. These groups can be represented as weighted networks, and the weights can be updated based on the number of indirect links for each relationship type. The higher the weights, the more times the nodes are indirectly connected, thus the more likely there exists a hidden link between them. In the third phase, the links with the highest weights are searched for, which would indicate that the entities do not have a direct link, but do have many indirect links across the different social networks generated based on relationship types. These are the links that are significant, and thus in need of more investigation. At the end, the investigators can create and analyze the hidden links based on different criteria, such as having weights above a certain threshold, or being in the top portion of the highest weighted links.

5.3.2 Phase One: Main and Sub-Networks Creation

The method uses data that is structured to connect two entities with a specific relationship type. The idea is to start by creating a main social network of all the entities, with a link that specifies the different relationship types connecting the two entities. The next step is to split the main social network up into several smaller sub-networks based on the relationship type, such that each relationship type has its own social network, and the links of the sub-network represent that there is a link of that relationship type in the main network. Afterwards, we need to partition each of the sub-networks into groups of nodes that represent potential hidden links in phase two.

Entity	***Relationship Type***	***Entity***
A	Leader of	B
A	Friend of	C
A	Works at	D
B	Friend of	A
C	Friend of	A

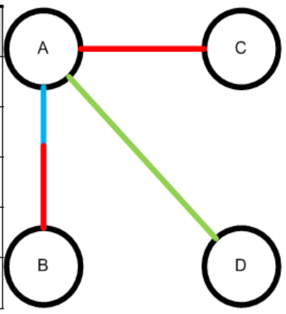


Figure 5.1: Main Network Creation Example

This phase is illustrated in the following example, where the creation of the main social network is achieved by making each entity a node and linking the nodes with the relationship type in the source data. The process is illustrated in Figure 5.1, and with the data presented in Figure 5.1, each entity is made into a node. The next step makes the links by processing each relationship, in this instance there are relationships between A and B, A and C, as well as A and D, so there are links created with A and all the other nodes (see Figure 5.1). The relationships between B and A, as well as C and A, also indicate there are links, but the links are already formed by the previous relationships processed.

Algorithm 7 Sub-Network Creation

```
INPUT: Main network
OUTPUT: Sub-network by relationship types
for each relationship type in data do
  make each entity a node
  for each relationship do
    if the relationship is the same as the relationship type then
      link the two entities together
    end if
  end for
end for
```

The next part constructs the sub-networks based on the main network just created, by dividing the network into separate components, as described in Algorithm 7. To illustrate the process, an example is shown with the data presented in Figure 5.1. For relationship “Leader of”, there is a relation of this type between A and B, so there is a link between A

and B (see Figure 5.2). The other relationships are not of type “Leader of”, so there are no more links to be formed for this sub-network. For relationship “Friend of”, there is a relation of this type between B and A, as well as A and C, so there is a link created between A and B, as well as A and C. The relationship between C and A, also indicates that there should be a link, but that link is already formed by a previous relationship that has already been processed. The other relationships are not of type “Friend of”, so there are no more links to be formed for this sub-network. For relationship “Works at”, there is a relation of this type between A and D, so there is a link between A and D. The other relationships are not of type “Works at”, so there are no more links to be formed for this sub-network.

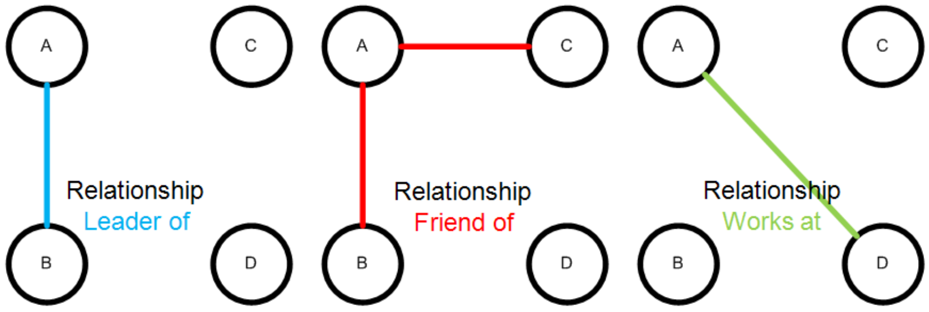


Figure 5.2: Sub-Networks Creation Example

5.3.3 Phase Two: Hidden Network Detection

The method presented in this case generates networks that represent all the possible hidden links, and the links of these generated networks represent the number of times the two entities are indirectly connected in each relationship type. This solution incorporates Algorithm 2 presented in phase two of the previous case study (see Chapter 4). The algorithm concepts are generalized and customized for the problem in this case study (see Algorithm 8), but the implementation of the procedures remain the same. Given a network, represented with a graph $G = (V, E)$, where V is the vertex set and E is the set of edges connecting pairs of vertices. Vertices are referred as *nodes*, *actors*, or *individuals* and edges are termed *links*, *ties*, or *relationships* between nodes in social networks. The method presented in this case

Algorithm 8 Graph Partitioning to Find Hidden Vertex Sets

```
INPUT: Graph $G = (V, E)$
OUTPUT: Subsets of $V$ that are non-contradictory
for each component of $G$ do
  $N$={any node with the lowest degree}
  for each node in $N$ do
    if node is not assigned a set number (denoted as $\aleph$) then
      assign node the next available $\aleph$
      mark neighbors the same $\aleph$ is n/a
      assign non-neighbors the same $\aleph$
    else
      for each $\aleph$ assigned to the node do
        mark $\aleph$ as n/a for all node's neighbors
      end for
    end if
    add all the node's neighbors to $N$
  end for
  partition into vertex sets based on the max $\aleph$
  for each node in $N$ do
    for every $\aleph$ assigned do
      if $\aleph$ is not marked n/a then
        add the node to the set indicated by $\aleph$
      end if
    end for
  end for
  for each $\aleph$ do
    create a subset
    apply the algorithm to subgraphs of all nodes with the same $\aleph$ and nodes marked n/a
  end for
end for
```

makes use of the algorithm to partition the vertex set of a graph into non-contradictory sets, which in this case represents possible hidden groups of nodes. The network partitioning algorithm will return results similar to graph coloring techniques. The specialty of the terrorist network prompts the need for a graph traversal method to effectively find these hidden groups.

Algorithm 9 is applied to find the hidden networks, by partitioning each sub-network into all possible unconnected groups for each component of the sub-networks, which represent the possible hidden links. Subsequently, a set hierarchy with sets of nodes will be generated,

Algorithm 9 Find Hidden Networks

```
INPUT: Main network
OUTPUT: Hidden networks
Apply Algorithm 8 to the main network
for each non-contradictory vertex set do
  make a network with each entity in the set being a node
  create a link between every node of weight 1
end for
```

and from this a depth first method is employed to create the hidden groups. All the hidden groups represent the maximal of all the possible groupings of nodes with no direct links to each other in the sub-network. These grouping of nodes will be referred to as the hidden sets. The partitioning method is also applied to the main network, where all the different relationship types are considered for links. Each hidden group generated by this method can treated as a complete graph, so that all the nodes in the hidden group are connected to all the other nodes of the hidden group with a link weight of 1. These new networks will be referred to as the hidden networks. For example, if the process gets the non-contradictory vertex set $\{B, G, E, D\}$ from the application of Algorithm 8 to the main network, the process will make each entity a node, and create every possible link between each node of weight 1 (see Figure 5.3). This conversion of the set to a network representation needs to be applied to each hidden set.

5.3.4 Phase Three: Hidden Link Weights Computation

The final phase is to update the weights of the links in the hidden networks, which will reflect the number of indirect links between two nodes in a hidden network across the different social networks generated based on the type of relationship. This is achieved by comparing every hidden set generated from each sub-network to every hidden network, and if there is a link in the hidden network, where the two nodes are both in the hidden set, then increase the weight of the link by some value; as long as the weight has not already been increased for that run through of that sub-network hidden set comparison. The value to increment the

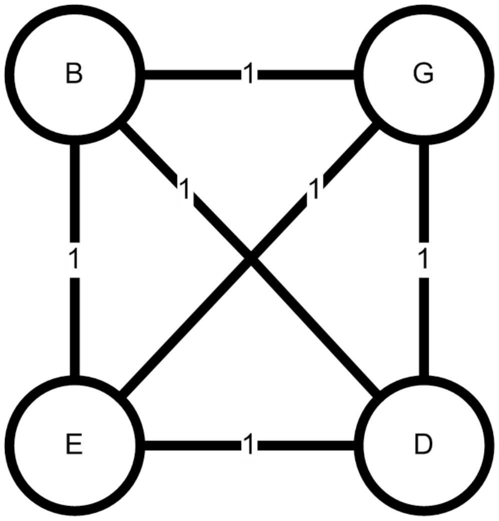


Figure 5.3: Hidden Networks Creation Example

weight is set to 1 in the experiment ran for the results section. When the increment is set to 1 for all situations, then this simply counts the number of times the nodes are indirectly linked in the social networks of relationship types. Although the increment can be different values for different relationship types, in the case where one relationship type should be given more weight if there is an indirect connection with that relationship type. The increment can even become a decrement for relationship types that lowers the likelihood of a hidden link. When the computation of the weights for the links of the hidden networks is complete, then the last step is to locate the links with the highest weights for investigation.

Algorithm 10 Compute Hidden Weights

```
INPUT: Non-contradictory vertex set
OUTPUT: Hidden networks with weights
for each non-contradictory vertex set do
  for each hidden network from Algorithm 9 do
    for all links in the hidden network do
      if the two nodes linked together are in the hidden set, and the link weight has not
      been increased during this run through each sub-network then
        Increase the weight of the link by $\delta$
      end if
    end for
  end for
end for
```

In order to differentiate different relationship types in terrorist networks, Algorithm 10 is used to determine the weight of hidden links. The variable δ can be any number. It can be 1 to count the number of times two nodes are indirectly connected in each network of different relationship types. The variable δ can take positive values to check whether different relationship types have more significance. For example, sub-networks for the relationship type "met" will have $\delta = 1$, whereas sub-networks of the relationship type "friend of" will have $\delta = 2$. This is to signify that indirect connections through being a "friend of" are more likely to have a hidden link, than indirect connections through having "met" the different entities. In addition, the variable δ can also be negative values to take into account if certain relationship types represent that links are less likely between indirectly connected entities. Different values of δ will need sociological studies, and expert involvement to determine the values for the different relationship types.

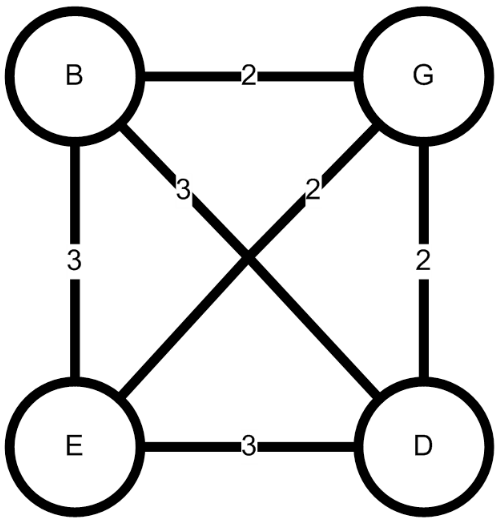


Figure 5.4: Compute Hidden Networks Weights Example

The hidden network presented in Figure 5.3 is used as a reference for the last example, along with the assumption that the process has obtained the hidden sets $\{B, D, E\}$, and $\{B, D\}$ for relationship type 1, and the hidden set $\{B, G, E, D\}$ for relationship type 2. Using $\delta = 1$ in this example will count the number of times unconnected nodes are indirectly linked. The first step here is to increase the weights of the links between B and D, B and

E, as well as D and E by 1, because the entities B, D, and E are in one of the hidden sets for relationship type 1. Relationship type 1 has another hidden set with B and D, but the weight for B and D will not be increased again, because it has already been increased for this sub-network for relationship type 1. The following step will increase the weights of all the links by 1, because the entities B, G, E, and D are part of the hidden set for relationship type 2. The new weights for the hidden network are the ones presented in Figure 5.4.

5.4 Results

5.4.1 Experimental Method

The proposed approach has been applied to small terrorist data sets based on 2002, 2005, London, Madrid, and WTCB information (se Table 5.1) [18]. The value for δ, used in all the relationship types was set to 1. The conducted experiments used the proposed approach to measure the number of indirect links for potential hidden links. The highest weights were then noted to be the most interesting.

Table 5.1: Experimental Data Sets

	2002	**2005**	**London**	**Madrid**	**WTCB**
Event	General	General	General	2004 Madrid Train Bombings	September 11 Attacks
Number of Entities	166	14	31	67	19
Number of Relationship Types	36	13	11	12	14
Number of Links	260	21	44	88	23

5.4.2 Experimental Results

All components of the hidden group networks generated will always be cliques, but with different weights that give information on the likelihood of the hidden link based on the

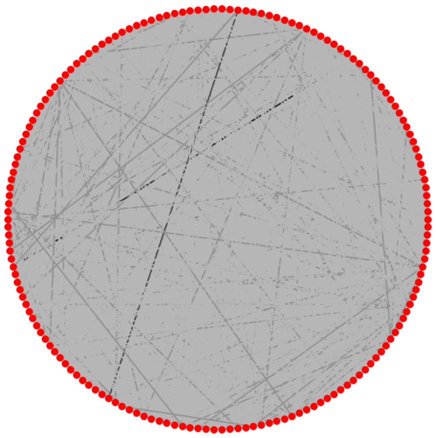


Figure 5.5: Hidden Group Network Example

data. Figure 5.5 shows an a generated hidden network in a circle layout with the links colored to be lighter if they are closer to 1, and darker if they are closer to the highest value based on the 2002 data set. The majority of the cells are light gray, but there are several that stand out. There is a black link of weight 7 and a dark gray link of weight 6, and they indicate interesting links, as the other link weights are quite low in comparison. Figure 5.6 shows a smaller graph visualization of several nodes in the network in Figure 5.5, and this again highlights the interesting link, because there is a large weight of 7, and the other weights are 1 and 2.

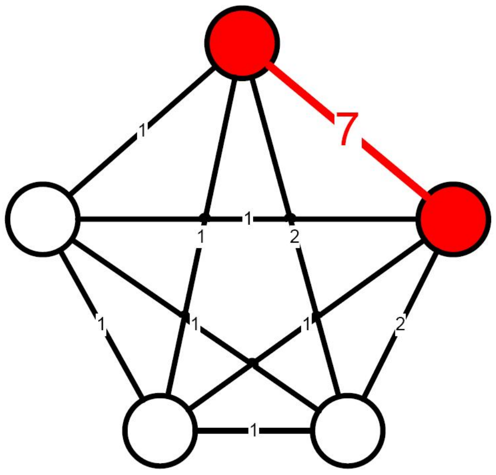


Figure 5.6: Hidden Group Sub-Network Example

For the 2002 terrorist data set, applying the proposed approach generated hidden networks with the majority of the link weights between 1 and 3. There was one hidden link exposed with a weight of 7 between Azahari Husin and Wan Min Wan Mat; thus, this suggests that there is a higher likelihood that these two entities are linked, than other entities based on this data, because the weight of 7 was a very high weight that occurred only once.

For the 2005 terrorist data set, applying the proposed approach generated hidden networks with the majority of the link weights as 1 or 2. There was no hidden link weight that was interesting, thus this suggests that there is likely no hidden links in the data set based on this data, because all the weights were low and remained in the same low range.

For the London and Madrid terrorist data set, applying the proposed approach generated hidden networks with the majority of the link weights between 1 and 3. There was no hidden link weight that was interesting, thus this again suggests that there is likely no hidden links in the data set based on this data, because all the weights were low and remained in the same low range.

For the WTCB terrorist data set, applying the proposed approach generated hidden networks with the majority of the link weights to be 1. There were three hidden links with a weight of 2 between Abd al-Karim Yousef and Abd al-Mun'im Yousef, Konsonjaya and Mohamed Jamal Khalifa, as well as Mohamed Salameh and Sheikh Omar Abdul Rahman; thus this suggests that there is a higher likelihood that these three hidden links exist compared to the other links based on this data, because the weight of 2 was a weight that rarely occurred.

5.5 Summary

The proposed method creates hidden networks that give weights to all the possible hidden links. This provides valuable information for law enforcement agencies, as they can use the weights to help determine what hidden links are more likely to exist for further investigation, and therefore reduce the number of hidden links in their network. The problem with the proposed method is that the solution addresses only part of the problem of incompleteness by identifying hidden links. The problem of missing nodes still remains, and thus there is likely also some hidden links associated with the hidden nodes, which will not be uncovered. The problem of discovering hidden nodes in criminal networks is discussed in the next case study (see Chapter 6).

The proposed method considers all the possible hidden connections, and allows investigators to compare the likelihood of hidden links against other hidden links. In addition, the method also provides flexibility to law enforcement agencies in that it allows for modifications for different scenarios, as the increment weight can be adjusted for different relationship types, by modifying the δ value in the algorithm. This is a valuable technique in criminal network analysis, because it can help investigators find hidden links in the network, and reduce the amount of missing data. The δ value in the algorithm will allow for this method to be used with different types of data, and the weights generated give investigators a measure on the likelihood of the hidden links' existence. This weight can be used in a variety of analysis methods, such as finding the certainty of the links' existence and comparing against other hidden links. Accordingly, this is why the method is an excellent tool for law enforcement agencies to reduce the number of hidden links in criminal networks, which will improve the results of the analysis by addressing one of the major issues for these types of networks.

Chapter 6

CASE STUDY THREE: Finding Hidden Nodes in Criminal Networks

6.1 Introduction

This case study demonstrates the effectiveness of a unique grouping technique for finding hidden nodes in criminal networks, which can supply substantial clues for investigators to consider. In this case study, the developed unique grouping technique is an initial step in the series of processes. The unique grouping technique is the first part of the solution, when the network properties are calculated for the plotting of each node from the two networks; then the plots of the nodes are partitioned into several groups by just grouping nodes with similar properties in relation to the average value of each property measure. In the solution, similar nodes are just partitioned by determining if the node properties are above or below the average for each property measure. This shows that unique grouping techniques can be simple and not overly complex to be an effective solution, which means that unique grouping techniques can be simple to develop and implement. The developed solution in this case study is an important technique in social network analysis, especially for law enforcement agencies as they need to have correct information. Finding hidden nodes in criminal networks will assist investigators by revealing hidden information in the criminal network, so it will be less likely that the criminals will continue to avoid law enforcement agencies.

6.1.1 Background

Social networks can be represented as graphs, where the nodes are the vertexes and the links are the edges. This means that the well researched graph theories can also be applied

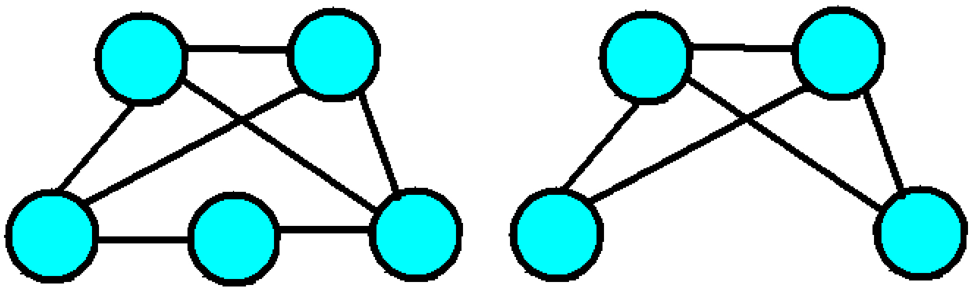


Figure 6.1: Induced Subgraph Example

to social networks. In graph theory, an induced subgraph is a subgraph that contains all possible edges of the supergraph with respect to the vertexes (see Figure 6.1). This means there are no more edges that can be added to the subgraph, such that the graph remains a subgraph. Another important concept for this case study is isomorphism, which is a mapping of vertexes between two graphs, such that if there is an edge between two vertexes in one graph, then there is also an edge between the two mapped vertexes in the other graph (see Figure 6.2).

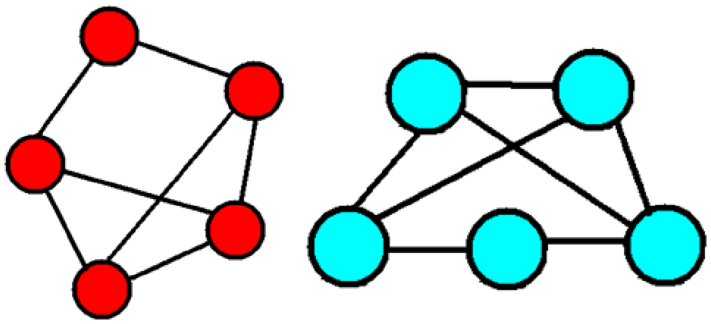


Figure 6.2: Isomorphism Example

The solution established in this case study is based on two social psychological theories, known as the status quo bias and system justification theories. Status quo bias states that people will not be inclined to change an established behavior, unless there is a strong enough incentive to change [47]. System justification theory states that people have a reason to defend and strengthen the status quo [31]. Applying these theories to the individuals in social networks implies that the overall structure of the social network will not change dramatically overtime, unless there is a dramatic event that gives the individuals in the network incentive to change.

6.1.2 Problem and Solution

One of the issues with using social networks for analysis is the problem of having missing nodes in the network. Having missing nodes can significantly impact the results of the analysis, and should be avoided as much as possible [35]. Missing nodes are the result of researchers having incomplete information due to their inability to attain all the information needed, given the uncertainty of which entities to include in the network, or the entities being hidden.

Social network analysis can be applied to criminal networks to elaborate on good strategies to prosecute or prevent criminal activities [39]. For social network analysis to be more effective when applied to criminal networks, where there are likely missing nodes, the amount of missing nodes needs to be reduced. The problem of having missing nodes in a network is an important problem in criminal network analysis. Incompleteness is identified by Malcolm Sparrow as one of the three main problems for using social network analysis to analyze criminal activity [57]. Incompleteness is having missing nodes and links in the network, because the investigators are not able to discover all the nodes and links during the initial investigation phase. Investigators also cannot keep investigating to include all entities found in the network, because there is the uncertainty of when to stop looking for missing nodes; furthermore, networks can become inconceivably large if all entities are included, as there can be links connecting insignificant entities. This is the problem of fuzzy boundaries that Malcolm Sparrow has identified as another one of the three main problems for using social network analysis to analyze criminal activity [57].

This case study provides a method to help identify hidden nodes in a network using previously collected social network data. The method maps the nodes in the network to the nodes in a past network using the similarity of various measures from social network analysis. The mapping also provides a confidence value to help determine the likelihood of a missing node, and the confidence in the overall mapping. The social network measures of the nodes that are not mapped, or have a low confidence value, indicate the importance that the missing nodes would likely have. This information will help investigators identify the importance of the missing nodes, and decide whether the missing nodes are significant enough to spend the time to uncover or not.

The proposed method for finding hidden or missing nodes in social networks is done by an assignment of nodes from an old network to the current network, based on the properties of the nodes, such as the betweenness, closeness, degree, and eigenvector centrality measures. The nodes can be modeled as points on Cartesian coordinate system, and the idea is that similar nodes will stay close together if there are slight differences in the networks, due to the hidden or missing node element. There will be two sets of points S_1 and S_2 from each of the networks, and each point in S_1 can only be mapped to one point in S_2. The points are assigned such that the sum of the distances between the assigned points is minimized. This assignment will imply what nodes in the first network are the same nodes in the second network, and the bad assignments or no assignments will indicate missing nodes.

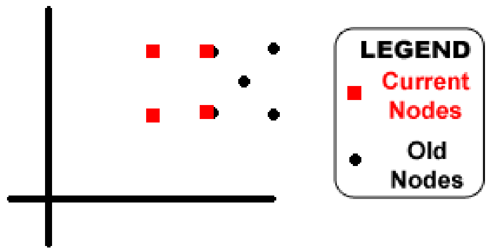


Figure 6.3: Incorrect Assignment Example (Translation)

There is a problem that a missing node and/or additional nodes can produce too much of a shift in the properties of all the nodes in the network, such as the betweenness, closeness, degree, and eigenvector centrality measures. This can reposition their points in a certain direction, thus making it difficult to find the correct assignment of points based on these properties, such as the shift in degree from a missing node that is connected to every other node. For example by considering the information in Figure 6.3, the minimum assignment would result in an incorrect assignment, because the center black point represents the missing node, but will be part of the optimal solution, thus it will be incorrectly assigned to another node, instead of being indicated as the missing node. The solution to this is to translate the points to the right, and this is why translation is considered in the solution. There can also be the possibility that a new node will produce slight shifts in some nodes, but not others, such as the degree of a missing node that is connected to several other nodes, but the new node's point location may be closer to another node's previous point. For example, by considering Figure 6.4, the minimum assignment could result in an incorrect assignment, because the center bottom red point which represents a new node, may be closer to the center black point than the center top red point, which would be the correct assignment. The solution to this is to scale the points along the y-axis of the top red point, and this is why scaling is also considered. The two problems can also occur together, which is why both translation and scaling are considered together in the solution.

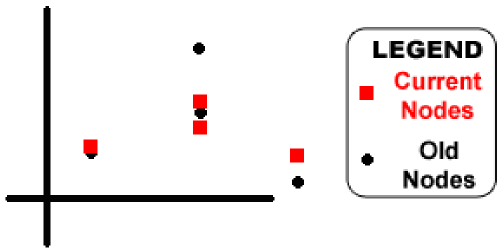


Figure 6.4: Incorrect Assignment Example (Scaling)

6.2 Related Work

Finding hidden nodes in social networks is a hard problem due to the difficulty of determining if a node is missing, because networks can take any shape and can be significantly different for each situation. Kim and Leskovec published a solution that uses the Kronecker graphs model as the basis for the structure of real world networks [32]. The information from the available network is used in the Kronecker graphs model to produce a completed network, which is then divided into the observed part of the available network, and the unobserved part of the missing nodes and links; the two parts are linked based on a probabilistic model. This is similar to the proposed solution which tries to fill in the missing pieces by comparing the current known network to a previously uncovered network. The idea of comparing the structures of networks to fill in the missing blanks is also used by Clauset. Clauset proposes that networks are likely to have a hierarchical structure, where nodes "divide into groups that further subdivide into groups of groups, and so forth over multiple scales" [15, 38]. This hierarchical network structure is used as a basis for predicting links once the available network is fitted to this model.

There is a related graph problem that is similar, and can be seen as an alternative solution to the method proposed in this case study. The related graph problem is the maximum common subgraph isomorphism problem, which takes two graphs as inputs, and outputs the largest induced subgraph of one graph which is isomorphic to a subgraph of the other graph (see Figure 6.5). This graph problem can be applied to solve the problem of missing nodes by employing the current social network and a past social network. The output can then be examined, and the measures of the nodes of the past social network that are not part of the subgraph can help indicate the importance of the missing nodes. The problem with this approach is that the structure of the social network needs to remain the same and cannot change even slightly, except for the missing nodes. A change in the structure of the social network will mean that using this approach will not yield the most optimal mapping, and many nodes may be said to be missing when in fact they are not (see Figure 6.6).

For the challenge of repositioning the points representing the nodes, there is a similar problem in the field of pattern matching and computer vision called the absolute orientation problem; where there are two sets of points and the objective is to find the similarity transformation, which includes translation, scaling, and rotation, that would return the least

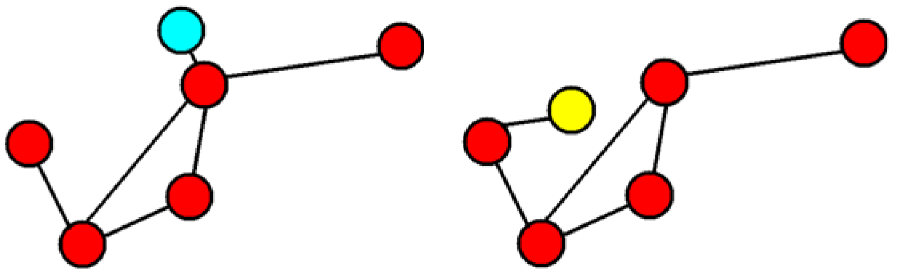


Figure 6.5: Maximum Common Subgraph Isomorphism Problem Example

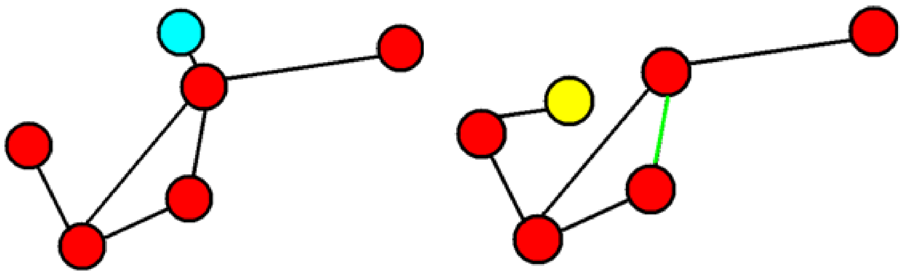


Figure 6.6: Maximum Common Subgraph Isomorphism Problem Issue

mean square error between the two sets of points [61]. There are various types of algorithms developed to solve this problem with different performances, and there is also a variation of the problem where instead of seeking a similarity transformation, a rigid transformation is sought to maintain the size and shape, containing only translations, rotations, or both [44]. Shih-Hsu Chang et al. also proposes an alternative algorithm with a limit on the transformation, such that it only gives transformations of scaling, and/or rotation, which has the advantage of not considering the noise effects of translations [13].

There is also a similar problem addressed in statistics called Procrustes analysis for biological applications, such as analyzing bones [54]. In Procrustes analysis, shapes of objects are considered for comparison, and landmark points of the shape are selected to represent the shape; then the transformation, which consists of translation, scaling, rotation, and reflection, is found by determining the transformation with the minimum sum of squared errors between the two objects [24]. As with the absolute orientation problem, there are also variations with Procrustes analysis, such as the common variant where only scaling is considered in the transformation.

Current solutions to the problem look for the optimal minimum or close to it, but what is needed in this case study is a approximation to transform the points close to their original position, so that the algorithm can match the correct nodes together. The reason for this is because the two sets of data may not have the exact same shape, but only an approximation is needed to reduce errors during the assignment phase, seeing as a few missing nodes in a large network is not expected to radically change the social network structure. Furthermore, the current solutions incorporate rotation or reflection, which is not necessary in this problem, because the missing data in this problem does not give rise to differences that can be solved by rotation or reflection. Many of the solutions developed also specifically address 3-D and 2-D data, but this problem needs to be able to deal with any number of dimensions, because considering different combinations with the properties of nodes may lead to better results depending on the situation.

6.3 Proposed Solution

6.3.1 Overview

In general, the main idea is to take a social network and match its nodes to the nodes of a older version of that social network, where most of the overall structure of the social network is preserved. This mapping is completed by using various measures of the social network as distance measures, and adjusting the measures for the nodes of the network to better fit the measures for the nodes of a previous social network. Afterwards, confidence measures on the mappings can be calculated to show the confidence in the mappings, as well as the likelihood of missing nodes. The measures and connections of the nodes from the previous social network with no mapping would then indicate the importance, and links of the missing node.

This approach can only be applied to stable systems. Stable systems are where the nodes and links of the social network will change minimally over a significant period of time,

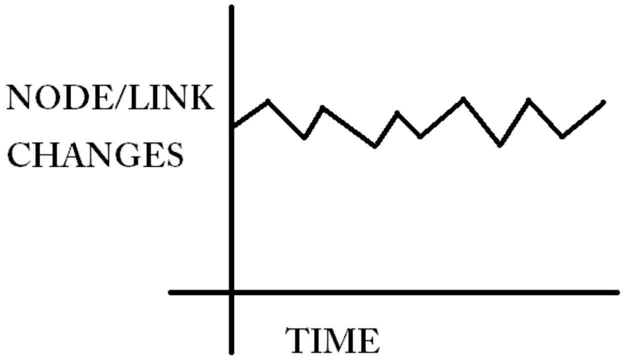


Figure 6.7: Stable Network Plot

thus the overall structure of the network will remain quite similar to the original state (see Figure 6.7). For example, the structure of a business organization network will likely remain the same over time with employees coming and leaving, because the job positions will still be there. A few links and nodes may appear and disappear due to factors such as unemployment or training. The approach should not be applied to unstable systems. Unstable systems are where the nodes and links of the social network suddenly change dramatically, thus the overall structure of the network will be significantly different from the original state (see Figure 6.8). For example, the structure of a business organization network will likely be significantly different after a restructuring of the whole company, because many employees will be coming and leaving for new job positions. There will be many links and nodes appearing and disappearing, because of factors like merging or new ownership.

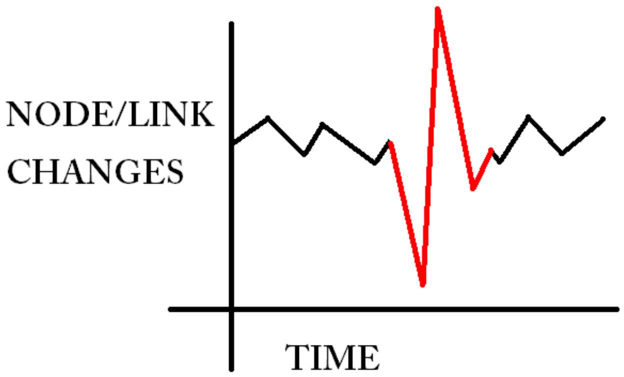


Figure 6.8: Unstable Network Plot

6.3.2 Mapping Method

Algorithm 11 Mapping of Social Networks

```
INPUT: oldNetwork, newNetwork
OUTPUT: Mapping of nodes with similarity measures
Calculate measures for both networks
Adjust_Measures(set of oldNetwork measures, set of newNetwork measures, 0) //Algorithm 12
for every oldNetwork node do
  for every newNetwork node do
    Calculate distances of network measures
  end for
end for
Find the best matching using the Hungarian algorithm
```

Algorithm 12 Adjust_Measures

```
INPUT: Set of measures $S_O$, Set of points $S_N$, k level
Transposition = Center($S_N$) - Center($S_O$)
Partition_Points($S_N$, $S_O$, 0)
//Algorithm 13
Apply transposition and scaling to newNetwork measures
```

The mapping method is detailed in Algorithm 11, and in this solution, the main idea when adjusting the measures is to treat the nodes as points with their measures as the dimensions. The method uses the difference between the two sets' centers to determine the translation, and then partitions the points of each set into similar components that make up the shape of each set. Each set's partitions will be compared to their corresponding partitions of the other set, and the scale will be determined by finding the scaling needed to scale a partition's center to its counterpart, and this will be weighted by the number of points in both partitions in relation to the total number of points. Ultimately, each partition is stretching the other set's points in a certain direction, and these are all averaged together using the density of each partition as the weights.

The points in each set are partitioned using a concept based on orthants. In two dimensions, the plane can be divided up into four quadrants, which represent the entire positive

Algorithm 13 Partition_Points

```
INPUT: Set of measures $S_O$, Set of points $S_N$, k level
if $S_N$ and $S_O$ are not empty then
  for each point in $S_N$ do
    Find it's relative orthant i
    Add the point to $S_{Ni}$
  end for
  for each point in $S_O$ do
    Find it's relative orthant i
    Add the point to $S_{Oi}$
  end for
  for each relative orthant i do
    if level is not reached then
      ParitionPoints($S_{Ni}$, $S_{Oi}$, level+1)
    else
      Portion = $|S_{Ni} + S_{Oi}|$ / $|S_N + S_O|$
      OrthantScale = (Center($S_{Oi}$) − Center($S_O$)) / (Center($S_{Ni}$) − Center($S_N$))
      Scaling = Scaling + OrthantScale * Portion
    end if
  end for
end if
```

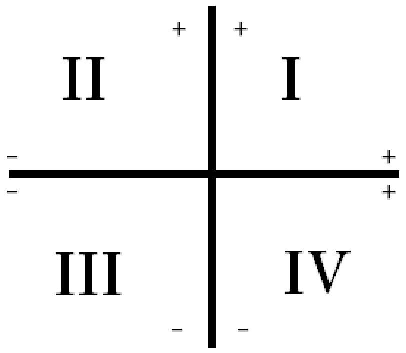


Figure 6.9: Quadrants (2-D Orthants)

and negative set of combinations for the dimensions (see Figure 6.9). Octants are the same idea applied to three dimensions, with eight divisions, each representing a unique combination of positive and negative values for the dimensions (see Figure 6.10). The orthant is the n-dimensional concept of the idea, with 2^n orthants. The points in each set will be partitioned into their relative orthants. Relative orthants for a set of points are found by treating the center of the set of points as the origin, or translating the center of the set of

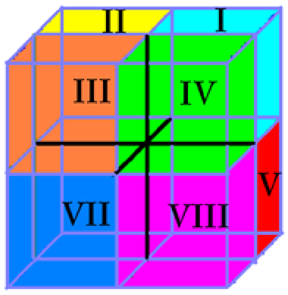


Figure 6.10: Octants (3-D Orthants)

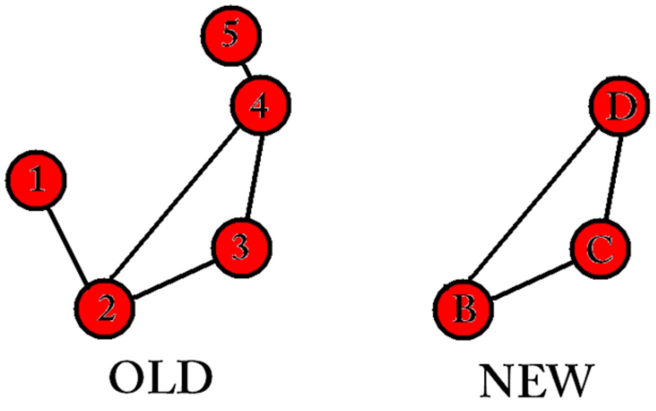


Figure 6.11: Mapping Example - Inputs

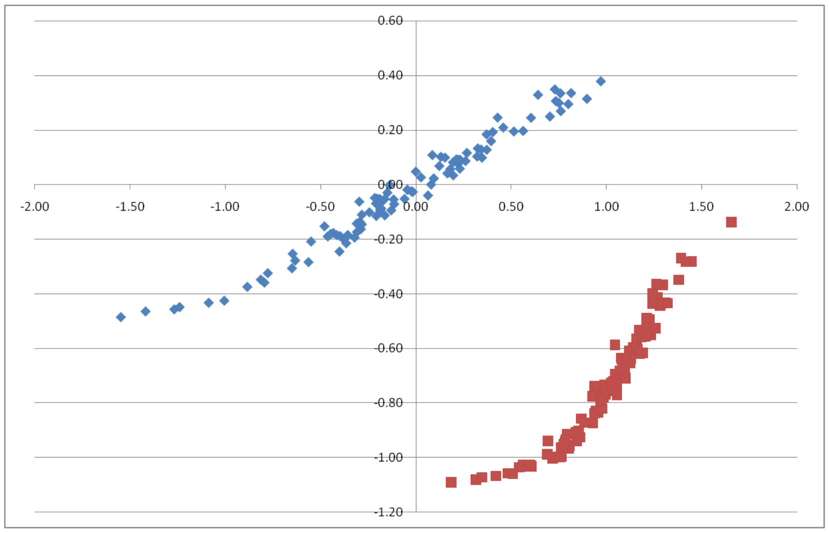


Figure 6.12: Proposed Method Example Sets

points to the origin, and then determining which orthant the points reside in.

An example of the mapping method would start with the calculation of all the associated measures of both networks (see Figure 6.11). The proposed method will treat the nodes from each network as points on a Cartesian coordinate system with their measures as the values of the numerical coordinates. For example, it will find the translation to translate the red set of points (network nodes) to the blue set of points (past network nodes) in Figure 6.12, by finding the center point of each set (see Figure 6.13), and calculating the difference between the two center points to determine the amount needed to translate the red set of points to the blue set of points.

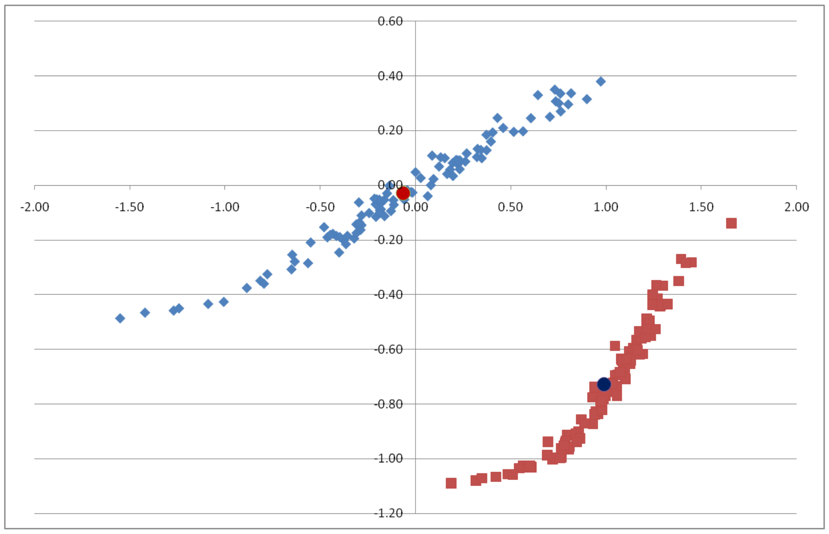


Figure 6.13: Centers of Sets by Considering the Network in Figure 6.12

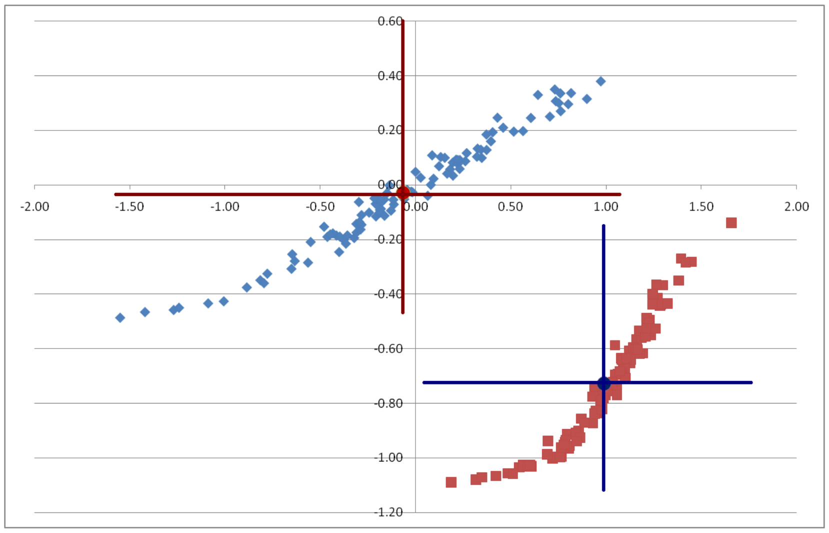


Figure 6.14: Set Partitions for First Level

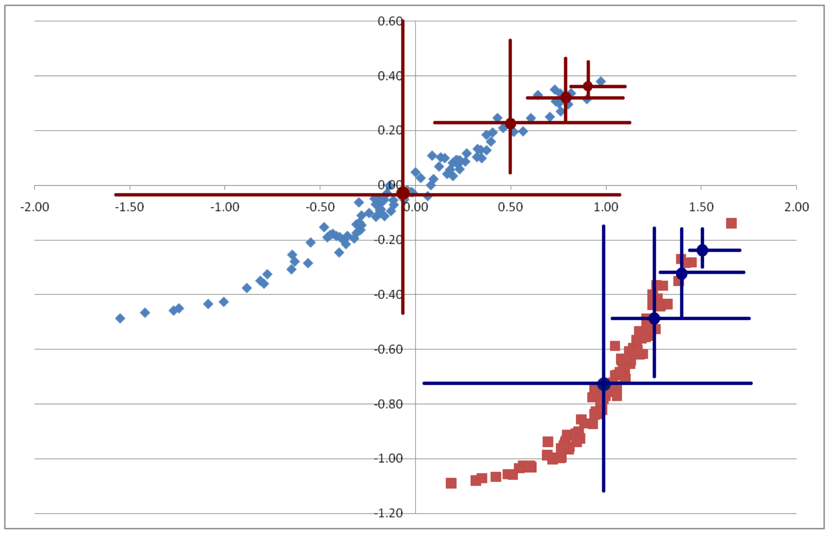


Figure 6.15: First Set Partitions for $k = 4$

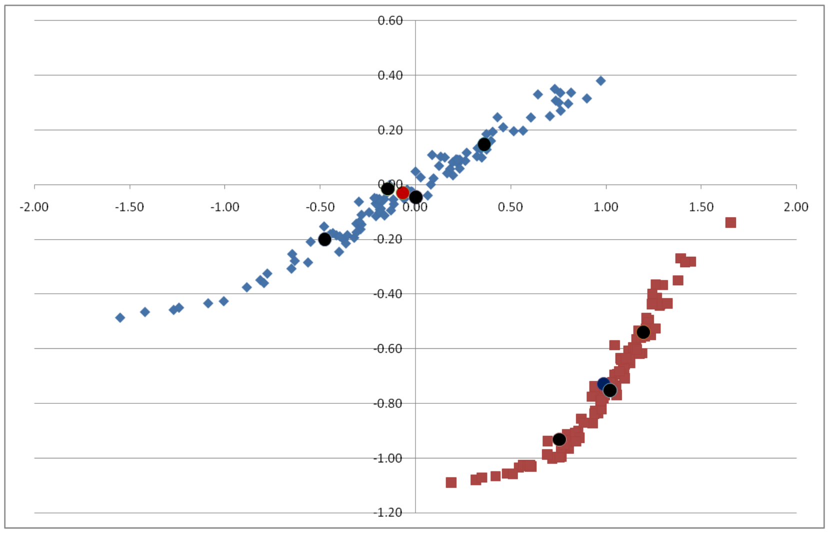


Figure 6.16: Set Partition Centers for $k = 1$

Afterwards, the proposed method will find the non-uniform scale needed to stretch the red set of points to become similar to the blue set of points. The method will partition the points into their relative orthants (see Figure 6.14), and will continue to repeat this step on each partition for $k - 1$ more repetitions (see Figure 6.15). The method then finds the center point of each partition (see Figure 6.16), and ignores partitions with an empty set of points. The corresponding center points of the two sets are used to calculate the scale, and center points with no other corresponding point from the other set are ignored.

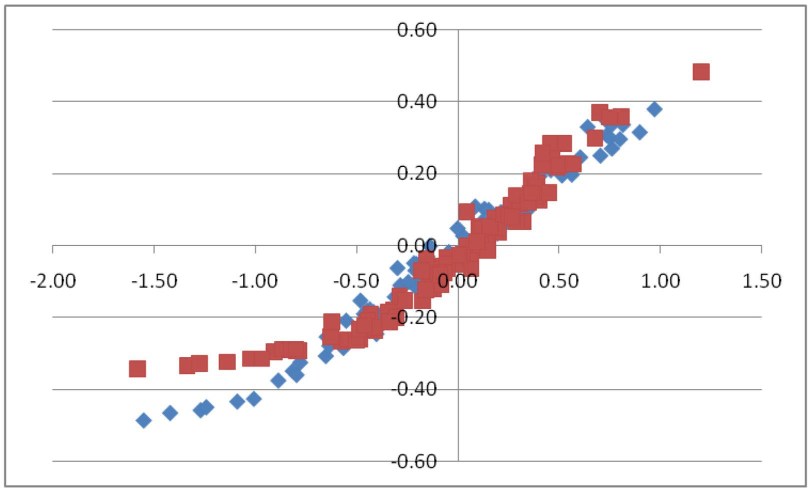


Figure 6.17: Transformation Result

Corresponding points are center points from the two sets that were determined in the same relative orthants. Applying the translation and scaling to the red set of points will result in a transformation that keeps the structure and orientation of the red set of points the same, while shifting it to fit with the blue set of points (see Figure 6.17). The adjustment of the network measures is then complete.

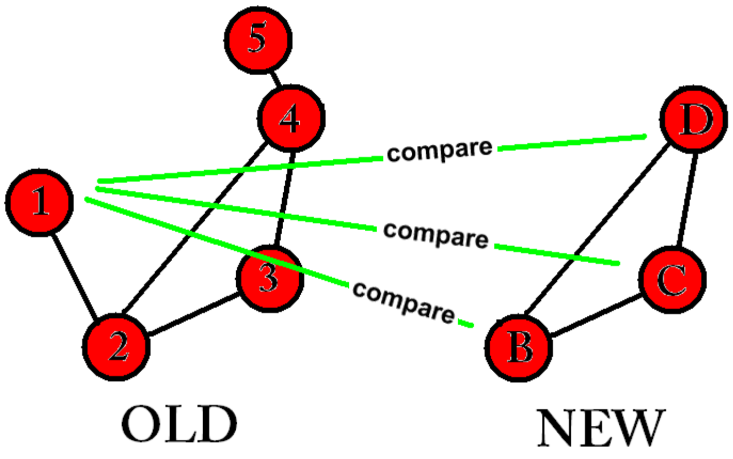


Figure 6.18: Mapping Example - Distance Calculations

Using the calculated and adjusted measures, the distance for every pair of nodes in the past network and current network would be calculated and stored; this step will compare the nodes between the two networks to determine their similarity (see Figure 6.18 and 6.19).

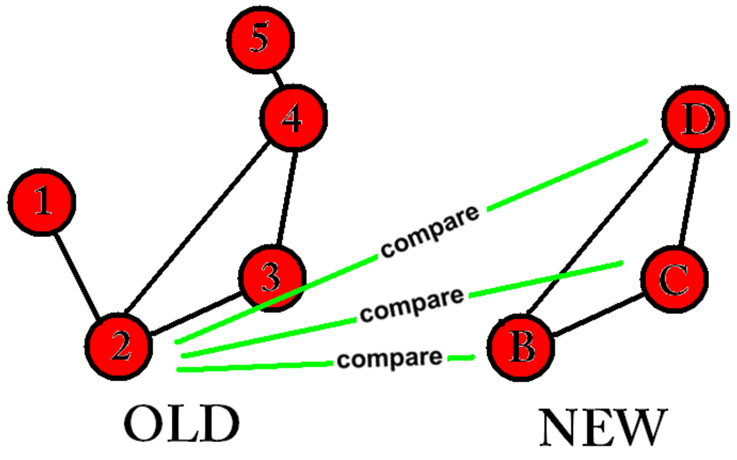


Figure 6.19: Mapping Example - More Distance Calculations

The solution then uses the Hungarian algorithm to find the best mapping of nodes using the distances calculated between the past network nodes, and the current network nodes. The output from the algorithm will be a mapping of nodes from the current network to the previous network (see Figure 6.20). The nodes from the previous network that are not mapped (nodes 1 and 5) are the missing nodes from the network. The missing nodes social network measures will tell investigators their importance to help determine if they are worth the effort of further investigations, and the missing nodes connections will help give investigators leads on what links likely exist for the hidden nodes found.

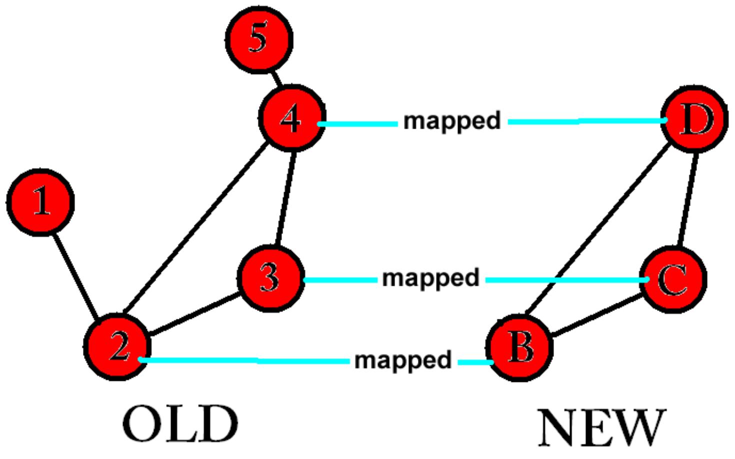


Figure 6.20: Mapping Example - Output

6.3.3 Measures

There are eight measures used in the distance calculation for this case study, which include the betweenness centrality (normalized to be between 0 and 1), closeness centrality (brought down to be 0 if it is infinity), degree centrality, and eigenvector centrality. The other four measures are the averages of all the node's neighbors' centrality measures to better differentiate between nodes and more accurately match nodes. The eight measures all have values between 0 and 1, which makes all the centrality measures equal; meaning no one centrality measure carries more weight than another in the distance calculation. The maximum distance will be between a node that has 0 on all centrality measures, and a node that has 1 on all centrality measures ((0,0,0,0,0,0,0,0) and (1,1,1,1,1,1,1,1)), hence the maximum distance is 2.83 ($\sqrt{(1-0)^2+(1-0)^2+(1-0)^2+(1-0)^2...}=2.83$). As a general rule for normalized measures, the maximum distance is the square root of the number of measures ($\sqrt{\text{number of measures}}$). This maximum distance can be used to calculate a similarity measure for the entire mapping produced, or for each individual mapping. The similarity measure for the entire set of mappings can be calculated using Equation 6.1.

$$\frac{\text{number of mapped nodes} * \text{maximum distance} - \sum \text{distance}}{\text{number of mapped nodes} * \text{maximum distance}} \tag{6.1}$$

This gives a confidence measure of the entire set of mappings assigned, which can be used to indicate the probability that the mappings are correct. In the case of a 100% similarity measure for the entire set of mappings, this means that the two social networks are isomorphic. The similarity measure for a individual assigned mapping can be calculated using Equation 6.2.

$$\frac{\text{maximum distance} - \text{distance of the mapping}}{\text{maximum distance}} \tag{6.2}$$

This gives a confidence measure of a individual mapping assigned, which can also indicate the probability that the mapping is correct. In the case of a low percentage similarity measure for a individual mapping, this means that there is a high likelihood that the mapping is incorrect, and the mapped past network node indicates a missing node.

6.4 Results

6.4.1 Adjust Measures Method

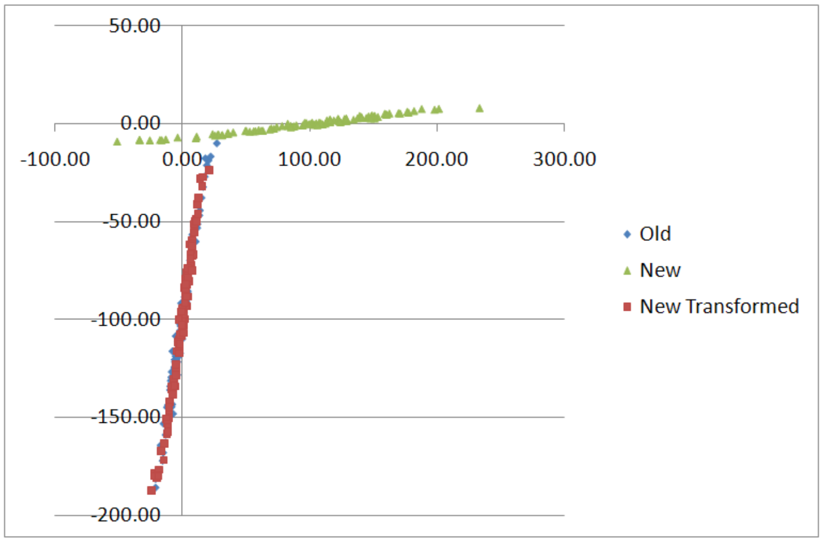


Figure 6.21: Final Transformed Set of Points for Experiment 1

Three experiments were conducted in two dimensions for easy visualization, with randomly generated data based on probability distributions that were 99% correlated for the point dimensions. The result for the first experiment is shown in Figure 6.21, and consists of 100 points for S_O (blue points) and S_N (green points), along with the final transformed S_N (red points). The x dimension of S_O was generated using a normal distribution with a mean of 0 and standard deviation of 10, and the y dimension was generated using a triangular distribution with a minimum of -200, a likelihood of -100, and a maximum of 0. The x dimension of S_N was generated using a normal distribution with a mean of 100 and standard deviation of 55, and the y dimension was generated using a triangular distribution with a minimum of -10, a likelihood of 0, and a maximum of 10.

The result for the second experiment is shown in Figure 6.22, and consists of 500 points

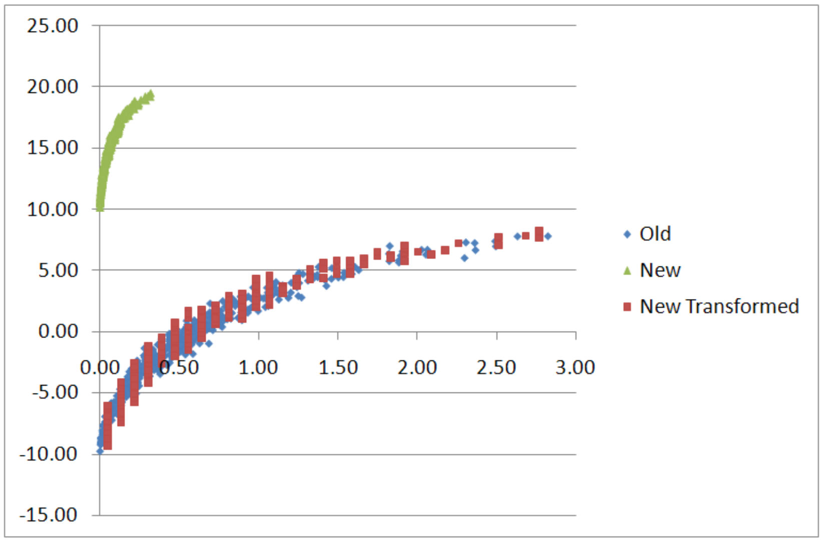


Figure 6.22: Final Transformed Set of Points for Experiment 2

for S_O (blue points) and 505 points for S_N (green points), along with the final transformed S_N (red points). The additional 5 points in S_N represent possible additional new nodes. The x dimension of S_O was generated using an exponential distribution with a rate of 2, and the y dimension was generated using a beta distribution with a minimum of -10, a maximum of 10, an alpha of 2, and a beta of 3. The x dimension of S_N was generated using an exponential distribution with a rate of 10, and the y dimension was generated using a beta distribution with a minimum of 10, a maximum of 20, a alpha of 2, and a beta of 3.

The result for the third experiment is shown in Figure 6.23, and consists of 1100 points for S_O (blue points) and 1000 points for S_N (green points), along with the final transformed S_N (red points). The missing 100 points in S_N represent possible hidden or missing nodes. The x dimension of S_O was generated using a Poisson distribution with a rate of 100, and the y dimension was generated using a uniform distribution with a minimum of -1000, and a maximum of 1000. The x dimension of S_N was generated using a Poisson distribution with a rate of 100, and the y dimension was generated using a uniform distribution with a minimum of -1, and a maximum of 1. The results show that the adjust measures method does a reasonably good job in shifting and stretching one set of points to fit another set, as Figures 6.21 - 6.23 show with the final transformed set of points (red points) mostly

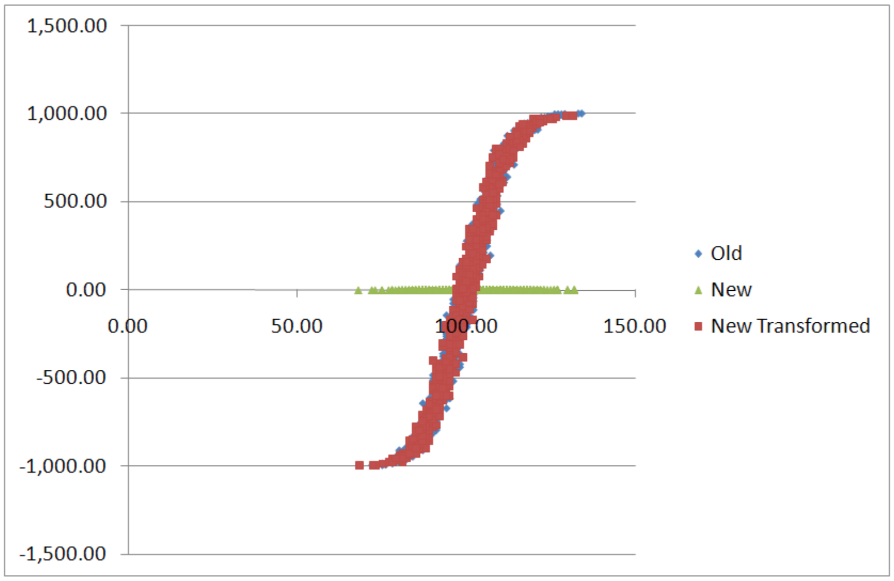


Figure 6.23: Final Transformed Set of Points for Experiment 3

overlaying and covering the original set of points (blue points). This means that using this method can help readjust the measures of the current network to better match the measures of the previous network.

6.4.2 Proposed Method

The proposed method was implemented in Java using the JUNG library to create the social networks and calculate the measures. It was tested on a randomly generated network of 100 nodes using the small world method. The generated network used in the testing is unweighted and undirected, because this is the type of network with the least amount of information available. This means that if the method proves to be effective for unweighted and undirected networks, then it will also be effective for weighted and directed networks, as they can be reduced to unweighted and undirected networks. The original network was used as the old network input, and modified versions of the network were used as the new network input.

The network was modified in three different ways to simulate missing data or slight changes in the network. In the first trial, a random node was removed from the network to simulate the case of investigators failing to uncover an entity that is part of the network. In the second trial, two random nodes were removed from the network to simulate the case of investigators failing to uncover several entities that are part of the network. This also simulates the case of investigators failing to uncover an entity that is part of the network, and a slight change in the network with the absence of one other node. In the third trial, two random nodes were removed from the network, and two random links were removed in the network. The removal of the links in this case simulates a slight change in the network structure with the absence of two links.

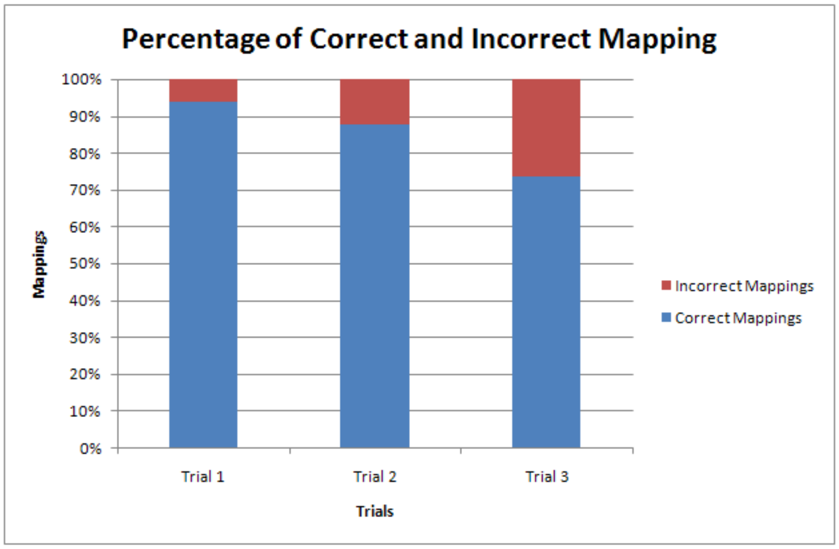


Figure 6.24: Chart of Results

The results of the first trial showed that 93% of the nodes had been mapped correctly (see Figure 6.24). Many of the nodes with incorrect mappings, had similarity measures similar to the correct nodes in the original social network. The first trial also successfully identified which node was the missing node (the node randomly removed). The results of the second trial showed that 87% of the nodes had been mapped correctly (see Figure 6.24). The drop in the proportion of correct mappings was expected, because the removal of more data from the network will make it more difficult to determine the original state of the network. Several nodes with incorrect mappings, had similarity measures that were noticeably different from the correct nodes in the original social network. This is because the removal of the two nodes had significantly altered the centrality of several of the nodes in the network. The second trial also successfully identified both nodes which were missing. The results of the third trial showed that 72% of the nodes had been mapped correctly (see Figure 6.24). The third trial was unsuccessful in identifying both missing nodes. The incorrect mapping of the missing node did have a lower similarity measure, and several of the other incorrect mappings had some of the lowest similarity measures. This indicates a lower confidence in their mapping, which also helps to indicate that there might be an incorrect mapping. The third trial had the worst results due the most amount of missing information.

6.5 Summary

This case study provides a solution to assist in detecting hidden nodes in criminal networks. This is accomplished by using two social networks, one that represents the current structure of the network, and another that represents the past structure of the network. The solution plots the nodes of each network on a Cartesian coordinate system using various network elements, and adjusts the plot points of one network to account for the missing data, while maintaining the overall structure of the social network. The last step locates the nodes of the other social network which are the closest and maps them together, where then the confidence measures of these mappings are computed. This information can help investigators identify missing nodes, thus preventing further criminal activity.

The solution presented in this case study can be used to determine if two graphs are isomorphic, and thus determine whether there are missing nodes or not. The experiment results show that the method can correctly map the majority of nodes in cases where the amount of data missing is minimal. However, as the amount of missing data grows, the proportion of the correctly mapped nodes will shrink. The method also relies on the past network data collected, which means that using unreliable past network data will also give unreliable results. In terms of criminal network analysis, this also relates to the problem of never having reliable past network data; because the acquisition of the network data will compromise the criminal network, thus introducing an incentive for the criminal network to change. This means the network can be unstable and the method can lose its effectiveness.

Chapter 7

PLACE2GIVE CASE STUDY: Applying Neural Networks to Score Surveys

7.1 Introduction

This real world case study demonstrates the effectiveness of a unique grouping technique in producing a solution for allowing the application of neural networks in the following situation, where there are some technical issues with applying neural networks to solve the problem just as they are. In this case study, the developed unique grouping technique is an initial step in the series of processes. The unique grouping technique plays an important role in the creation of the neural network, where it manipulates the composition of the input nodes in the neural network. This operation will adapt the neural network to function effectively for this particular real world problem. The unique grouping technique groups the input nodes in the neural network based on their associated questions and benchmarks, then determines which groups need special input nodes to be incorporated. This shows that unique grouping techniques can be applicable to real world problems, which means that unique grouping techniques are an important technique for both academic and industry research. The developed solution in this case study is an important technique for modifying neural networks to be effective in different situations. The unique grouping technique in this case study has been developed and implemented for Dexterity Consulting's online service named Place2Give.

7.1.1 Background

Automation is a practice that incorporates a system which independently assists in completing a process, and thereby reduces the amount of human intervention required for said process. This is an important practice for businesses due to its advantages of freeing up human resources, as well as the consistency and precision of the technologies implemented for such systems. Automation technology involves the usage of mechanical devices to reduce the physical labour required; now, with the advances in technology and machine learning techniques, automation is also capable of reducing examination and analysis work.

This real world case study will detail the implementation of a machine learning technique, neural networks, for Dexterity Consulting, and the modifications designed for neural networks to suit the needs of Dexterity Consulting. Dexterity Consulting is a company who has had over fifteen years of experience in the charitable sector, and whose mission is "To change the way that North America's Non-Profit sector operates by maximizing donor impact" [28]. Dexterity Consulting has created an online service that connects charities and donors called Place2Give (www.place2give.com). This website allows users to search a database containing thousands of charities with financial, social media, and other relevant information regarding the charity. This information is an amalgamation of data that is collected from various sources, which includes Charity Intelligence, The Donner Award, Canada Revenue Agency, and other publicly available information, such as the websites of the charities themselves. The goal of Place2Give is to help find the best combination of charities for donors to support, by matching the individual preferences and needs of the donor to the best corresponding charities. This will ensure that the donations from their clients are reaching the intended parties to maximize the impact of the donation.

The matching process consists of collecting information about charities and donors, by having charities and donors complete surveys; then the matching of charities and donors is achieved by calculating a risk assessment score for both charities and donors using information primarily collected from the surveys. Another source of information used to calculate the risk assessment score for charities come from the financials of the charity, which is provided by Canada Revenue Agency. The risk assessment score of the charities and donors are based on seven benchmarks: governance, funding, financial, delivery, volunteering, administration, and community perception. Based on this score, the charities and donors are classified into one of three categories: maverick, steady, and informed. The charities and donors in the same categories with the lowest differences in the risk assessment scores are then matched to one another.

The charity survey is static in that the questions do not regularly change, whereas the survey for donors incorporates dynamic questions to accommodate inquires regarding current events, such as recent disasters to better score and categorize the donor at that moment. Both surveys contain many questions which are optional for the charity or donor to answer, and the questions contribute a score of zero to ten for one or more benchmarks related to the content of the question.

The risk assessment score is calculated by averaging every average benchmark score; the average benchmark score is calculated by averaging all the benchmark scores for that benchmark. The risk assessment score uses the following equation:

$$RAScore = {}^{1}/_{7} \times avg(b_1) + {}^{1}/_{7} \times avg(b_2) + ... + {}^{1}/_{7} \times avg(b_7) \tag{7.1}$$

where $avg(b_x)$ is the average of the question scores for benchmark x, excluding skipped questions. The reason that the average function is used here is because many of the questions in the surveys are optional, and there was a desire to not punish the charity or donor for skipping questions. Therefore, using the average allows for the system to best score the charity or donor based on the current available information. By default, the risk assessment score is an average of the benchmark scores, which is why each benchmark score is multiplied by ${}^{1}/_{7}$, as there are seven benchmarks.

Equation (7.1) is the only formula used to find the risk assessment score of donors, but each average benchmark score of the donor can be used to help tweak the risk assessment score of a charity to better match the individual preferences of that particular donor. This is accomplished by making the risk assessment score for charities dependent on the donor's preferences, which can be based on the average benchmark scores of the donor that the charity is being compared to. This risk assessment score would change based on the individual preferences of each donor, by weighing the benchmark scores of the charity to the donor's corresponding benchmark scores. This would give more weight to the benchmark scores which are important to the donor, and less weight to the benchmarks for which the donor has little concern for. The risk assessment formula for the charities when being compared to a donor is presented in the following equation:

$$RAScore_{CHARITY} = w_1 \times avg(b_1) + w_2 \times avg(b_2) + ... + w_7 \times avg(b_7) \tag{7.2}$$

where w_x is the weight adjustment for benchmark x, which is determined based on the average benchmark scores of the donor. The weight adjustment is determined by the percentage of the total risk assessment score that the donor scored for benchmark x multiplied by seven, because there are seven benchmarks. The formula for this weight adjustment is shown in the following equation:

$$w_x = \frac{avg(b_x)}{RAScore \times 7} \tag{7.3}$$

where both the $avg(b_x)$ and the $RAScore$ are the scores of the donor. This will assign each charity a unique risk assessment score for each donor based on their benchmark preferences, which will allow for a more accurate categorization of the charity from the perception of the donors.

7.1.2 Problem and Solution

The scoring process for charities was originally completed by an expert manually reviewing the survey results, and then assigning the charity a risk assessment score based on their knowledge of the charitable sector. There was a need to automate this process due to the thousands of charities in the database, thus a system was needed to autonomously score charities similar to the expert's assessment by learning the score of each question and answer from the expert.

The scoring process for donors was originally automated with scores assigned to each answer based on the best approximation by the expert. There was also an option for the donors to manually readjust their assigned categories, in case they did not feel that the system had properly categorized their donation preferences; there was a need for a more accurate scoring system, which will continue to allow for the donors to correct their categorization, and learn from these readjustments to assign more accurate categorizations in future donor surveys.

The main problem in automating the matching process is the need for a system that is able to learn and assign scores to the answers from the survey. Neural networks were the chosen machine learning technique to help automate the process by learning from the previous risk assessment scores evaluated by the expert, and the categorization corrections recorded from the donors. Neural networks were chosen, because of its ability to find patterns by learning from various examples, and also because the neural network's output node's value equation (7.4) is similar to the risk assessment score equation (7.1). In using neural networks to tackle this challenge, two issues needed to be addressed in order for neural network's to succeed in learning to assign scores to the questions. One is that although the risk assessment score equation (7.1) and the neural network's output node's value equation (7.5) are similar, there remain slight differences between the equations that need a resolution. The other issue is that the system will also need to take into consideration the dynamic questions in the donor survey, and dynamic learning examples due to donors completing the survey at different times with different questions.

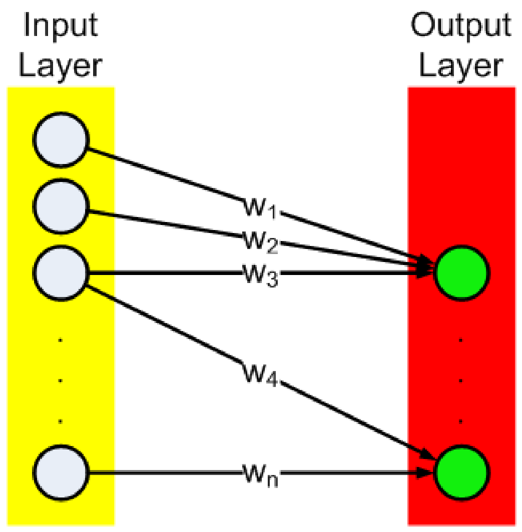


Figure 7.1: General Single-Layer Feed Forward Neural Network

7.2 Related Work

There are many different types of neural networks; a simple type of neural network was implemented for the system, which is called a single-layer feed forward neural network (see Figure 7.1). These neural networks consist of one set of input nodes known as the input layer, and one set of output nodes known as the output layer [17]. Each input node is linked to one or more output nodes with a numerical weight attached to each of these links. The input node is said to have fired if the input value of the node is greater than some threshold defined by the user, which will be zero in the implemented system. Values for the input nodes are provided by the user, and the values for the output nodes are calculated by summing all the products of the link weights, and their input node values, for each input node that has fired [17]. The output node value calculation for each output node would involve the following equation:

$$y_j = x_1 \times w_{1,j} + x_2 \times w_{2,j} + ... + x_n \times w_{n,j} \tag{7.4}$$

where y_j is the jth output node's value, x_i is the ith input node's value, and $w_{i,j}$ is the weight of the link from the ith input node to the jth output node. In the simplest case of the single-layer feed forward neural network, where there is only one output node, the output value formula for the output node becomes the following equation:

$$y = x_1 \times w_1 + x_2 \times w_2 + ... + x_n \times w_n \tag{7.5}$$

where y is the output value of the neural network, x_i is the ith input node's value, and w_i is the weight of the link from the ith input node to the output node.

The weights of the links in a single-layer feed forward neural network can be trained to find a pattern, by using a learning algorithm called the delta rule with a training data set consisting of input values and output values (see Algorithm 14). The delta rule is a simple learning algorithm that calculates the error between the output nodes' calculated values, and the correct output values based on the output data corresponding to the provided inputs in the training data set [7]. This error is used to adjust the weights of the links in the network, and all of the data will generate weight adjustments that gradually correct the weight of the links in the neural network in repeated iterations; this will teach the system to reproduce the same pattern in the training data set. The following equation shows the formula for calculating the new weight of the link:

$$w_{i,jNEW} = w_{i,jOLD} + \alpha \times (d_j - y_j) \times x_i \tag{7.6}$$

where $w_{i,jNEW}$ is the new weight of the link between the ith input node and the jth output node, $w_{i,jOLD}$ is the old weight between the two input and output nodes, α is a learning rate that is usually set to 0.1, and d_j is the output value for the jth output node from the data set.

Algorithm 14 Learning Algorithm

```
**INPUT:** Neural network(N) with input nodes($x_i$), links($w_{i,j}$), and output nodes($y_j$);
training set(T) of input samples($c_i$) and output samples($d_j$);
error threshold($\gamma$) and/or iteration limit($n$)

Initalize link weights($w_{i,j}$) to 0
for each sample in the training set(T) and
while error($d_j - y_j$) is less than the threshold($\gamma$) or
the iteration is less than the limit($n$) do
    Set the input node values($x_i$) to the input sample values($c_i$)
    Calculate each of the output node's value($y_j$) (see Equation (7.4))
    Calculate each links' new weight value($w_{i,jNEW}$) (see Equation (7.6))
    Set all link weights($w_{i,j}$) to their new value($w_{i,jNEW}$)
end for
```

7.3 Implementation

7.3.1 Overview

Two similar neural networks were implemented, one for the system to mimic the behavior of the expert in assigning a risk assessment score using the charity surveys. The other was for the system to learn from the donors corrections to more accurately assign risk assessment scores in the future, based on the donor surveys. The neural networks will do this by finding the best score to assign for each answer to the questions from the charity and donor surveys. Although the equation to calculate the value of the output node (7.5) closely resembles the risk assessment score equation (7.1), modifications to the single-layer feed forward neural network were designed to accommodate the slight differences in the risk assessment score, and output node value formulas in both neural networks. Furthermore, there were also modifications designed for the neural network and the learning algorithm implemented for computing the risk assessment score of donors, and finding the scores to assign the questions from the donor survey. This modification is to address the donor questions being dynamic, and the training data for this neural network being dynamic as well.

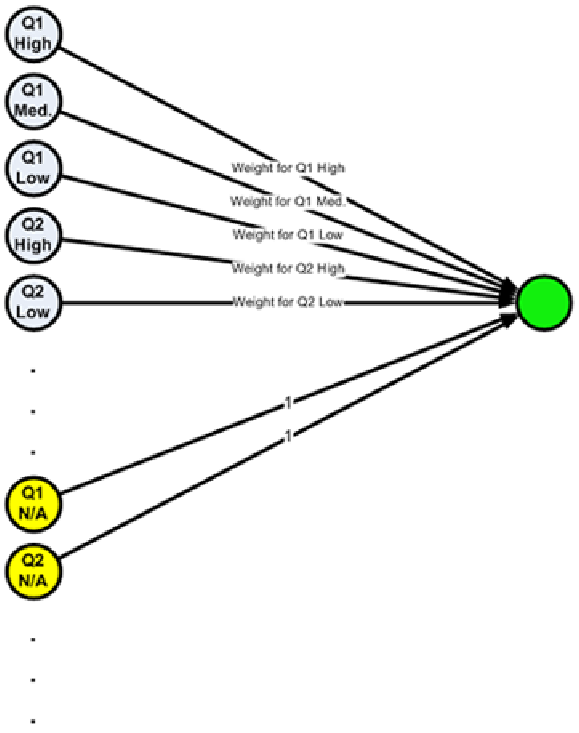


Figure 7.2: Implemented Neural Network

7.3.2 Padded Static Input Nodes Modification

For the charity component, the neural network implemented for learning the scores to the answers in the charity survey, contains a set of input nodes representing the answers for each question (see Figure 7.2 [gray nodes]). The input nodes' values are set to one if the question answered corresponds to the answer representation of that input node, otherwise the value is set to zero. For example, in the charity survey for question one, if the answer was determined to score a low value, then the input nodes "Q1 High" and "Q1 Med" would have their values set to zero, and the input node "Q1 Low" would have a value of one. This will fire the nodes based on how the survey is answered, and thus help the network determine what score to assign for a specific answer. The neural network also consists of one output node, which represents the risk assessment score of the charity based on their survey answers (see Figure 7.2 [green node]). The output from this node will be compared to the output specified by the expert in the training data to train the weights of the network, which represent the score for the answer associated with the link's node. The charity survey answers are used in this model as the inputs of the training set, and the corresponding risk assessment scores provided by the expert serve as the outputs of the training set (see Figure 7.3).

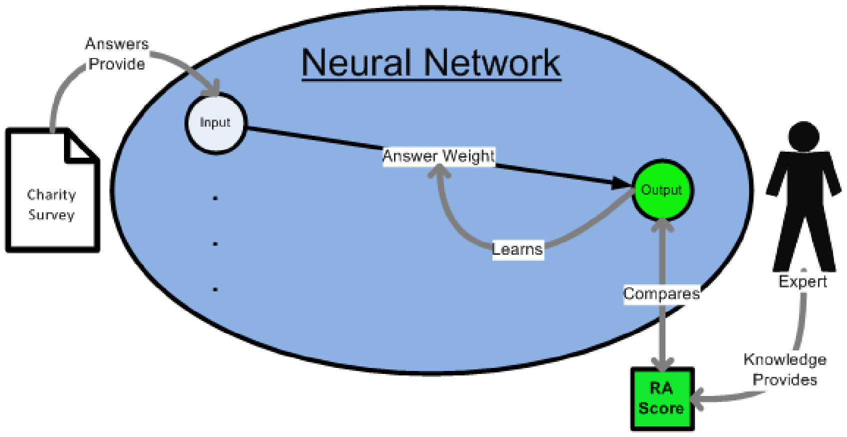


Figure 7.3: System (Charity Side)

The solution to applying neural networks in this situation with the difference between the risk assessment score equation and the neural network's output value equation is simple, in the case where every benchmark has the same number of questions associated with it. This is because the risk assessment score equation can be algebraically altered to have the same form as the neural network's output value equation (see Table 7.1). The equation in step 5 of Table 7.1 shows an altered form of the risk assessment score equation that matches the output node value equation. In this form, n represents the number of questions for each benchmark, and $q_{i,b}$ represents the score of the answer for the ith question related to benchmark b. Here the "$n \times 7 \times RAScore$" part represents the output value in the output node's value equation, and each $q_{i,b}$ represents a input node that has fired. Although this would have to rely on each benchmark having the same number of questions associated with it, and that each question is also required to be answered in the surveys.

1.	$RAScore$	$=$	${}^1/_7 \times avg(b_1) + {}^1/_7 \times avg(b_2) + ... + {}^1/_7 \times avg(b_7)$
2.	$7 \times RAScore$	$=$	$avg(b_1) + avg(b_2) + ... + avg(b_7)$
3.	$7 \times RAScore$	$=$	${}^1/_{n_1} \times (q_{1,1} + q_{2,1} + ... + q_{n_1,1}) + {}^1/_{n_2} \times (q_{1,2} + q_{2,2} + ... + q_{n_2,2}) + ... + {}^1/_{n_7} \times (q_{1,7} + q_{2,7} + ... + q_{n_7,7})$
4.	$7 \times RAScore$	$=$	${}^1/_n \times (q_{1,1} + q_{2,1} + ... + q_{n,1}) + {}^1/_n \times (q_{1,2} + q_{2,2} + ... + q_{n,2}) + ... + {}^1/_n \times (q_{1,7} + q_{2,7} + ... + q_{n,7})$
5.	$n \times 7 \times RAScore$	$=$	$q_{1,1}+q_{2,1}+...+q_{n,1}+q_{1,2}+q_{2,2}+...+q_{n,2}+...+q_{1,7}+q_{2,7}+...+q_{n,7}$

Table 7.1: *RAScore* Equation

The reason there is a need for a modification is because we cannot change the form of the risk assessment score equation to match the form of the output value equation, without adding in extra variables in the case where the benchmarks do not have the same number of associated questions. In this case, we cannot continue after step 3 in Table 7.1, where n_b represents the number of questions for benchmark b, because we cannot move all the n_b variables to the other side, and separate them from the $q_{i,b}$ variables that represent a fired input node value multiplied by the input node's link weight.

The solution to changing the risk assessment score equation's form to match the output value's equation, is to pad each benchmark to the maximum number of related questions in any of the benchmarks, with variables that represents the average of that benchmark. This will keep the equation the same, because adding the average score of the benchmark scores will not change the overall average score of the benchmark scores; this is because the new padded variable is the original average, thus it will not be pulling the average up or down. In doing so, all n_b become equal (n_{MAX}), because this essentially brings up the count for the number of questions in the benchmarks with lower number of associated questions, to the same number as the benchmark with the highest number of associated questions. Once this is done, the risk assessment equation can be rearranged to match the form of the output node's value formula (see Table 7.2). In this form, "$n_{MAX} \times 7 \times RAScore$" becomes the output of the neural network; and each $q_{i,b}$ becomes the input nodes which are fired, with the padded $avg(b_x)$ becoming special input nodes that would have a link weight of one.

A modification called padded static input nodes was designed for the neural networks

1.	$7 \times RAScore =$	${}^1/_{n_1} \times (q_{1,1} + q_{2,1} + ... + q_{n_1,1}) + {}^1/_{n_2} \times (q_{1,2} + q_{2,2} + ... + q_{n_2,2}) + ... + {}^1/_{n_7} \times (q_{1,7} + q_{2,7} + ... + q_{n_7,7})$
2.	$7 \times RAScore =$	${}^1/_{n_{MAX}} \times (q_{1,1} + q_{2,1} + ... + q_{n_1,1} + (n_{MAX} - n_1) \times avg(b_1)) + {}^1/_{n_{MAX}} \times (q_{1,2} + q_{2,2} + ... + q_{n_2,2} + (n_{MAX} - n_2) \times avg(b_2)) + ... + {}^1/_{n_{MAX}} \times (q_{1,7} + q_{2,7} + ... + q_{n_7,7} + (n_{MAX} - n_7) \times avg(b_7))$
3.	$n_{MAX} \times 7 \times RAScore =$	$q_{1,1} + q_{2,1} + ... + q_{n_1,1} + (n_{MAX} - n_1) \times avg(b_1) + q_{1,2} + q_{2,2} + ... + q_{n_2,2} + (n_{MAX} - n_2) \times avg(b_2) + ... + q_{1,7} + q_{2,7} + ... + q_{n_7,7} + (n_{MAX} - n_7) \times avg(b_7)$
4.	$n_{MAX} \times 7 \times RAScore =$	$q_{1,1} + q_{2,1} + ... + q_{n_1,1} + avg(b_1) + ... + q_{1,2} + q_{2,2} + ... + q_{n_2,2} + avg(b_2) + ... + ... + q_{1,7} + q_{2,7} + ... + q_{n_7,7} + avg(b_7) + ...$

Table 7.2: Adjusted *RAScore* Equation (© 2012 Dexterity Ventures Inc.)

implemented, which function similar to the basic input nodes; their purpose is to address the differences between the risk assessment score and output node's value equations (see Figure 7.2 [yellow nodes]). Padded static input nodes are the special input nodes that have a input value of either $avg(b_x)$ or zero, and a link weight of one. These nodes are padded into the network, and will not have their link weights adjusted in the learning algorithm to address the issue of having a different number of questions for each benchmark, and a question being skipped on the charity survey. The latter issue is quite similar to the former issue, because if a question is skipped, then it could create an imbalance in the number of questions associated with each benchmark.

Each input node in the neural networks represents a question associated with a benchmark, thus each input node is associated with a benchmark, and there needs to be an equal amount of input node nodes associated with each benchmark. Therefore, these padded static nodes are added to the neural network to balance out the number of input nodes corresponding to each benchmark, so that each benchmark has the same number of input nodes, and these padded static nodes are always assigned the value of the average for the benchmark scores that correspond to the node, meaning they are always fired. These padded static nodes are also added to the neural network for each question in the survey, and are assigned a zero value if the question they represent is not skipped, but if the question is skipped then the values of these nodes become the average of the benchmark scores that correspond to the question. In essence, for each question on the survey, a group of network nodes is created along with a padded static input node, where only the node that represents the answer chosen is fired, or in the case of a skipped question the padded static input node is fired. Every node created is then essentially grouped together based on the benchmark of the questions the nodes are associated with; subsequently, the groups are all filled with padded static input nodes that are always fired to bring the number of nodes in each group to the same capacity.

7.3.3 Dynamic Neural Network Modification

For the donor component, the neural network implemented for learning the scores to the answers in the donor survey is similar to the neural network for the charity component, but this neural network needs to take into consideration the recurring changes in the questions of the donor survey. It is important to note that as the questions for the donor survey are dynamic, the neural network also needs to be dynamic, as the input nodes that represent the answers to the questions may no longer be required in the model of the neural network at later times. This neural network contains a set of input nodes representing the answers for each question, but the input nodes are generated each time the survey is taken, because the questions in the donor survey may change frequently; hence, each time the survey is taken, an input node needs to be generated for each question answered, and thus all input nodes in this network are always fired. For this neural network, there is no need to create input nodes that will not be fired, because the neural network needs to be reconstructed each time to represent the current set of questions in the donor survey, which is what makes this neural network dynamic.

The dynamic neural network implemented also contains the padded static input nodes modification to address the differences in the risk assessment score and output node's value equations, which are also dynamically generated. These nodes are used to pad the network to resolve the issue of having a different number of questions associated to each benchmark, and a question being skipped the donor survey. These padded static input nodes are similar to the input nodes of this dynamic neural network as they are also always fired, because they are only generated if they are needed.

The dynamic neural network also consists of one output node, which represents the risk assessment score of the donor based on their survey answers. The output from this node will be compared to the output specified by the donor's readjustment of their category in the training data, to learn the weights of the network, which represents the score for the answer associated with the link's node. The donor survey answers are used in this model as the inputs of the training set, and the corresponding risk assessment score readjustments provided by the donor in the case where they correct their categorization, serve as the outputs of the training set (see Figure 7.4). Similar to the dynamic neural network, the associated training set is also generated each time the donor survey is completed; therefore the training set will always only contain one training sample. The learning phase can only use one training sample during the learning phase. The reason for this is because of the changing input nodes in the neural network, which means that the input data in training samples will not always match the neural network, since the input nodes representing the input data may have disappeared. For this reason, the dynamic neural network needs to be reconstructed and trained each time a donor completes and readjusts their categorization in the donor survey.

There is a problem with using the original learning algorithm for training the weights of the dynamic neural network, because as input nodes are generated on the fly, there is only one training sample to use for the learning phase. This causes a problem because the learning algorithm would only ever consider the current training sample, and disregard any changes made to similar questions from the past learning phases. The solution to this is to keep track of the link weights for each node, and the average change of the link weight between the ith input node and the output node (Δw_i). One of the changes to the learning

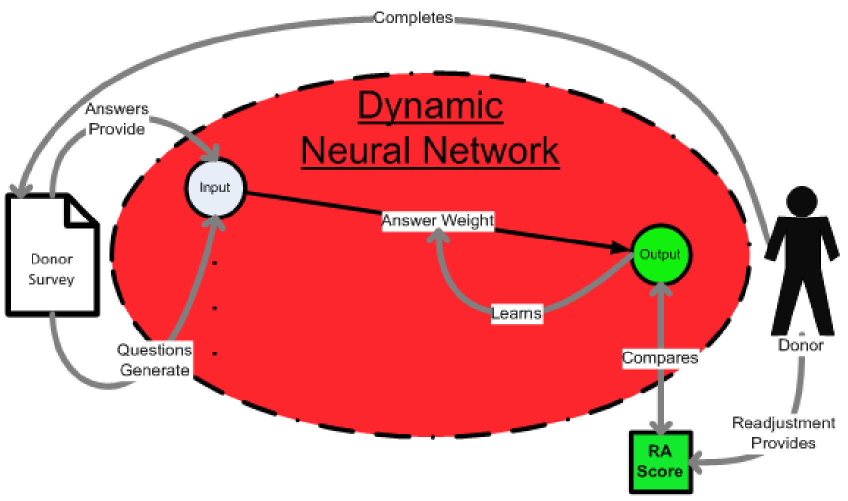


Figure 7.4: System (Donor Side)

algorithm implemented for training on only one sample, is to start off by reinitializing the weights of each input node to their previous value from the last iteration of the learning algorithm (see Algorithm 15 [first step]). This will start the neural network off at the last point in the previous learning phase, but it is important to note that there must be some method of identifying each input node uniquely. The next change to the learning algorithm is to add another step of adjustment after the first adjustment of weights, based on the average change ($\Delta w_{i,j}$) of that weight in the previous learning phases (see Algorithm 15 [second last step]). This will simulate running the learning algorithm on the previous training samples, while giving more influence to the current training sample. This step would be ignored for the weights of links that have just appeared in the network, or weights of links that have disappeared altogether. The final step in the new learning algorithm is to record the new average change ($\Delta w_{i,j}$) for each weight after the weight adjustment iterations have completed, so it can be used once again in the next generation of the dynamic neural network (see Algorithm 15 [last step]). The resulting system learns from the donors to assign a risk assessment score based on the current version of the donor survey.

Algorithm 15 Dynamic Learning Algorithm (© 2012 Dexterity Ventures Inc.)

```
INPUT: Neural network(N) with input nodes($x_i$), links($w_{i,j}$), and output nodes($y_j$);
training sample($T_s$) with input samples($c_i$) and output samples($d_j$);
error threshold($\gamma$) and/or iteration limit($n$)

Initialize link weights($w_{i,j}$) to previous values or
0 for new link weights
for the training sample($T_s$) and
while error($d_j - y_j$) is less than the threshold($\gamma$) or
the iteration is less than the limit($n$) do
  Set the input node values($x_i$) to the input sample values($c_i$)
  Calculate each of the output node's value($y_j$) (see Equation (7.4))
  Calculate each links' new weight value($w_{i,jNEW}$) (see Equation (7.6))
  Set all link weights($w_{i,j}$) to their new value($w_{i,jNEW}$)
  Adjust all link weights($w_{i,j}$) by their average change($\Delta w_{i,j}$) if available
end for
Update and record all link weights' average change($\Delta w_{i,j}$)
```

7.4 Summary

The implementations of the two neural networks with the designed modifications to address the specific issues in this situation, will create a system that is able to autonomously calculate the risk assessment scores of charities and donors based on information primarily from the surveys. The neural networks learn from the expert and donors to model the same patterns in evaluating the two surveys. This will greatly reduce the analytical work of the expert in examining the charity surveys, and evaluating risk assessment scores for the many charities in the database. This will also increase the accuracy of the system's categorization of donors, and reduce the frequency of donors readjusting their categorizations after they have taken the survey. The automation for the remainder of the matching process, where the risk assessment scores of the charities and donors are compared, was completed with simple query operations to the database.

The required modifications to the neural networks, which were essential for their implementation in this situation, show that while machine learning techniques provide an effective option for computers to automate analytical work, the machine learning techniques cannot always be utilized in the same manner. This is because machine learning techniques cannot accomplish every objective with great results, which is why they are not suited for every situation. This is also evidenced by the large variety of different machine learning techniques to address different problems [11]. There are even many different types of neural networks in research to consider for different issues, such as spiking neural networks which address time dependent issues [23]. For businesses looking to machine learning techniques as a solution to automate analysis work, they will also need to consider altering the machine learning technique they have chosen to best fit their particular situation.

Chapter 8

CONCLUSION AND FUTURE RESEARCH DIRECTIONS

8.1 Conclusions

In conclusion, this thesis establishes the effectiveness and significance of unique grouping techniques through the development and application of unique grouping techniques in four distinctive cases. A general framework for developing and implementing unique grouping techniques is proposed. The proposed framework is made up of two phases, where analysis is initially prepared to determine the structure of the network and related research. The second phase guides the design of the unique grouping technique through searching for patterns and exploring different groupings of nodes, and finishes with examining and revising the developed technique. Four case studies in the thesis illustrates that unique grouping techniques are applicable to various types of networks, and fields of study.

The first case study proves that unique grouping techniques are applicable to abstract networks, by grouping nodes in a network of association rules to find the customer behaviors of separate market segments. This case study shows that unique grouping techniques do not need to be completely original, but can be based on grouping techniques that currently exist; part of the unique grouping technique developed in this case study is very similar to graph coloring techniques. This case study establishes an effective method in reducing the difficulty for businesses to review the association rules of different customer segments, and track the changes of market segments based on their buying behaviors. This has great importance for businesses, because it can help businesses focus on their target market by reviewing the rule set representing the target market.

The second case study proves that unique grouping techniques are applicable to social networks, by grouping nodes in a criminal network to find the hidden links. This case study shows that unique grouping techniques can be reused for other vastly dissimilar problems, despite being uniquely developed for a specific situation; the unique grouping technique developed in this case study uses the graph coloring like technique developed in the first case study. The case study demonstrates a method to locate possible hidden links, thus providing valuable information for law enforcement agencies, and reducing the number of hidden links in their network. Accordingly, this is will improve the results for the analysis of such networks by addressing one of the major issues in analyzing criminal networks.

The third case study also proves that unique grouping techniques are applicable to social networks, by grouping nodes in a criminal network to find the hidden nodes. This case study shows that unique grouping techniques do not need to be complex in nature, but can be kept simple and still provide a effective solution; the unique grouping technique developed in this case study groups nodes, with similar measured properties together with respect to being above or below the average of each property. This case study provides a solution to assist in detecting hidden nodes for criminal networks. This information can help investigators identify missing nodes, thus preventing further criminal activity.

The last case study proves that unique grouping techniques are applicable to neural networks, by grouping nodes in a neural network to help score surveys in a charity and donor matching service. This case study shows that unique grouping techniques are not just applicable in theory, but can be used for real world problems; the unique grouping technique developed in this case study is used to modify neural networks, and adapt them to the particular conditions required by the surveys for the charity and donor matching service. The implementations of the two neural networks in this case study will create a system that is able to autonomously calculate the risk assessment scores of charities, and donors based on information primarily from the surveys. The modifications to the neural networks were essential for their implementation in this situation.

This thesis shows that unique grouping techniques should be a serious consideration alongside general grouping techniques for research work dealing with networks.

8.2 Future Research Directions

This thesis presents three theoretical unique grouping techniques with central roles in solving three vastly different research problems; each of these three unique grouping technique solutions can be further expanded upon. The drawback with the method presented in the first case study is in finding the thresholds values to use. Future work can be conducted on researching heuristics for finding the best thresholds to input at each phase, based on the database and rule information available. Future research for the solution in the second case study can delve into pattern mining methods to identify what the relationship types of the determined hidden links are; this will provide even more valuable information to investigators using social network analysis. Additional research can be conducted on the unique grouping technique in the third case study, to determine the effectiveness for comparing the similarity between different types of social networks; for example, the method can be applied to different successful information technology companies, to determine if there are similar key aspects in their social network structure which lead to the success of the business.

This thesis presents a general framework for implementing unique grouping techniques; however, there are opportunities to build on this framework, and develop specialized unique grouping technique frameworks for different types of networks. Different guiding principles may perhaps be better suited for the progression on the development of unique grouping techniques dealing with different types of networks. In addition, the system of the framework can be further focused for diverse areas of study, thus altogether creating a comprehensive and in depth framework for the development of unique grouping techniques.

Bibliography

[1] M. Agarwal, H. Agrawal, N. Jain, and M. Kumar. Face recognition using principle component analysis, eigenface and neural network. In *Signal Acquisition and Processing, 2010. ICSAP '10. International Conference on*, pages 310–314, Feburary 2010.

[2] A. An, S. Khan, and X. Huang. Objective and subjective algorithms for grouping association rules. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 477–, Washington, DC, USA, 2003. IEEE Computer Society.

[3] C. Apte, B. Liu, E. P. D. Pednault, and P. Smyth. Business applications of data mining. *Commun. ACM*, 45:49–53, August 2002.

[4] L. Backstrom and J. Leskovec. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 635–644, New York, NY, USA, 2011. ACM.

[5] B. Baesens, S. Viaene, and J. Vanthienen. Post-processing of association rules. Technical report, 2000.

[6] W. Baker and R. Faulkner. The social organization of conspiracy: Illegal networks in the heavy electrical equipment industry. *American Sociological Review*, 58(6):837–860, 1993.

[7] I. A. Basheer and M. N. Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1):3–31, 2000.

[8] J. Baumes, M. Goldberg, M. Magdon-Ismail, and W. A. Wallace. Discovering hidden groups in communication networks. In H. Chen, R. Moore, D. D. Zeng, and J. Leavitt,

editors, *Intelligence and Security Informatics*, volume 3073 of *Lecture Notes in Computer Science*, pages 378–389. Springer Berlin / Heidelberg, 2004.

[9] A. Bhan, D. J. Galas, and T. G. Dewey. A duplication growth model of gene expression networks. *Bioinformatics*, 18(11):1486–1493, 2002.

[10] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[11] I. Bose and R. K. Mahapatra. Business data mining - a machine learning perspective. *Information & Management*, 39(3):211–225, 2001.

[12] D. Brélaz. New methods to color the vertices of a graph. *Commun. ACM*, 22(4):251–256, Apr. 1979.

[13] S.-H. Chang, F.-H. Cheng, W.-H. Hsu, and G.-Z. Wu. Fast algorithm for point pattern matching: Invariant to translations, rotations and scale changes. *Pattern Recognition*, 30(2):311 – 320, 1997.

[14] R. Christley, G. Pinchbeck, R. Bowers, D. Clancy, N. French, R. Bennett, and J. Turner. Infection in social networks: using network analysis to identify high-risk individuals. *American journal of epidemiology*, 162(10):1024–1031, 2005.

[15] A. Clauset, C. Moore, and M. E. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[16] F. D. Breaking al qaeda cells: A mathematical analysis of counterterrorism operations. *Studies in Conflict Terrorism*, 26(6):399–411, 2003.

[17] J. Dalton and A. Deshmane. Artificial neural networks. *Potentials, IEEE*, 10(2):33–36, April 1991.

[18] K. Dawoud, R. Alhajj, and J. Rokne. A global measure for estimating the degree of organization of terrorist networks. In *International Conference on Advances in Social Networks Analysis and Mining*, pages 421–427, August 9-11 2010.

[19] J. E. Dayhoff and J. M. DeLeo. Artificial neural networks: Opening the black box. *Cancer*, 91(Supplement 8):1615–1635, 2001.

[20] G. Dede and M. H. Sazl. Speech recognition with artificial neural networks. *Digital Signal Processing*, 20(3):763–768, 2010.

[21] M. Domingues and S. Rezende. Post-processing of association rules using taxonomies. In *Artificial intelligence, 2005. epia 2005. portuguese conference on*, pages 192 –197, dec. 2005.

[22] C. Fyfe. Artificial neural networks. In *Do Smart Adaptive Systems Exist?*, volume 173 of *Studies in Fuzziness and Soft Computing*, pages 57–79. Springer Berlin / Heidelberg, 2005.

[23] S. Ghosh-Dastidar and H. Adeli. Spiking neural netwoks. *International Journal of Neural Systems*, 19(4):295–308, 2009.

[24] C. Goodall. Procrustes Methods in the Statistical Analysis of Shape. *Journal of the Royal Statistical Society. Series B (Methodological)*, 53(2):285–339, 1991.

[25] G. Gupta, A. Strehl, and J. Ghosh. Distance based clustering of association rules. In *In Intelligent Engineering Systems Through Artificial Neural Networks (Proceedings of ANNIE 1999*, pages 759–764. ASME Press, 1999.

[26] M. Hasan and M. Zaki. A survey of link prediction in social networks. In *Social Network Data Analytics*, pages 243–275. Springer US, 2011.

[27] J. J. Hopfield. Artificial neural networks. *Circuits and Devices Magazine, IEEE*, 4(5):3–10, September 1988.

[28] D. V. Inc. Who is dexterity consulting?, 2008.

[29] M. G. I. Inc. Supermarket facts industry overview 2008, 2010.

[30] B. J. Jain and K. Obermayer. Extending bron kerbosch for solving the maximum weight clique problem. *CoRR*, abs/1101.1266, 2011.

[31] J. T. Jost and R. Andrews. *System Justification Theory.* Blackwell Publishing Ltd, 2011.

[32] M. Kim and J. Leskovec. The Network Completion Problem: Inferring Missing Nodes and Edges in Networks. In *SDM*, pages 47–58. SIAM / Omnipress, 2011.

[33] H. Koga, T. Ishibashi, and T. Watanabe. Fast agglomerative hierarchical clustering algorithm using locality-sensitive hashing. *Knowledge and Information Systems*, 12:25–53, 2007.

[34] N. Koochakzadeh, F. Keshavarz, A. Sarraf, A. Rahmani, K. Kianmehr, M. Rifaie, R. Alhajj, and J. G. Rokne. Stock investment decision making: A social network approach. In *ISMIS Industrial Session*, pages 47–57, 2011.

[35] G. Kossinets. Effects of missing data in social networks. *Social Networks*, 28:247–268, 2003.

[36] V. Krebs. Mapping networks of terrorist cells. *Connections*, 24:43–52, 2002.

[37] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[38] L. L and T. Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150 – 1170, 2011.

[39] V. Latora and M. Marchiori. How the science of complex networks can help developing strategies against terrorism. *Chaos, Solitons & Fractals*, 20(1):69–75, 2004.

[40] B. Lent, A. Swami, and J. Widom. Clustering association rules. In *Data Engineering, 1997. Proceedings. 13th International Conference on*, pages 220 –231, apr 1997.

[41] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 641–650, New York, NY, USA, 2010. ACM.

[42] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[43] H. Liu, J. Sun, and H. Zhang. Post-processing of associative classification rules using closed sets. *Expert Syst. Appl.*, 36:6659–6667, April 2009.

[44] A. Lorusso, D. W. Eggert, and R. B. Fisher. A comparison of four algorithms for estimating 3-d rigid transformations. In *Proceedings of the 1995 British conference on Machine vision (Vol. 1)*, BMVC '95, pages 237–246, Surrey, UK, UK, 1995. BMVA Press.

[45] S. M., , and W. J. Discovering hierarchical structure in terrorist networks. In *Proceedings of the International Conference on Emerging Technologies*, pages 238–244, 2006.

[46] M. Magdon-Ismail, M. Goldberg, W. Wallace, and D. Siebecker. Locating hidden groups in communication networks using hidden markov models. In H. Chen, R. Miranda, D. Zeng, C. Demchak, J. Schroeder, and T. Madhusudan, editors, *Intelligence and Security Informatics*, volume 2665 of *Lecture Notes in Computer Science*, page 958. Springer Berlin / Heidelberg, 2010.

[47] Y. Masatlioglu and E. A. Ok. Rational choice with status quo bias. *Journal of Economic Theory*, 121(1):1 – 29, 2005.

[48] N. Memon, U. Wiil, and A. Qureshi. Design and development of an early warning system to prevent terrorist attacks. In *Proceedings of the International Conference on Artificial Intelligence and Neural Networks*, pages 222–226, 2009.

[49] T. Murata and S. Moriyasu. Link prediction of social networks based on weighted proximity measures. In *Web Intelligence, IEEE/WIC/ACM International Conference on*, pages 85 –88, nov. 2007.

[50] K. P. The network paradigm applied to criminal organizations. *Connections*, 24(3):53–65, 2001.

[51] J. Qin, J. Xu, D. Hu, M. Sageman, and H. Chen. Analyzing terrorist networks: A case study of the global salafi jihad network. pages 287–304, 2005.

[52] T. Randall, P. Cowling, R. Baker, and P. Jiang. Using neural networks for strategy selection in real-time strategy games. In *AISB Symposium on AI & Games*, 2009.

[53] C. J. Rhodes and P. Jones. Inferring missing links in partially observed social networks. *Journal of the Operational Research Society*, 60(10):1373–1383, 2009.

[54] F. J. Rohlf and D. Slice. Extensions of the procrustes method for the optimal superimposition of landmarks. *Systematic Biology*, 39(1):40–59, 1990.

[55] S. S. Exploring complex networks. *Nature*, 6825(410):268–276, 2002.

[56] H. H. Song, T. W. Cho, V. Dave, Y. Zhang, and L. Qiu. Scalable proximity estimation and link prediction in online social networks. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 322–335. ACM, 2009.

[57] M. Sparrow. The application of network analysis to criminal intelligence: An assessment of the prospects. *Social Networks*, 13(3):251–274, 1991.

[58] S. Y. Sung, Z. Li, C. L. Tan, and P. A. Ng. Forecasting association rules using existing data sets. *IEEE Transactions on Knowledge and Data Engineering*, 15:1448–1459, 2003.

[59] H. Toivonen, M. Klemettinen, P. Ronkainen, K. Htnen, and H. Mannila. Pruning and grouping discovered association rules, 1995.

[60] M. Tsvetovat and K. Carley. Structural knowledge and success of anti-terrorist activity: The downside of structural equivalence. *Journal of Social Structures*, 6(2), 2005.

[61] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:376–380, 1991.

[62] D. Won, B. M. Song, and D. McLeod. An approach to clustering marketing data. In *2 nd International Advanced Database Conference (IADC*, 2006.

[63] J. Xu and H. Chen. Crimenet explorer: A framework for criminal network knowledge discovery. *CM Transactions on Information Systems*, 23(2):201–226, 2005.

[64] R. Xu and I. Wunsch, D. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, May 2005.