UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Analysing network motifs in a complex network of freight movements

by

Sumarie Meintjes
28408129

Submitted in partial fulfillment of the requirements
for the degree

Masters of
Industrial Engineering

in the

Faculty of Engineering, Built Environment

and Information Technology

University of Pretoria

October 2015

Acknowledgements

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Executive summary

Motifs are over-represented subgraphs in a complex network, and represent the building blocks of the network. There is a lack of studies that apply complex network theory in a supply chain context. In this dissertation 3-node motifs were identified and analysed in a complex network representing direct freight trips between firms in the Nelson Mandela Bay Metropolitan, South Africa. The G-Tries and ISMAGS algorithms were tested on small complex networks, and were compared according to quantitative and qualitative properties. It was found that ISMAGS is the most suitable for this dissertation. Freight activities were identified from raw GPS traces of freight vehicles, and the activities were clustered into firms using a density-based clustering algorithm. Multi-objective optimisation indicated that the clustering parameter configuration $\gamma = (20, 20)$ can be used to increase the visual accuracy of the firms, while maximising the completeness of the complex network. The freight complex network was built by identifying direct trips between firms. Using ISMAGS, it was found that three firms with two ($X0X$) or three (XXX) reciprocal freight trips between them are statistically overrepresented in the network. A brewery, shopping centres, distribution centres, and truck stops frequently appeared in the motifs. Motifs that contain the brewery and one of the truck stops were identified as the most central motifs based on the number of direct freight trips that occur in the motifs. Some freight trips in XXX motifs occur frequently over long distances, increasing total transport costs of the firms. Supply chain improvements can be applied to these identified firms. It was also found that there is a relationship between ranking firms according to the number of motifs they appear in and their degree centrality scores. This relationship can be studied more rigorously in future work. Another avenue for future research is to study the supply chain structures of firms in motifs, as well as the commodity flows between firms in motifs.

Contents

List of Figures

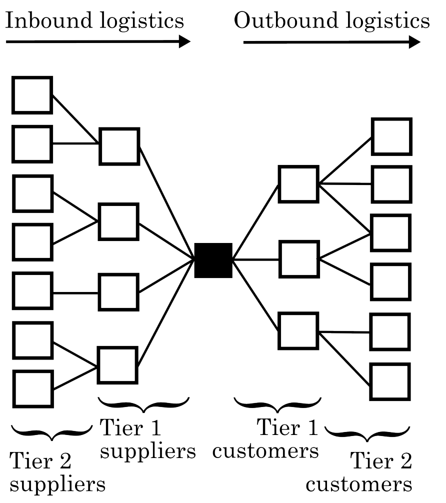
List of Tables

List of Algorithms

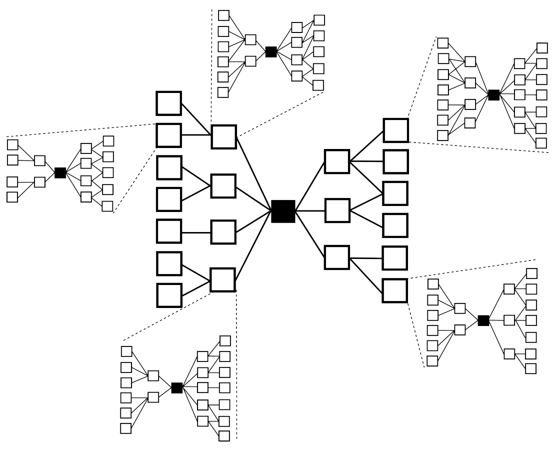
Chapter 1

Introduction

Traditionally, supply chains are studied as linear structures, with one firm as the focal point, and its suppliers and customers organised in tier levels above and below the focal firm. An example can be seen in Figure 1.1(a), with firms represented by squares, and supply relationships as lines between firms. The focal firm is at the centre of the supply chain, and is shown as a black square.



(a) A linear supply chain with a single focal company.



(b) Each firm is focal in its own supply chain.

Figure 1.1: A supply chain has a simple linear structure. When multiple supply chains are considered together, they form a web, or network, of supply chains.

Min and Zhou (2002) explain that the primary focus of traditional supply chain analysis is on the movement of goods from the suppliers to the focal firm (inbound logistics) and the physical distribution of goods from the focal firm to the customers (outbound logistics).

A supply chain perspective is limited to the biased view the focal firm has on its suppliers and customers. In reality, each supplier and customer is the focal firm in its own supply chain, as shown in Figure 1.1(b). If all the supply chains of all the firms are considered together, it is not a simple linear structure, but a more complex one. This structure is referred to as a web by Battini et al. (2007), while Galaskiewicz (2011) and Bellamy and Basole (2013) refer to it as a network of firm relationships. This network view of supply chains is not biased to one firm, and provides an objective view of how firms are connected by relationships.

The network view of supply chains is not a new concept. Choi et al. (2001) were the first to consider a supply network not as a chain, but as a complex adaptive system. In a subsequent paper, Choi and Hong (2002) map the supply networks of Honda Accord, Acura, and DaimlerChrysler Grand Cherokee. They study the structure of the supply network structure in terms of formalisation, centralisation, and complexity, and find that these dimensions continually affect each other. Battini et al. (2007) compute the complexity of a supply network by applying network analysis paradigms and ecological indicators to it. More recently, Mari et al. (2015) develop resilience metrics for supply networks based on complex network theory. Authors such as Choi and Hong (2002) and Battini et al. (2007) refer to these networks as supply (chain) networks, while others, such as Joubert and Axhausen (2013) and Bellamy and Basole (2013), call it a complex network. This dissertation will use the phrase *complex network*.

The relationships in a complex network do not form haphazardly. Firms make conscious decisions about their suppliers, and will likely make the same decisions over and over again. This reinforces the relationships between firms, and over time, these decisions will manifest as patterns in the complex network. In this study the these *patterns* are identified, as well as the *key players* in these patterns.

1.1 Complex networks

To study these complex network patterns, a notation is needed that can accurately represent the firms and the relationships between them. Graph theory provides a vocabulary that can be used to describe relationships between actors, and its mathematical operations are suitable to study and quantify these relations. Using the vocabulary and mathematical operations, theorems about graphs can be proved, and therefore it can also be proved for the relations upon which the graph is based. Therefore graph theory provides a useful model of a complex network that contains a set of actors with connections between them, as explained by Wasserman and Faust (1994).

Boccaletti et al. (2006) explain that graph notation can be used to represent a directed and weighted network as a directed and weighted graph, $G^W = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{W})$, consisting of three sets, namely the set of nodes (or vertices), $\boldsymbol{V} = \{v_1, v_2, \ldots, v_N\}$, the set of edges (or connections, dyads or links) between the nodes, $\boldsymbol{E} = \{e_1, e_2, \ldots, e_K\}$, and the set of weights associated with each edge, $\boldsymbol{W} = \{w_1, w_2, \ldots, w_K\}$. There are N elements in set $\boldsymbol{V}$ and K elements in sets $\boldsymbol{E}$ and $\boldsymbol{W}$. Ribeiro and Silva (2010) state that each edge has a pair of nodes, (u, v), associated with it, where u is the origin node and v the destination node.

Network analysis is extremely versatile, and Borgatti and Li (2009) explain that the nodes and edges of a complex network can be defined as any set of actors and relations between them. All edges can be grouped into four categories, and Borgatti and Li (2009) explain what they are. *Similarities* between actors can be indicated using edges, such as when two actors belong to the same organisation. Edges can also represent *relations* between actors, such as friends or colleagues. *Interactions* between actors, such as meetings or negotiations, can also be represented using edges. The last edge category is that of *flows*, and an example is the delivery of goods from one actor to another, or the exchange of information between actors. Flows are the most difficult to measure due to a lack of data and the difficulty of capturing the data. Flows are therefore assumed from interactions or social relationships. The four edge categories (similarities, relations, interactions, and flows) can be further grouped into continuous ties (similarities and relations) and discrete ties (interactions and flows). This categorisation is based on the duration of the connection between actors: continuous ties exist indefinitely, while discrete ties are based on discrete events which can be counted up over time.

Borgatti and Li (2009) explain that multiplexity occurs when any, or all, types of ties exist simultaneously for any given set of actors. Using the concept of multiplexity, separate networks can be built using the same actors, each with a different type of edge between them. For example, the edges of one network can represent the negotiation of prices (interactions), and the edges of another network can represent the payment of money by one firm to another (flows). Studying these two networks simultaneously may provide more insight than when only studying one of these in isolation.

Edge direction indicates the direction of the relation between two actors. Ribeiro and Silva (2010) explain that edge existence and direction can be captured using an adjacency matrix, A, with its size equal to $|\boldsymbol{V}|$ (the number of nodes). A '0' in position a_{ij} indicates that there does not exist an edge between nodes i and j, while a '1' indicates that an edge exists between these two nodes. Undirected networks have symmetrical adjacency matrices, since a_{ij} and a_{ji} have the same value, but a directed network could have different values in these positions, since Borgatti and Li (2009) explain that a directed edge between i and j does not indicate the existence of a directed edge in the opposite direction.

Edge weight indicates the frequency, or the strength, of the connection between the two nodes. Boccaletti et al. (2006) note that it is especially useful when the network consists of connections that vary greatly in capacity and intensity, for example, when there are long-term delivery agreements and once-off deliveries in the same network.

In this dissertation, nodes are defined as the firms where freight vehicles stop to perform activities. Direct freight movements between two firms are represented as directed and weighted edges between nodes.

1.1.1 Studying complex network structure

Studying complex networks in its entirety can be daunting. Rather, a network is often reduced to its sub-components, also referred to as subgraphs. These structures are of particular interest in this dissertation. Milo et al. (2002) find that by identifying statistically significant subgraphs in a complex network, it can provide insight into the building blocks of that network.

In the seminal work by Milo et al. (2002), a *motif* is defined as a recurring, significant pattern of interconnection. Finding motifs in a network has developed as a science in its own right, and Tran et al. (2015) define the *motif discovery problem* as the searching for induced isomorphic subgraphs that occur significantly more in a target network than in an ensemble of random networks.

Again, graph notation can be used to concisely describe subgraphs. A *subgraph instance*, G_k, is a graph of size k found in the larger graph G. The subgraph instance nodes are a subset of G's nodes, therefore $\boldsymbol{V}(G_k) \subseteq \boldsymbol{V}$, and the subgraph instance edges are a subset of G's edges, $\boldsymbol{E}(G_k) \subseteq \boldsymbol{E}$. A subgraph is *induced* for any pair of nodes $(u, v) \in \boldsymbol{E}(G_k)$ if, and only if, $(u, v) \in \boldsymbol{E}$ (Ribeiro and Silva, 2010). Given two graphs, $G(\boldsymbol{V}, \boldsymbol{E})$ and $H(\boldsymbol{V'}, \boldsymbol{E'})$, G and H are *isomorphic* if there exists a bijective function, f, between $\boldsymbol{V}$ and $\boldsymbol{V'}$ such that for each edge $(u, v) \in \boldsymbol{E}$ there exists an edge $(f(u), f(v)) \in \boldsymbol{E'}$ (Tran et al., 2015). A *bijective function* provides a one-to-one mapping of all elements in two sets. Isomorphic graphs are therefore mathematically identical, even though they may appear different.

To explain the concept of motif discovery, consider the network shown in Figure 1.2.

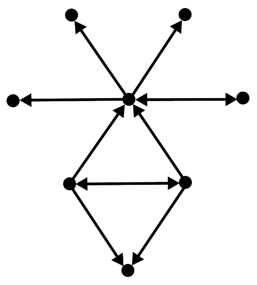


Figure 1.2: A network in which 3-node subgraphs must be found.

Figure 1.3 shows all possible directed edge configurations for subgraphs with three nodes. To easily refer to each subgraph, a set $\boldsymbol{I}$ is created, containing all possible 3-node subgraphs.

The number of occurrences of each 3-node subgraph in the network can be counted, shown as black nodes and edges in Figures 1.4(a)–1.4(f). One $i = 1$ instance is identified in Figure 1.4(a).

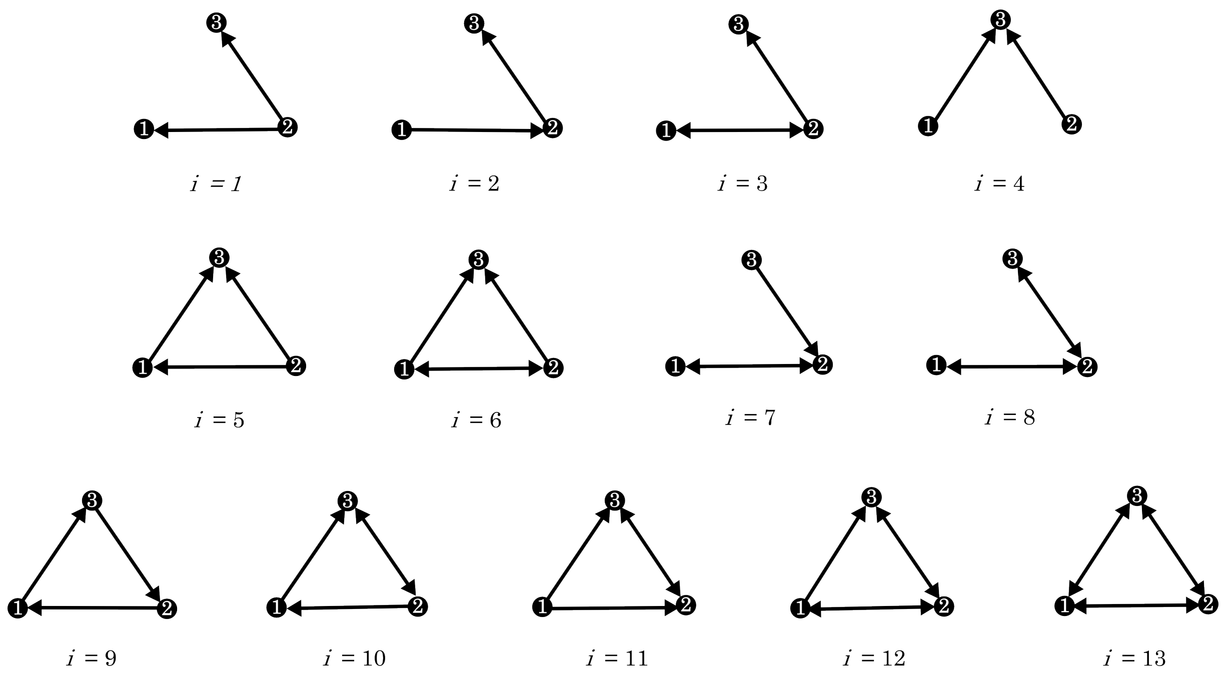
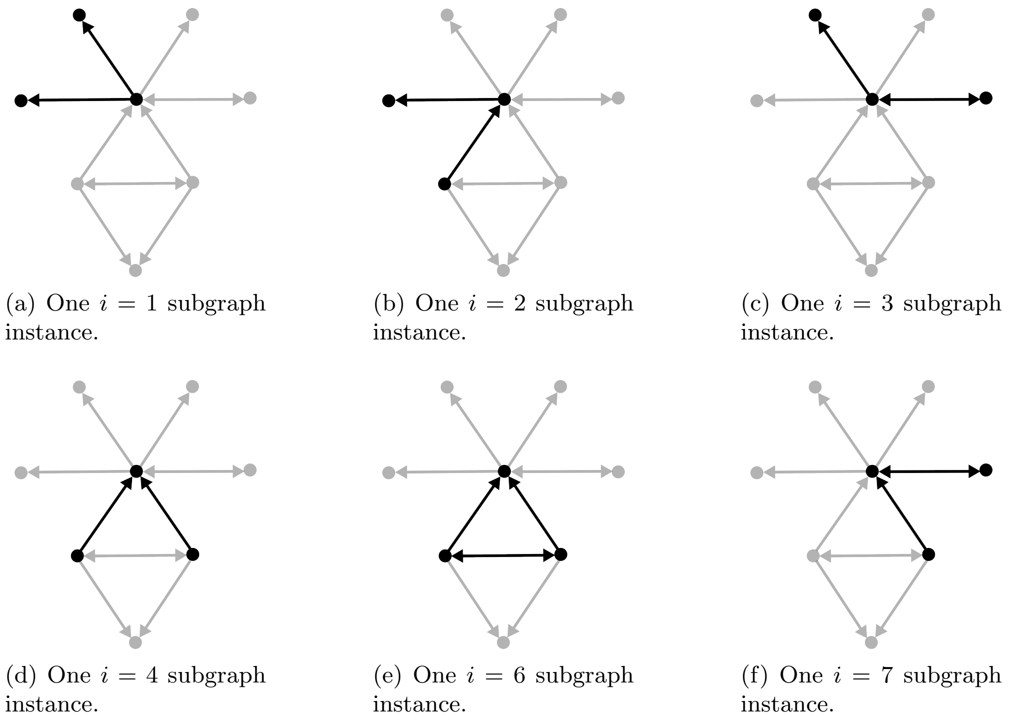


Figure 1.3: The 13 directed 3-node subgraphs.

Figure 1.4(b) shows where an $i = 2$ instance can be found, while 1.4(c) shows one $i = 3$ instance in the network. Figures 1.4(d)–1.4(f) indicate where one subgraph instance of $i = 4$, $i = 6$, and $i = 7$ can be found respectively. This is not an exhaustive list of all possible subgraphs, but gives an idea of how subgraphs can be identified in a network. All possible 3-node subgraph instances that can be identified in this network are listed in Appendix A.



(a) One $i = 1$ subgraph instance.

(b) One $i = 2$ subgraph instance.

(c) One $i = 3$ subgraph instance.

(d) One $i = 4$ subgraph instance.

(e) One $i = 6$ subgraph instance.

(f) One $i = 7$ subgraph instance.

Figure 1.4: Illustrating how subgraph instances are found in a network.

To determine which of these subgraphs are statistically significant, random networks are generated, and all subgraph instances in these are counted. If the subgraph count in the original network exceeds the mean subgraph count in the random networks, that subgraph is generally accepted as being a statistically significant motif (Milo et al., 2002; Wong et al., 2012; Kavurucu, 2015).

1.1.2 The value of studying network motifs

Discovering network motifs has been applied in a number of different fields. Wu et al. (2012) classify Wikipedia articles (based on the quality of editing) by using motif counts and ratios. In the music industry, Liu (2011) construct tonal and rhythmic complex networks, and subsequently identify network motifs which represent basic groups of chronologically ordered notes. These network motifs are used to artificially compose music. Network motifs are also used extensively in biological networks in the medical field. Tran et al. (2015) use it to better understand functional roles of genes in gene regulation; Iturria-Medina (2013) identify abnormal brain states by studying motifs found in brain networks; and Chen et al. (2013) accurately identify breast cancer patients by finding 3-node network motifs in human signaling networks. Therefore identifying and analysing network motifs have proved to be a valuable network analytic tool in a number of fields.

1.2 Research design

Wasserman and Faust (1994) state that when firm behaviour is repeated over time, it manifests as patterns or groups of nodes with specific structural properties in the complex network. Recurring structural patterns, such as motifs, can be identified in a complex network of firms. Bellamy and Basole (2013) show that there has been some translation of network properties in a supply chain context, and Borgatti and Li (2009) discuss how it can benefit supply chain researchers. In addition to these findings, Bellamy and Basole (2013) state that there are still many opportunities to identify and translate the structural properties of supply chain complex networks into a supply chain context. Therefore the following research questions are formulated in this dissertation:

1. *Can statistically significant motifs be identified in a complex network of freight movements?*
2. *Which firms frequent these patterns of interconnections?*

1.3 Research methodology

The context of this research is an empirical complex network of direct freight vehicle trips between firms. The methodology followed in this dissertation is made up of four steps.

1.3.1 Identifying a motif discovery algorithm

After reviewing available algorithms to discover statistically significant motifs, the Index-Based Subgraph Matching Algorithm with General Symmetries (ISMAGS) (Houbraken et al., 2013) and the Graph Retrieval algorithm (G-Tries) (Ribeiro and Silva, 2010) are analysed and discussed in detail. The algorithms are compared based on quantitative and qualitative properties. The former include run time, memory usage, and the number of subgraphs identified, while the latter include considerations such as the ease of extendability of the algorithm.

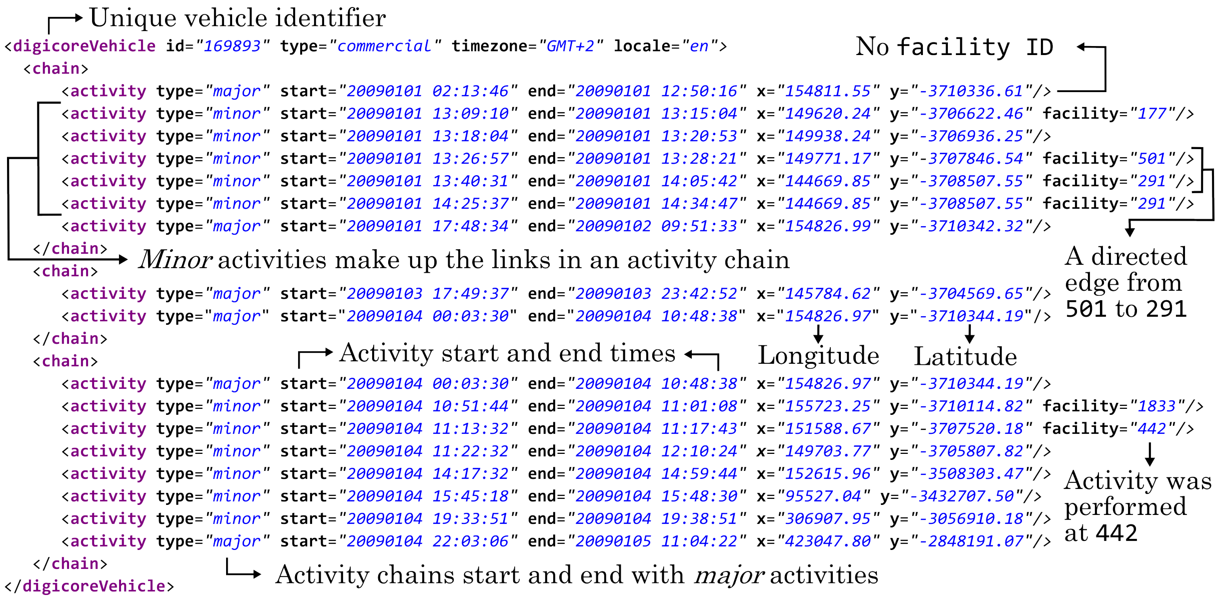
Based on the quantitative and qualitative comparison of algorithms, ISMAGS is chosen to identify statistically significant motifs in the complex network of freight movements.

1.3.2 Building the complex network

Digicore Fleet Management provided GPS data of over 40,000 freight vehicles travelling in South Africa over a six-month period, from which Joubert and Axhausen (2011) extract activity chains. The activity chains capture the duration and location of the activities, and describe the sequence in which activities are performed.

To extract activity chains from raw GPS traces, Joubert and Axhausen (2011) use ignition on- and off-signals to identify freight activities. When a freight vehicle switched off its engine for more than five hours, it is assumed to be a depot-like activity, which is called a *major* activity. These signal the start and end of an activity chain. Any activity lasting less than five hours is called a *minor* activity, and make up the remainder of the chain.

Figure 1.5 shows three activity chains of a freight vehicle, which are represented using the XML data structure. XML format creates a human-readable format to structure data. Each freight vehicle is identified by a unique identified caller a `digicoreVehicle ID`. This identifier signifies the start of the data associated with this vehicle. Since one freight vehicle may have more than one activity chain, the start and end of chains must be indicated with <`chain`> and < /`chain`> respectively. The `activity type` variable indicates whether it is a major or minor activity, and each activity has start and end times and coordinates associated with it. The Albers projection, which is a projected coordinate reference system, is used to indicate the location of activities. By using the Albers projection, the Euclidean distance between two points can be calculated, which is valuable for later analyses.


Unique vehicle identifier
No facility ID
Minor activities make up the links in an activity chain
A directed edge from 501 to 291
Activity start and end times
Longitude
Latitude
Activity was performed at 442
Activity chains start and end with major activities

```xml
<digicoreVehicle id="169893" type="commercial" timezone="GMT+2" locale="en">
 <chain>
    <activity type="major" start="20090101 02:13:46" end="20090101 12:50:16" x="154811.55" y="-3710336.61"/>
    <activity type="minor" start="20090101 13:09:10" end="20090101 13:15:04" x="149620.24" y="-3706622.46" facility="177"/>
    <activity type="minor" start="20090101 13:18:04" end="20090101 13:20:53" x="149938.24" y="-3706936.25"/>
    <activity type="minor" start="20090101 13:26:57" end="20090101 13:28:21" x="149771.17" y="-3707846.54" facility="501"/>
    <activity type="minor" start="20090101 13:40:31" end="20090101 14:05:42" x="144669.85" y="-3708507.55" facility="291"/>
    <activity type="minor" start="20090101 14:25:37" end="20090101 14:34:47" x="144669.85" y="-3708507.55" facility="291"/>
    <activity type="major" start="20090101 17:48:34" end="20090102 09:51:33" x="154826.99" y="-3710342.32"/>
  </chain>
  <chain>
    <activity type="major" start="20090103 17:49:37" end="20090103 23:42:52" x="145784.62" y="-3704569.65"/>
    <activity type="major" start="20090104 00:03:30" end="20090104 10:48:38" x="154826.97" y="-3710344.19"/>
  </chain>
  <chain>
    <activity type="major" start="20090104 00:03:30" end="20090104 10:48:38" x="154826.97" y="-3710344.19"/>
    <activity type="minor" start="20090104 10:51:44" end="20090104 11:01:08" x="155723.25" y="-3710114.82" facility="1833"/>
    <activity type="minor" start="20090104 11:13:32" end="20090104 11:17:43" x="151588.67" y="-3707520.18" facility="442"/>
    <activity type="minor" start="20090104 11:22:32" end="20090104 12:10:24" x="149703.77" y="-3705807.82"/>
    <activity type="minor" start="20090104 14:17:32" end="20090104 14:59:44" x="152615.96" y="-3508303.47"/>
    <activity type="minor" start="20090104 15:45:18" end="20090104 15:48:30" x="95527.04" y="-3432707.50"/>
    <activity type="minor" start="20090104 19:33:51" end="20090104 19:38:51" x="306907.95" y="-3056910.18"/>
    <activity type="major" start="20090104 22:03:06" end="20090105 11:04:22" x="423047.80" y="-2848191.07"/>
  </chain>
</digicoreVehicle>
```

Figure 1.5: A freight vehicle's activity chain represented in XML format.

In analysing the connectivity between firms caused by freight vehicles, Joubert and Axhausen (2013) extract a complex network from the activity chains. When two consecutive activities in an activity chain both contain `facility IDs`, a direct freight trip occurred between two interesting firms, and this trip is included in the complex network as a directed edge. One example is indicated in Figure 1.5, where a direct freight trip occurred from firm `501` to firm `291`, therefore a directed edge is created between these two nodes in the complex network.

This dissertation forms part of a research project funded by the National Research Foundation's Competitive Program for Rated Researchers (Grant 87749). The study area of the research project, and therefore also this dissertation, is the Nelson Mandela Bay Metropolitan, which is situated in the Eastern Cape province, South Africa. Only intra-area commercial traffic is considered in this dissertation, which are defined as vehicles that perform at least 60% of their activities in the study area (Joubert and Axhausen, 2011).

1.3.3 Identifying statistically significant motifs

To limit the scope of this dissertation, only 3-node motifs are identified in the complex network. This is done by implementing ISMAGS with a number of algorithms so that it is properly customised for this dissertation.

1.3.4 Analysing statistically significant motifs

The analyses are performed at three levels of detail. At a *motif-level*, it is illustrated how different activity chains of different freight vehicles cause motif instances to occur. Motifs are also ranked according to their total edge weight, and it is found that total edge weight for each motif is biased towards the edges with the highest weights. It is determined whether the relationships in each motif instance are of equal strength, or whether they have differing strengths.

At an *edge-level*, edge weights are compared to edge distances (the Euclidean distance between two firms of an edge) to determine if firms with more freight movements between them are located close to each other. Why is this interesting? The opportunity to decrease transport costs can be explored, since firms with high edge weights between them should intuitively be located close to each other.

At a *node-level*, firms are ranked based on two aspects. Firms are ranked according to the number of motifs that they appear in, after which it is determined whether considering edge weights in this ranking changes the importance of firms. It is found that these rankings are closely related to the degree centrality scores of the firms.

1.4 Document structure

Chapter 2 deals with the literature surrounding motif discovery algorithms, and two are studied in detail: ISMAGS (Houbraken et al., 2013) and G-Tries (Ribeiro and Silva, 2010). The two motif discovery algorithms are implemented in Chapter 3 by running them on smaller complex networks. The algorithms are compared based on quantitative and qualitative properties, and the chapter concludes by recommending ISMAGS to be implemented on a large-scale complex network of freight movements. The process of extracting a large-scale complex network of freight vehicle movement is explained in Chapter 4, where some summary statistics of the network are reported on. In Chapter 5, ISMAGS is used to identify statistically significant 3-node motifs in the large-scale freight complex network. A number of analyses are performed on these motif instances in this chapter to gain insight into the behaviour of the firms in the complex network. This dissertation is concluded in Chapter 6, where some recommendations for future work are made.

Chapter 2

Literature review

This chapter starts with a discussion on different ways that complex networks can be analysed. Methods to generate random networks with prescribed degree sequences are then reviewed. When determining subgraph frequencies, isomorphic graphs and frequency definitions have to be considered, therefore these topics are also discussed. The chapter ends with a detailed discussion of two algorithms that can be used to perform motif discovery, namely ISMAGS and G-Tries.

2.1 Analysing complex networks

Bellamy and Basole (2013) identify three levels of interest in studying complex network structure in a supply chain context, namely node-, edge-, and network-level properties.

2.1.1 Node-level properties of complex networks

Boccaletti et al. (2006) state that the importance of an actor in a network can be quantified using node *centrality*. Centrality scores are relative metrics, therefore evaluating each node's score in isolation will not prove useful; they must be evaluated relative to each other. Borgatti and Li (2009), Bellamy and Basole (2013) and Joubert and Axhausen (2013) mention that it is a critical node-level property. Borgatti and Li (2009) and Joubert and Axhausen (2013) agree that nodes with high centrality scores are structurally important in the network, and Bellamy and Basole (2013) state that they have more power and control over other nodes in the network. The importance of a node can be measured in a number of ways, therefore different node centrality metrics exist. The complex network in Figure 2.1 will be used to explain different centrality scores.

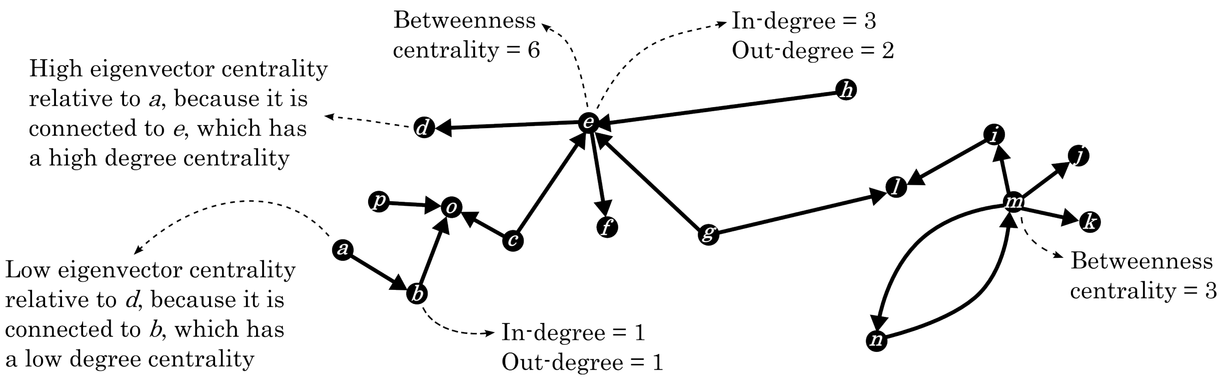

Betweenness centrality = 6
In-degree = 3
Out-degree = 2
High eigenvector centrality relative to *a*, because it is connected to *e*, which has a high degree centrality
Low eigenvector centrality relative to *d*, because it is connected to *b*, which has a low degree centrality
In-degree = 1
Out-degree = 1
Betweenness centrality = 3
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p

Figure 2.1: A complex network with directed edges.

Degree centrality is the number of nodes a node is connected to (Boccaletti et al., 2006; Borgatti and Li, 2009). Wasserman and Faust (1994) and Newman (2003) explain that in directed networks, degree centrality can be divided into in-degree centrality (the number of incoming edges), and out-degree centrality (the number of outgoing edges). Since the network in Figure 2.1 is directed, the degree centrality can also be divided into in-degree and out-degree. Node b is connected to two other nodes, with both an in- and out-degree of 1. Node e has a higher degree centrality than node b. Node e has an in-degree of 3 and out-degree of 2, which indicates that node e has access to more information than node b.

Betweenness centrality is the number of shortest paths between other nodes a node appears on. Node e has a betweenness centrality of 6, because it occurs on the shortest paths between six node pairs: c-e-d, c-e-f, g-e-d, g-e-f, h-e-f, and h-e-d. Node m has a lower betweenness centrality than node e, as it lies on the shortest paths of only three node pairs: n-m-i, n-m-j, and n-m-k. Nodes with high betweenness centrality scores are important to the health of the entire network because they can control and possibly filter flows between other nodes. They can also potentially become bottlenecks that slow down the network.

Eigenvector centrality is described by Newman (2008) to be a function of the number and quality of the connections of a node, and Borgatti and Li (2009) state that it measures whether a node is connected to well-connected nodes. Node e is central in the complex network because it scores high on all other centrality scores. Therefore any node that is directly connected to node e is considered to be "central by association", such as nodes c, d, f, g and h. Both nodes a and d have a degree centrality of 1, however, node d has a higher eigenvector centrality relative to node a, since node d is connected to node e, a central node. Node a, on the other hand, is only connected to node b, with a degree centrality of 2. Therefore node d is more central than node a in terms of eigenvector centrality.

Kim et al. (2011) use node centrality scores to determine the most central firms in three automotive supply networks, namely Honda Accord, Acura, and DaimlerChrysler Grand Cherokee. Only suppliers are considered in the study, excluding customers from the analysis. Multiple firms are identified as central, indicating that more than one firm can be considered central in the same network. The centrality scores are also used to determine which firm is most critical to the network's health. In other words, if this firm is removed from the network, it would disrupt all other firms in the network.

Other node-level properties include *structural equivalence* and *clustering coefficients*. Suppose nodes e and i are distribution centres, delivering goods to supermarkets f and l. Nodes e and i can be thought of as structurally equivalent, with similar edges connected to similar nodes. Borgatti and Li (2009) hypothesise that structurally equivalent nodes react to their environment in a similar manner and develop similar characteristics.

Nodes g and i are both connected to node l, and the clustering coefficient of these two nodes will measure the probability that nodes g and i are also connected to each other, as defined by Strogatz (2001). It is called a node's *transitivity* or *clustering* by authors such as Barabási (2007) and Newman (2003). Schilling and Phelps (2007) explain that the average clustering coefficient of the network is the average of this metric across all actors in the network.

Wen et al. (2013) use simulated data of suppliers, firms, and customers, to calculate the clustering coefficients of nodes in a complex network, and compare it to the degree centralities of the nodes. They find that the average clustering coefficient of the complex network follows a power law distribution.

In another study by Schilling and Phelps (2007), an inter-firm collaboration network is built using 1,106 firms in 11 industry-level alliance networks. A hypothesis is tested to determine whether highly clustered alliance networks that include a wide range of firms will have higher firm innovation than networks that do not possess these qualities. The hypothesis is found to be true, since being highly clustered enables a network to easily spread information within these highly clustered groups. Having a large range of firms involved in the network diversifies the information that is spread throughout the network, which ultimately leads to more innovation at firms.

2.1.2 Edge-level properties of complex networks

Edge-level properties include edge type, edge direction, and edge weight. In the automotive supply network study by Kim et al. (2011), six complex networks are studied using the concept of multiplexity: for each automotive supply network two complex networks are built, one where the edges represented material flow (flows), and another where the edges represented contractual relationships (relations).

Wen et al. (2013) define the operational relations (the movement of goods) between firms as directed edges in their complex network, while Kim et al. (2011) also use directed edges to indicate the direction of flows and interactions between firms in their automotive supply networks. When both edge weight and edge direction can be used simultaneously in a complex network, the strength and direction of relationships between two actors can be determined.

2.1.3 Network-level properties of complex networks

Edges (m, n) and (n, m) in Figure 2.1 are reciprocal, and can also be called bi-directional. Network reciprocity is defined by Wasserman and Faust (1994) as the probability of edges being reciprocal. Since there is only one bi-directional edge in the complex network in Figure 2.1, this network will have a low reciprocity. A high reciprocity will indicate that edges in one direction will have a high likelihood of being reciprocated.

Wasserman and Faust (1994) also provide a definition for the network diameter, which is the longest of all shortest paths between two nodes in a network. All edges in a complex network are potential message-carriers, and information can spread faster when a network has a low diameter.

Bellamy and Basole (2013) define network density as the proportion of edges to the maximum number of possible edges in the network. The maximum number of possible edges in the networks shown in Figure 2.2 is 15, with the dense network (shown in Figure 2.2(a)) having 13 edges, and the sparse network (shown in Figure 2.2(b)) having only 5 edges. The density for the dense network can be calculated as shown in Equation (2.1).

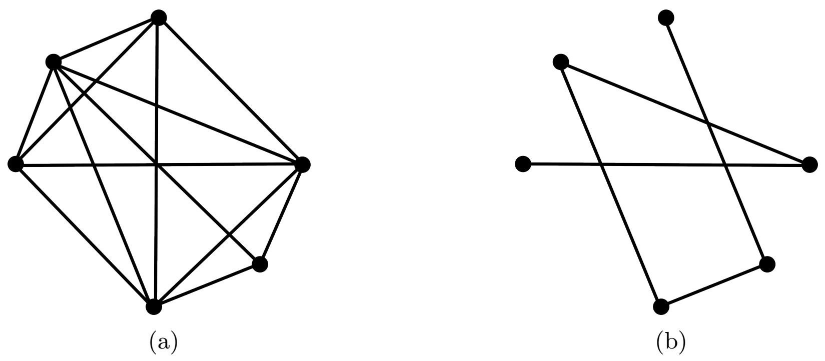


Figure 2.2: Illustrating the difference between a dense and a sparse network.

$$\text{Density} = \frac{\text{Number of edges in the network}}{\text{Maximum possible edges in the network}} = \frac{13}{15} = 0.867. \quad (2.1)$$

The density of the sparse network is much lower than that of the dense network, as shown in Equation (2.2).

$$\text{Density} = \frac{\text{Number of edges in the network}}{\text{Maximum possible edges in the network}} = \frac{5}{15} = 0.333. \tag{2.2}$$

Wasserman and Faust (1994) state that dense networks indicate tightly bounded networks, and Bellamy and Basole (2013) mention that it may place additional burden on actors to maintain and coordinate all their connections. The density of all six complex networks in the automotive study by Kim et al. (2011) are determined and one of the supply networks (Honda) have a comparatively higher density than the other two supply networks. This is attributed to the fact that Honda prefers to directly source raw materials from second- and third-tier suppliers, increasing the number of edges in their complex network.

The distribution of power in a network can be determined by plotting the total centrality score for each node. Bellamy and Basole (2013) explain that it measures the extent to which some actors are more central than others. Newman (2003) and Boccaletti et al. (2006) state that the distribution of power in a complex network is often expected to follow a power law distribution. However, Clauset et al. (2009) show that many distributions that are thought to follow a power law distribution, actually cannot be fitted to this distribution. They propose a *goodness-of-fit* test, where the null hypothesis is that the distribution follows a power law. For p-values > 0.1, the null hypothesis is accepted. When the power in a network follows a power law distribution, it means that few actors in the complex network possess high centrality scores (power), and many actors possess low centrality scores. Wen et al. (2013) determine that the power in their complex network is distributed according to a power law function, while Joubert and Axhausen (2013) determine that the power in their complex network is distributed according to a truncated power law function.

The above-mentioned complex network properties will be applied on the freight complex network to gain insight into the connectivity of firms. In the remainder of the chapter, concepts and algorithms for motif discovery are discussed.

2.2 Concepts for motif discovery

Generally speaking, there are four steps to discover statistically significant motifs:

1. Generate random networks.
2. Determine subgraph frequency in the original network.
3. Determine subgraph frequencies in the random networks.
4. Determine statistical significant motifs using metrics and thresholds.

In Section 2.2.1 two algorithms that can be used to generate random networks with prescribed degree sequences are discussed. Thereafter, graph isomorphism is described in Section 2.2.2. This is an important concept in motif discovery since it has an impact on the frequencies of subgraphs, which are discussed in Section 2.2.3.

2.2.1 Random network generation

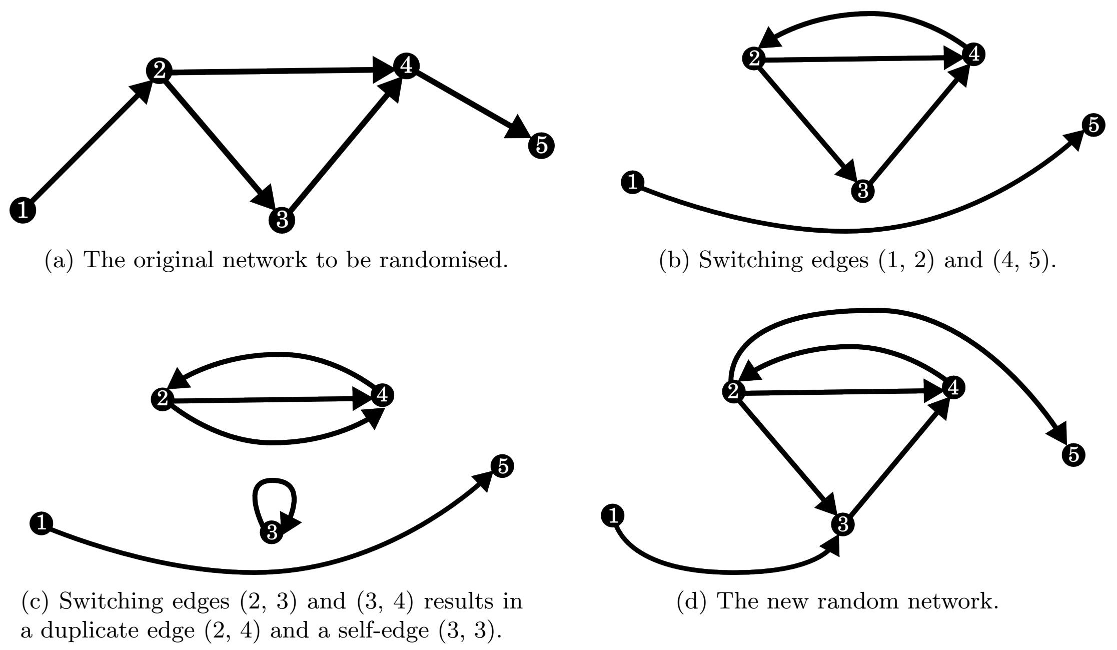
Castro and Azevedo (2012) state that the null hypothesis in motif discovery is that the subgraph instances have no statistical significance, and that the actual subgraph count is similar to the expected one. Therefore when the subgraph count exceeds the expected count, the subgraphs are significant and the null hypothesis can be rejected in favour of the alternative hypothesis (that the subgraph count is greater than the expected count).

To determine the expected subgraph counts, random networks are generated. The random networks are known as null models, and have the same degree distribution as the original network, as shown in the seminal work by Milo et al. (2002). The frequency of subgraphs are determined in each random network, and the average subgraph count is taken as the expected subgraph count. Tran et al. (2015) explain that this can be used as a baseline for comparison against the actual subgraph count in the original network.

For random networks to have the same degree sequence as the original network, it must have the same number of nodes, and each node must have the same in- and out-degree. In addition, the random networks cannot have any duplicate edges or self-edges (edges with the same origin and destination node). There are a number of algorithms to generate random networks, but for purposes of this dissertation, the switching and the matching algorithms as described by Milo et al. (2014), are discussed.

The switching algorithm

Milo et al. (2014) describe the switching algorithm as the random switching of edge ends. Consider the following example along with Algorithm 1. Suppose a random network has to be generated, and must have the same degree sequence as the network shown in Figure 2.3(a). The algorithm starts by randomly selecting two edges, say (1, 2) and (4, 5) (lines 6–7). It then switches the ends of the edges to create two new edges (1, 5) and (4, 2). If no self-edges or duplicate edges are created using this switch (line 8), the algorithm accepts the two new edges (line 9), resulting in the altered network shown in Figure 2.4(b). When the algorithm tries to switch the ends of the edges (2, 3) and (3, 4), a duplicate (2, 4) and self-edge (3, 3) will be created, shown in Figure 2.3(c). Therefore the algorithm does not accept this switch and randomly chooses two new edges to switch, say (2, 3) and (1, 5). Again, the ends of the edges are switched to create the altered network shown in Figure 2.3(d). The algorithm continues in this fashion for QE times, where Q is chosen large enough to ensure proper randomisation (Kavurucu (2015) suggests a value of 100), and E is the number of edges in the original network.



(a) The original network to be randomised.
(b) Switching edges (1, 2) and (4, 5).
(c) Switching edges (2, 3) and (3, 4) results in a duplicate edge (2, 4) and a self-edge (3, 3).
(d) The new random network.

Figure 2.3: Generating a random network using the switching algorithm.

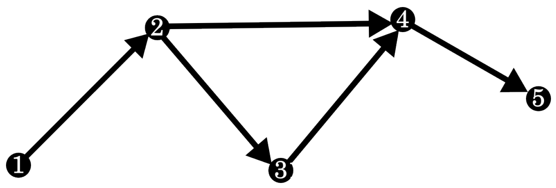
The matching algorithm

Suppose the matching algorithm has to be used to generate a random network with the same degree distribution as the network shown in Figure 2.4(a). Algorithm 2 describes the steps

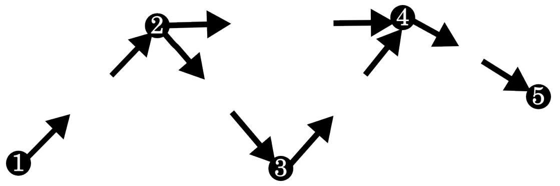
Algorithm 1 The switching algorithm.

```
Input: Edge list $\boldsymbol{E}$
Output: Random edge list $R$
 1: function SWITCHINGALGORITHM(edge list $\boldsymbol{E}$)
 2:     $Q \leftarrow 100$
 3:     $E \leftarrow$ number of edges in $\boldsymbol{E}$
 4:     copy all edges from $\boldsymbol{E}$ to $R$
 5:     for all $i \in 1 : QE$ do
 6:         $(o_1, d_1) \leftarrow$ first random edge
 7:         $(o_2, d_2) \leftarrow$ second random edge
 8:         if $o_1 \neq d_2 \wedge o_2 \neq d_1 \wedge R$ does not contain $(o_1, d_2)$ or $(o_2, d_1)$ then
 9:             switch ends to create $(o_1, d_2)$ and $(o_2, d_1)$
10:         end if
11:     end for
12:     return new random network $R$
13: end function
```

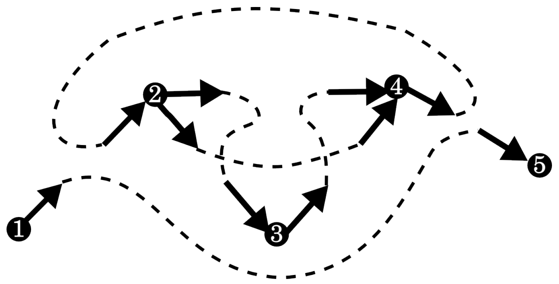
followed. Milo et al. (2014) explain that the matching algorithm starts by splitting all edges into incoming and outgoing edge stubs, as shown in Figure 2.4(b) (line 2–7). This will ensure that each node in the random network has the same degree as it had in the original network. In line 5 the algorithm randomly matches incoming and outgoing stubs (shown in Figure 2.4(c)) to create a random network. Sometimes self-edges or duplicate edges can be created during this step, and this is checked in line 7. The algorithm can either discard the entire network and reassign edge stubs, or simply reassign the problematic edge stubs. The former solution can result in unnecessary computations, therefore the latter method is preferred (line 11). After all the self-edges and duplicate edges have been resolved, the random network shown in Figure 2.4(d) is created.



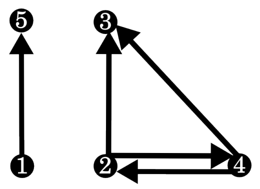
(a) The original network.



(b) Assigning incoming and outgoing edge stubs to each node.



(c) Matching incoming and outgoing stubs with each other.



(d) The new random network.

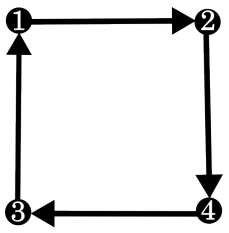
Figure 2.4: Generating a random network using the matching algorithm.

Algorithm 2 The matching algorithm.

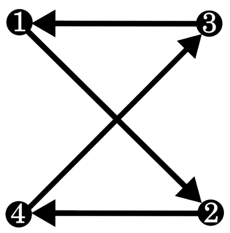
```
Input: Edge list $\boldsymbol{E}$
Output: Random edge list $R$
 1: function MATCHING(edge list $\boldsymbol{E}$)
 2:     list of nodes $O \leftarrow$ all origin nodes
 3:     list of nodes $D \leftarrow$ all destination nodes
 4:     for all origin nodes $o \in O$ do
 5:         $d \leftarrow$ randomly chosen destination node from $D$
 6:         create possible edge $\leftarrow (o, d)$
 7:         if $o \neq d \wedge R$ does not contain $(o, d)$ then
 8:             new edge is not duplicate edge or self-loop
 9:             add $(o, d)$ to $R$
10:         else
11:             randomly choose new $d$
12:         end if
13:     end for
14:     return new random network $R$
15: end function
```

2.2.2 Graph isomorphism

Motif discovery depends on the subgraph frequencies in the original and random networks. If the subgraph frequencies are incorrect, then the wrong subgraphs will be identified as motifs. But how can subgraph frequencies be incorrect? Consider subgraph A in Figure 2.5(a). It is isomorphic (mathematically identical) to subgraph B in Figure 2.5(b), even though they appear as two different subgraphs. These subgraphs are isomorphic because there exists a one-to-one mapping of the nodes of subgraph A (N_A) to the nodes in subgraph B (N_B): $N_A \rightarrow N_B = \{(1 \rightarrow 1), (2 \rightarrow 2), (3 \rightarrow 3), (4 \rightarrow 4)\}$. Therefore both subgraphs have the same adjacency matrix, shown in Figure 2.5(c).



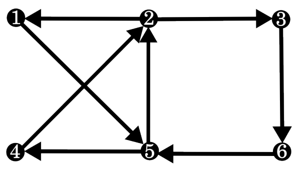
(a) Subgraph A.



(b) Subgraph B.

$$\begin{array}{c|cccc} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} \\ \mathbf{1} & 0 & 1 & 0 & 0 \\ \mathbf{2} & 0 & 0 & 0 & 1 \\ \mathbf{3} & 1 & 0 & 0 & 0 \\ \mathbf{4} & 0 & 0 & 1 & 0 \end{array}$$

(c) Adjacency matrix of subgraph A and B.



(d) Network in which subgraph A must be found.

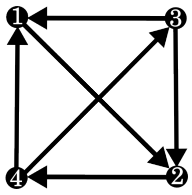
Figure 2.5: Illustrating how isomorphic subgraphs influence subgraph frequencies.

Suppose the frequency of subgraph A has to be determined in the network shown in Figure 2.5(d). If you do not consider the isomorphic subgraph B, only one subgraph instance (2, 3, 6, 5) will be identified. But when the subgraph B is also searched for in the network, the subgraph instance (1, 5, 4, 2) is also be identified, increasing the frequency of subgraph A to 2.

Fortin (1996) explains that there are two methods for determining whether two graphs are isomorphic. The first is to directly find isomorphisms by matching nodes to each other, the second is by using canonical labels. The canonical label of a graph is a string (or code) of values describing the connectivity in the graph. The canonical representation is usually taken as the concatenation of the rows or columns of a graph's adjacency matrix. For example, by concatenating the rows of the adjacency matrix in Figure 2.5(c), the canonical label '0 1 0 0 0

0 0 1 1 0 0 0 0 0 1 0' can be generated. When two graphs have the same canonical label, they are considered isomorphic.

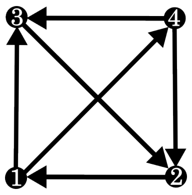
Using canonical labeling is not always a fail-safe method of determining isomorphic subgraphs. Consider subgraph C in Figure 2.6(a) and subgraph D in Figure 2.6(c). These subgraphs are isomorphic because there exists a one-to-one mapping between the nodes of the two subgraphs: $N_C \to N_D = \{(1 \to 3), (2 \to 2), (3 \to 4), (4 \to 1)\}$. Due to the different labelling of the nodes of each subgraph, the adjacency matrices differ, as shown in Figures 2.6(b) and 2.6(d). Therefore each subgraph will have a unique canonical label and will not be identified as isomorphic subgraphs. To overcome this, Fortin (1996) suggests relabelling the nodes to generate either the lexicographically smaller or larger adjacency matrices.



(a) Subgraph C.

$$\begin{array}{c|cccc} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} \\ \mathbf{1} & 0 & 1 & 0 & 0 \\ \mathbf{2} & 0 & 0 & 0 & 1 \\ \mathbf{3} & 1 & 1 & 0 & 0 \\ \mathbf{4} & 1 & 0 & 1 & 0 \end{array}$$

(b) Adjacency matrix of subgraph C.



(c) Subgraph D.

$$\begin{array}{c|cccc} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} \\ \mathbf{1} & 0 & 0 & 1 & 1 \\ \mathbf{2} & 1 & 0 & 0 & 0 \\ \mathbf{3} & 0 & 1 & 0 & 0 \\ \mathbf{4} & 0 & 0 & 1 & 0 \end{array}$$

(d) Adjacency matrix of subgraph D.

Figure 2.6: Two isomorphic graphs and their adjacency matrices.

Lexicographical order can be compared to alphabetical order: it determines how strings should be ranked according to some predefined order. To rank two strings, ABC and ABD, lexicographical order can be used. The first character in the strings that differs is in position 3: C in string 1 and D in string 2. Since $C < D$ in the alphabet, string 1 is lexicographically smaller than string 2, and the inverse is also true (string 2 is lexicographically larger than string 1). Similarly canonical labels can be compared according to lexicographical order. When comparing two canonical labels, '0 1 1 0' and '0 1 1 1', both labels have the same numbers in positions 1–3 ('0 1 1'). The first number that differs between the two labels is in position 4 (label 1 has '0' and label 2 has '1' in this position). Since $0 < 1$, label 1 is lexicographically smaller than label 2.

Motif discovery algorithms always consider subgraph isomorphism when determining subgraph frequencies, which are discussed in the next section.

2.2.3 Determining subgraph frequencies

According to Kavurucu (2015), determining subgraph frequencies requires solving the computationally complex graph isomorphism problem, which is discussed in the previous section. Other concepts to consider when determining subgraph frequencies are frequency definitions and scoring thresholds and metrics.

Frequency definitions

Frequencies of subgraphs can either be determined using estimation algorithms such as that of Kashtan et al. (2004) and Slota and Madduri (2014), or by exact counting. Tran et al. (2015) explain that exact counting is usually performed when looking for smaller subgraphs with six nodes or less, while estimation is performed on larger subgraphs. This dissertation focuses on the exact counting of motifs.

Determining statistically significant motifs is heavily dependent on the frequency of subgraphs, therefore Schreiber and Schwöbbermeyer (2005) defined three types of frequency:

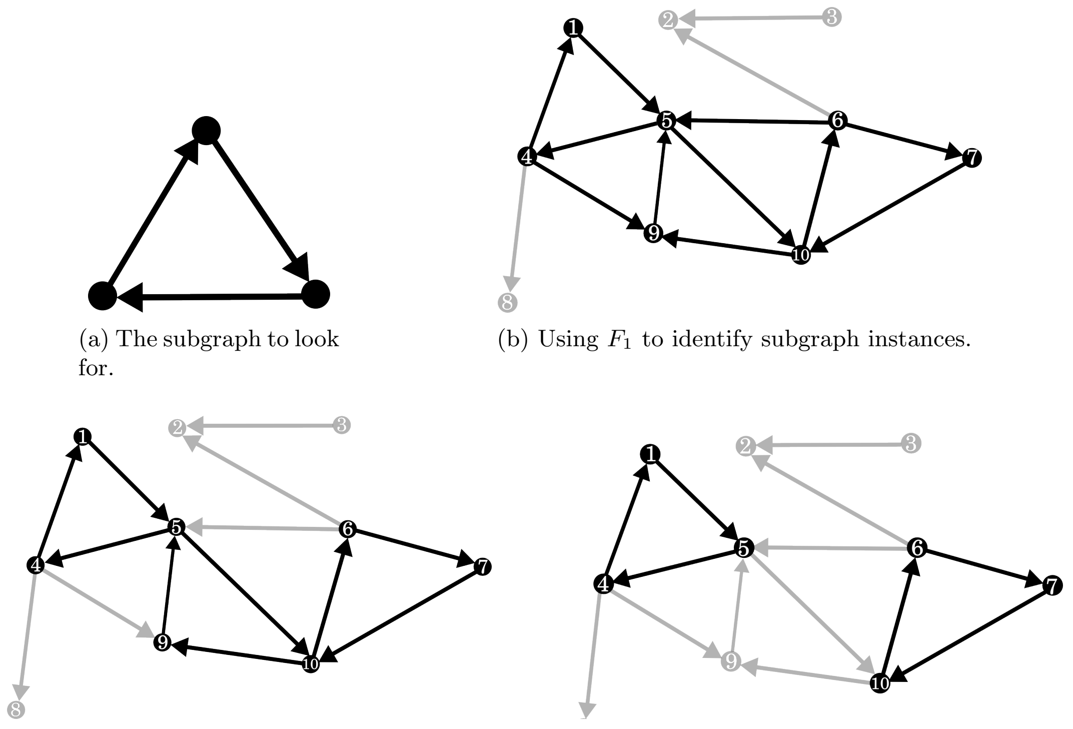
1. F_1, edges and nodes may be shared by different subgraph instances.

2. F_2, only nodes may be shared between different subgraph instances.

3. F_3, neither edges nor nodes may be shared by different subgraph instances.

The frequency definition depends on the algorithm and its application field. In some biological networks, for example, F_3 must be used because the edges represent an action that removes a node from the network. However, most algorithms implement the F_1 frequency definition.

Using the different frequency definitions can result in different frequencies of a particular subgraph. Suppose the number of occurrences of the subgraph shown in Figure 2.7(a) must be found. According to this definition, five subgraph instances are identified in Figure 2.7(b) ((1, 5, 4), (4, 9, 5), (5, 10, 9), (5, 10, 6) and (6, 7, 10)). When using F_2 to determine subgraph frequency, only node overlap is allowed and three subgraph instances can be identified ((1, 5, 4), (5, 10, 9), and (6, 7, 10)), as shown in Figure 2.7(c). Figure 2.7(d) shows the two instances ((1, 5, 4) and (6, 7, 10)) that can be identified when F_3 is used. Notice that the term *subgraph* refers to the configuration of edges that have to found in a network, and the subgraph *instance* refers to the actual network nodes that have the same edge configuration.



(a) The subgraph to look for.

(b) Using F_1 to identify subgraph instances.

(c) Using F_2 to identify subgraph instances.

(d) Using F_3 to identify subgraph instances.

Figure 2.7: Illustrating the effect that different frequency definitions have on frequency values.

After subgraph frequencies are determined in both the original and random networks, statistically significant motifs must be identified. For this purpose, different thresholds and metrics have been developed, which are discussed next.

Thresholds and metrics

Different thresholds and metrics may result in identifying different sets of statistically significant motifs, therefore using more than one method is desirable. In this way, the results can be compared and a decision can be made over which set of motifs to choose. Thresholds are based on the minimum number of occurrences of a motif (set by the user) for it to be considered statistically significant, and metrics are standard measurements of significance.

A frequency threshold requires that a motif appears at least f times in the original network for it to be considered a motif. Usually, f is the mean frequency of the subgraph in the random networks, $\bar{f}_{\text{rand}}$. The uniqueness threshold requires that a subgraph has at least u occurrences more in the original network than on average in the random network ensemble to be considered a motif.

Wong et al. (2012) list a number of metrics to determine statistical significance, such as the z-score, significance profiles, abundance and concentration. Threshold and metric calculations make use of a set $\boldsymbol{I}$, which contains all possible subgraphs of a particular size.

The z-score is calculated as

$$z_i = \frac{f_{i,\text{in}} - \bar{f}_{i,\text{rand}}}{\sqrt{\sigma^2_{i,\text{rand}}}}, \tag{2.3}$$

where z_i is the z-score for subgraph $i \in \boldsymbol{I}$, $f_{i,\text{in}}$ is the number of subgraph $i \in \boldsymbol{I}$ instances in the original network, $\bar{f}_{i,\text{rand}}$ is the mean frequency of subgraph $i \in \boldsymbol{I}$ instances in the set of random networks, and $\sqrt{\sigma^2_{i,\text{rand}}}$ is the standard deviation of subgraph $i \in \boldsymbol{I}$ instances in the random networks (Milo et al., 2002; Tran et al., 2015). Milo et al. (2002) explain that subgraphs in large networks will have higher z-scores, since the frequencies of subgraphs will be higher in these networks. They suggest that a z-score greater than 2.0 indicates a statistically significant motif, however z-scores can be normalised to 1 for easier comparison.

Milo et al. (2002) describe the significance profile as the relative significance of motifs, since it is based on normalised z-scores. This is in contrast to the absolute significance calculated by the z-scores. The significant profile for a subgraph is calculated to be

$$sp_i = \frac{z_i}{\sqrt{\sum\limits_{i=1}^{n} z_i^2}}, \tag{2.4}$$

where sp_i is the significance profile value for subgraph $i \in \boldsymbol{I}$, z_i is the z-score for subgraph $i \in \boldsymbol{I}$, and n is the number of subgraphs in the set $\boldsymbol{I}$. Subgraphs that have an sp_i value greater than 0 is considered to be a statistically significant motif.

(Kashtan et al., 2004) mentioned that the abundance ($\triangle_i$) of subgraph $i \in \boldsymbol{I}$ is similar to the z-score, and is calculated as follows:

$$\triangle_i = \frac{f_{i,\text{in}} - \bar{f}_{\text{rand}}}{f_{\text{in}} + \bar{f}_{i,\text{rand}} + \epsilon}, \tag{2.5}$$

where ϵ is a small positive integer ensuring that the abundance value does not approach infinity (Wong et al., 2012). Abundance values range from -1 to 1, where the former indicates that the subgraph is under-represented, and the latter indicates that the subgraph is over-represented and therefore statistically significant.

The concentration (C_i) of a subgraph $i \in \boldsymbol{I}$ compares the frequency of a subgraph to the frequencies of other subgraphs of the same size. The concentration of n-node subgraph of type i is calculated as

$$C_i = \frac{N_i}{\sum\limits_{i=1}^{n} N_i}, \tag{2.6}$$

where N_i is the number of occurrences of subgraph $i \in \boldsymbol{I}$. Therefore the concentration of subgraphs of a particular size is the ratio between the number of occurrences of that subgraph and the total number of subgraphs of the same size in the original network (Kashtan et al., 2004).

This section dealt with core concepts for motif discovery, namely the generation of random networks, subgraph isomorphism, and methods of identifying statistically significant motifs. In the next section, the application of these concepts are explained by two motif discovery algorithms.

2.3 Motif discovery algorithms

Two algorithms are considered for implementation in this study, namely the G-Tries algorithm (Ribeiro and Silva, 2010) and the IMSAGS algorithm (Houbraken et al., 2013).

2.3.1 The G-Tries algorithm

This section explains key aspects of the G-Tries algorithm, which is developed in the PhD thesis of Ribeiro (2011) and published by Ribeiro and Silva (2010).

The g-tries (Graph Retrieval) data structure is similar to a prefix tree, which exploits common substructures. Figure 2.8 shows how three words (ball, bale, and bake) can be represented as a prefix tree. All descendants of a node have the same prefix, and each path down the prefix tree represents a unique word. This makes it an efficient data structure, since common prefixes are only stored once.

This prefix tree can be used to efficiently search for these three words in a paragraph. To identify all occurrences of these words, all words starting with a *b* are identified. The first node of the tree is matched to the first letter in the identified words, *b*. Using the identified words, those that have an *a* as the second letter are identified, and the second tree node is matched to these letters. These words are all possible matches to the words *ball*, *bale*, and *bake*. When there are no more descendant nodes and letters to match, a word in the prefix tree has been found.

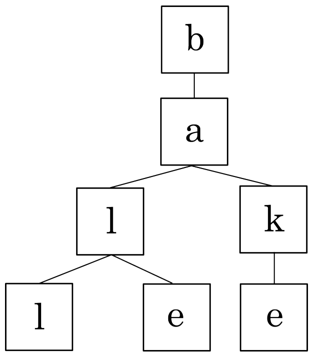
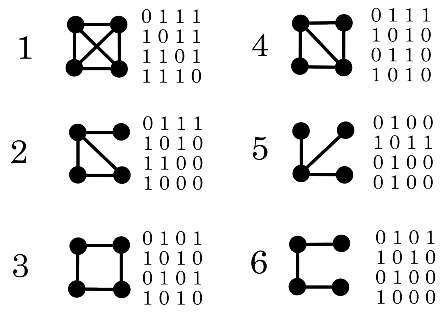
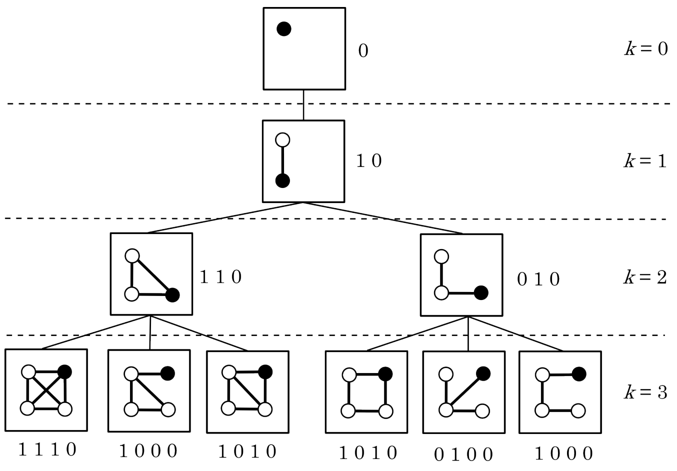


Figure 2.8: A prefix tree containing three words with similar substructures.

Similar to words with common prefixes, graphs can share common subgraphs, hence the development of the G-Tries algorithm. When referring to the algorithm, the term *G-Tries* is used, and when referring to the data structure(s), the term *g-trie(s)* is used. Each g-trie node represents a subgraph node, which is characterised by its ancestor nodes. The six undirected subgraphs (with their adjacency matrices) shown in Figure 2.9(a) can be represented as the g-trie shown in Figure 2.9(b). With each new g-trie node added to the g-trie, another graph node is added to the subgraph. The added node is shown as a black node, while the ancestors of that node are shown as white nodes. With each new graph node added, its connections to other nodes are known, and its partial adjacency matrix can therefore be determined (shown to the right of each g-trie node). Note that while the following examples explain G-Tries with undirected subgraphs, the same principles can be applied to directed subgraphs, which are used in this dissertation.



(a) Six undirected subgraphs with their corresponding adjacency matrices.



(b) The g-trie data structure representing the six undirected subgraphs.

Figure 2.9: A g-trie data structure representing six undirected subgraphs.

In the next subsection a custom canonical form to deal with isomorphic graphs is explained. Then, the iterative addition of subgraphs into a g-trie is discussed, after which the subgraph census using a constructed g-trie is explained.

Custom canonical labelling of graphs

Isomorphic graphs can produce different adjacency matrices, even though they essentially represent the same graph. G-Tries creates a g-trie from a graph's adjacency matrix, therefore isomorphic graphs should produce one single path down the same g-trie. However, some isomorphic graphs may have different adjacency matrices, and therefore result in two paths down a g-trie, making the g-trie an inefficient data structure. Therefore G-Tries implements custom canonical labels to ensure that isomorphic graphs will always be represented by the same adjacency matrix.

When comparing the g-trie size for a lexicographically larger and smaller canonical label, the former creates a smaller g-trie than the latter. The smaller g-trie is preferred, but these canonical labels do not ensure that each new subgraph node added to the g-trie has as many as possible edges to its ancestor nodes. This is a requirement set by Ribeiro (2011), and to overcome this, the custom canonical form is implemented.

The steps to create a custom canonical label of a graph is depicted in Algorithm 3. The algorithm starts in line 2 by getting any canonical label of the input graph. The popular NAUTY algorithm, which is the fastest known algorithm for detecting isomorphic graphs (McKay, 1981), is used in this instance. Iterating through all nodes in the graph (lines 3–6), tables containing node degrees are created. The core steps of the algorithm are depicted in lines 7–20. The algorithm iteratively searches for the node with the least possible edges (lines 10–14), ensuring that this node is not an articulation point that splits the graph in two (line 8). The chosen node, u_{min} is then labeled in line 17. Before proceeding to the next iteration, tables are updated in lines 18 and 19. When the algorithm has labeled all nodes in the graph, it returns the new custom canonical label of the graph, c (line 21).

Constructing a g-trie

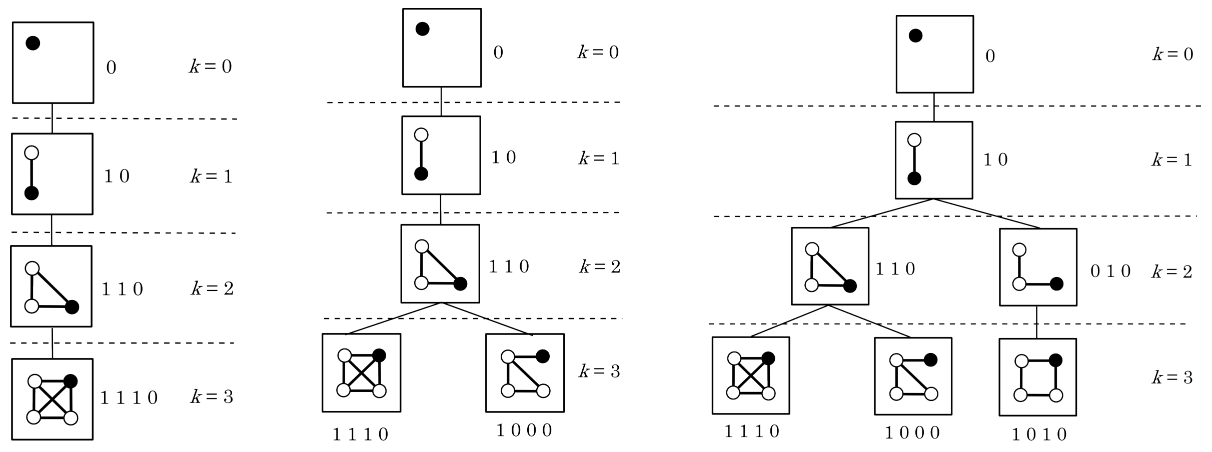
Algorithm 4 depicts the steps to construct a g-trie, t, from a given subgraph, s. The algorithm starts by determining the custom canonical adjacency matrix of the subgraph by calling the

Algorithm 3 Creating a custom canonical label for a graph.

```
Input: Graph, $g$
Output: Canonical string of $g$, $c$
 1: function GETCANON(graph $g$)
 2:     $c \leftarrow$ NAUTY($g$)
 3:     for all nodes $i \in V(g)$ do
 4:         currentDegree[$i$] $\leftarrow$ total number edges connected to $i$
 5:         globalDegree[$i$] $\leftarrow$ previousDegree[$i$] $\leftarrow$ currentDegree[i]
 6:     end for
 7:     for all position $p \in |\boldsymbol{V}| : 1$ do
 8:         if node $\boldsymbol{V}[p]$ is not labelled and not an articulation point then
 9:             $u_{\min} \leftarrow \boldsymbol{V}[p]$
10:         else if node V(g)[$p$] has minimum currentDegree then
11:             $u_{\min} \leftarrow \boldsymbol{V}[p]$
12:         else if there exists a tie then
13:             $u_{\min} \leftarrow$node with minimum previousDegree
14:         else if a new tie then
15:             $u_{\min} \leftarrow$node with minimum globalDegree
16:         end if
17:         $c[u_{\min}] \leftarrow p$
18:         previousDegree[] $\leftarrow$ currentDegree[]
19:         remove edges of $u_{\min}$ from currentDegree[]
20:     end for
21:     return canonical label $c$
22: end function
```

GETCANON function (line 2). The function INSERTRECURSIVE then recursively traverses the g-trie, inserting new children nodes when necessary.

Suppose subgraph 1 in Figure 2.9(a) must be added to an empty g-trie. Figure 2.10(a) shows the completed g-trie representing only this subgraph.



(a) Adding subgraph 1 to an empty g-trie.

(b) Adding subgraph 2 to the existing g-trie.

(c) Adding subgraph 3 to the existing g-trie.

Figure 2.10: Adding subgraphs 1–3 (Figure 2.9(a)) to an initially empty g-trie.

The first subgraph node of subgraph 1 is added to the g-trie ($k = 0$ in Figure 2.10(a)). Its partial adjacency matrix is the first $k + 1$ values in the k^{th} row of the adjacency matrix. Subgraph 1 has '0' in the [0, 0] position, which is added to the first g-trie node. Note that the

Algorithm 4 Inserting a subgraph into a g-trie.

```
Input: Subgraph $s$, g-trie $t$
Output: g-trie $t$
procedure INSERTGTRIE(subgraph $s$, g-trie $t$)
    $m \leftarrow$ GETCANON($g$)
    INSERTRECURSIVE($m$, $t$, 0)
end procedure
function INSERTRECURSIVE(matrix $m$, g-trie $t$, index $k$)
    if $k$ = number of rows of $m$ then
        $c$.isGraph $\leftarrow$ true
    else
        for all children $c$ of $t$ do
            if ($c$.out $= m[0 : k+1, k]$) and ($c$.in $= m[k, 0 : k+1]$) then
                INSERTRECURSIVE($m$, $c$, $k+1$)
                return $t$
            else
                $nc \leftarrow$ new g-trie node
                $nc$.in $\leftarrow m[k, 0 : k+1]$
                $nc$.out $\leftarrow m[0 : k+1, k]$
                $nc$.isGraph $\leftarrow$ false
                $t$.insertChild($nc$)
                INSERTRECURSIVE($m$, $nc$, $k+1$)
            end if
        end for
    end if
end function
```

indices in the adjacency matrices start at 0, while the nodes are numbered from 1.

The second, third, and fourth subgraph nodes are added recursively in lines 14–19 using $k = 1$, $k = 2$, and $k = 3$, resulting in the complete g-trie shown in Figure 2.10(a). The partial adjacency matrices of each new g-trie node are calculated as the first $k + 1$ values in the k^{th} row of subgraph 1's adjacency matrix, which are determined to be '1 0', '1 1 0', and '1 1 1 0'. The subgraph is now complete, and the last g-trie node's *isGraph* value is set to *true* in line 7. *isGraph* is a boolean variable indicating whether this g-trie node represents a completed subgraph.

When adding subgraph 2 in Figure 2.9(a) to the existing g-trie in Figure 2.10(a), the algorithm starts again with $k = 0$. The adjacency matrices of subgraph 1 and subgraph 2 have the same value in position [0, 0] = 0, therefore line 10 will return a *true* value. The *in* and *out* values of each graph node corresponds to the first $k + 1$ values in the k^{th} row (*in*) and the first $k + 1$ values in the k^{th} column of m (*out*), where m is the adjacency matrix for this subgraph. For undirected subgraphs, the adjacency matrix is symmetrical, resulting in the same values for *in* and *out*. For directed subgraphs, these values differ. No new g-trie nodes are added, and k is incremented by one. The next partial adjacency matrix corresponds to the position [0:1, 1] when $k = 1$, and again, subgraph 1 and subgraph 2 have the same values in this position ('1 0'). k is incremented, and the algorithm determines that the adjacency matrices of subgraph 1 and subgraph 2 have the same values in position [0:2, 2]. Again, the algorithm continues without adding a g-trie node.

When $k = 3$, line 10 will return a *false* value, because c.in (and c.out) = '1 1 1 0', and [0:3, 3] = '1 0 0 0'. Therefore a new g-trie node is added for the latter partial adjacency matrix. The g-trie in Figure 2.10(b) now contains both subgraph 1 and 2. In a similar way, subgraph 3 can be added to this g-trie, resulting in the altered g-trie shown in Figure 2.10(c). When $k = 2$, a new g-trie node is created, creating the third path down the g-trie.

Using the constructed g-trie to count all subgraph instances in a network

Algorithm 5 shows the steps followed to find all subgraph instances in a g-trie in a graph.

The algorithm starts in line 3 by calling the recursive procedure MATCH for all children of a g-trie node. At this point, $\boldsymbol{V}_{\text{used}}$ is an empty set, denoted by $\emptyset$. $\boldsymbol{V}_{\text{used}}$ is a collection of nodes that is a partial match of graph nodes to a g-trie path. The procedure MATCH then creates a set of nodes ($\boldsymbol{V}$) that matches the current g-trie node (line 7). In lines 8–15 the algorithm traverses the set $\boldsymbol{V}$ and recursively tries to expand it through all possible g-trie paths (lines 12–13). If a full subgraph can be matched to a g-trie node, then line 9 returns a *true* value and the algorithm has found a complete instance of the subgraph.

The MATCHINGVERTICES function (line 17) generates a set of matching nodes. A set of candidate nodes ($\boldsymbol{V}_{\text{cand}}$) is created in line 19, and all graph nodes are viable candidates if the current node is a root child (line 18). If the current node is not a root child, the selection is made in line 21 from the set of nodes connected to the new node (that has already been matched). From this set, the node with the smallest neighbourhood is selected in line 22, therefore reducing the possible candidates to the unused neighbours (line 23).

The set of candidates is then traversed in lines 26–30 to check whether the candidate respects all connections to ancestors (line 27). If it does, the node is added to the set of matching nodes, called $\boldsymbol{Vertices}$.

After all matches have been found, the algorithm terminates and returns the set of subgraph instances, $\boldsymbol{Vertices}$ (line 31).

Symmetry breaking

Katebi et al. (2012) define a *symmetry* as a permutation of the nodes of the graph that preserves the edge relations. A symmetry is also called an *automorphism* of the graph, and the group of symmetries of a graph is called the *automorphism group* or the *symmetry group*, denoted by $Aut(G)$. If a symmetry in an automorphism group is found, it means that all other symmetries in that group has also been found, therefore the search can be terminated.

Algorithm 6 explains the steps G-Tries follows to break symmetries in subgraphs. It starts in line 2 with an empty set of symmetry breaking conditions, and the automorphisms of the subgraph are calculated in line 3. Lines 4 – 10 iteratively adds constraints to the set $\boldsymbol{C}$. In line 5, it finds the minimum index m corresponding to a node that has at least one equivalent node. The condition (that the node in position m should have an index lower than every other equivalent position) is added to $\boldsymbol{C}$ in line 7. Therefore m is fixed in its position. Line 9 removes automorphisms from a that do not respect the newly added condition. When only the identity automorphism is left in a, the set of symmetry breaking conditions is returned in line 11

As G-Tries matches g-tries to network nodes in order to find subgraph instances, it always ensures that the symmetry breaking conditions are adhered to. Therefore the subgraph instances found by G-Tries do not contain any symmetric subgraphs.

G-Tries is able to generate random networks with prescribed degree sequences, and use z-scores to determine which subgraphs are statistically significant motifs. This intelligence does not form part of the ISMAGS algorithm, which is described in the next section.

2.3.2 The ISMAGS algorithm

The Index-Based Subgraph Matching Algorithm with General Symmetries (ISMAGS) is a general subgraph matching algorithm developed by Houbraken et al. (2013). It only determines the frequency of a subgraph and does not determine whether it is a motif or not. The *Java* source code is freely available on Maarten Houbraken's *GitHub* account (Houbraken, 2015). It can be

Algorithm 5 Performing subgraph census using g-tries.

```
Input: Graph $g$, g-trie $t$
Output: all occurrences of subgraphs in $t$
 1: procedure GTRIEMATCH(graph $g$, gtrie $t$)
 2:     for all children $c$ of $t.root$ do
 3:         MATCH($c$, $\emptyset$)
 4:     end for
 5: end procedure
 6: procedure MATCH(gtrie $t$, vertices $V_{\text{used}}$)
 7:     $V \leftarrow$ MATCHINGVERTICES($t$, $\boldsymbol{V}_{\text{used}}$)
 8:     for all node $v$ of $\boldsymbol{V}$ do
 9:         if $T.isGraph$ then
10:             FOUNDMATCH($\boldsymbol{V}_{\text{used}} \cup \{v\}$)
11:         end if
12:         for all children $c$ of $t$ do
13:             MATCH($c$, $\boldsymbol{V}_{\text{used}} \cup \{v\}$)
14:         end for
15:     end for
16: end procedure
17: function MATCHINGVERTICES($t$, $\boldsymbol{V}_{\text{used}}$)
18:     if $\boldsymbol{V}_{used} = \emptyset$ then
19:         $\boldsymbol{V}_{\text{cand}} \leftarrow \boldsymbol{V}(G)$
20:     else
21:         $\boldsymbol{V}_{\text{conn}} \leftarrow \{v : v \in N(\boldsymbol{V}_{\text{used}})\})$
22:         $m \leftarrow m \in \boldsymbol{V}_{\text{conn}} : \forall v \in \boldsymbol{V}, |N(m)| \leq |N(v)|$
23:         $\boldsymbol{V}_{\text{cand}} \leftarrow \{v \in N(m)\text{: } v \notin \boldsymbol{V}_{\text{used}}\})$
24:     end if
25:     $\boldsymbol{Vertices} \leftarrow \emptyset$
26:     for all $v \in \boldsymbol{V}_{\text{cand}}$ do
27:         if $\forall i \in [1 \ldots |\boldsymbol{V}_{\text{used}}] : T.in[i] \leftarrow G_{\text{adj}}[\boldsymbol{V}_{\text{used}}[i]][v] \wedge T.out[i] \leftarrow G_{\text{adj}}[v][\boldsymbol{V}_{\text{used}}[i]]$ then
28:             $\boldsymbol{Vertices} \leftarrow \boldsymbol{Vertices} \cup \{v\}$
29:         end if
30:     end for
31:     return $\boldsymbol{Vertices}$
32: end function
```

used in isolation to perform subgraph enumeration, or it can be used in conjunction with other algorithms to perform motif discovery. It was developed to be used in biological networks that have multiple edge types, such as the physical, genetic, and signalling interactions between kinases and phosphatases in yeast, studied by Breitkreutz et al. (2010). ISMAGS can also identify subgraphs with directed edges, but it was not designed to be able to identify subgraphs with bi-directional edges. A workaround to this shortcoming is available, which is explained next.

Motif specifications

A concept called *motif specifications* is used in ISMAGS to describe a subgraph, instead of the traditional method of using adjacency matrices (this term is somewhat misleading, since this algorithm only counts and identifies *subgraph* instances, and not statistically significant *motifs*).

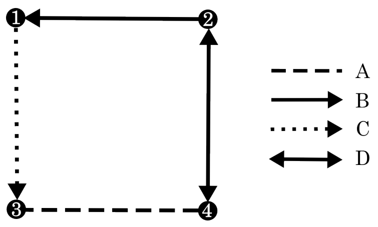
For example, subgraph node 1 in Figure 2.11(a) has an outgoing edge of type C, but subgraph node 3 has an outgoing edge of type c. This is because an edge is always referred to as outgoing from a node, even if the direction of the edge is actually towards the node. If the edge direction

Algorithm 6 Determining symmetry breaking conditions for a subgraph.

```
Input: Subgraph $s$
Output: symmetry breaking conditions $\boldsymbol{C}$
 1: function GTRIECONDITIONS(subgraph $s$)
 2:     conditions $\boldsymbol{C} \leftarrow \emptyset$
 3:     automorphisms $a \leftarrow$ SETAUTOMORPHISMS($s$)
 4:     while —$a$— $\geq 1$ do
 5:         $m \leftarrow$ minimum $v : \exists map \in a, map[v] \neq v$
 6:         for all $v \neq m : \exists map \in a, map[m] = v$ do
 7:             add $m \leq v$ to $c$
 8:         end for
 9:         $a \leftarrow \{map \in a : map[m] = m\}$
10:     end while
11:     return $\boldsymbol{C}$
12: end function
```

is towards the node in question, its outgoing edge is referred to as the lowercase letter of the edge type.

To handle subgraphs with bi-directional edges, these have to be defined as a new edge type, such as edge type D in Figure 2.11(a). Undirected and bi-directional edges are always referred to by using their uppercase letters. Therefore both subgraph node 3 and 4 have an outgoing edge of type A, and subgraph nodes 2 and 4 have an outgoing edge of type D.



(a) A subgraph with four edge types.

	1	2	3	4
1		b	C	0
2			0	D
3				A
4				

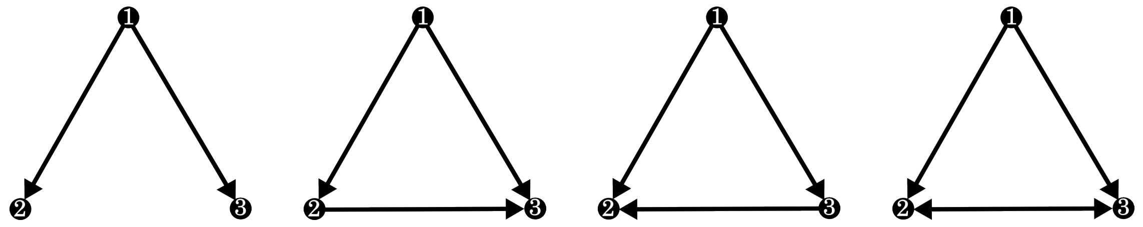
(b) Determining the motif specification.

Figure 2.11: Determining the motif specification for a 4-node subgraph with different edge types.

To determine the motif specification, an empty matrix with the same number of columns and rows as the subgraph size is created. Only the upper half of the matrix will be filled. Since there are no self-edges allowed, all positions $[i, j]$ where $i = j$ are empty. Note that the indices of the motif specification matrix are labelled according to the node labels, which start at 1. Position [1, 2] represents the edge type from node 1 to 2. It is an edge of type B, but the direction is towards node 1, therefore it is denoted with a lowercase b. Position [1, 3] is filled with a C, since there is an edge of type C from node 1 to node 3. There are no edges between nodes 1 and 4, nor between 2 and 3, therefore positions [1, 4] and [2, 3] are filled with zeros. Position [2, 4] is filled with a D and [3, 4] with an A. The motif specification is then determined by concatenating all edge types in the matrix, starting with all values in the first column, then all the values in the second column, and so forth. Thus the motif specification for the subgraph in Figure 2.11(a) is '*bC00DA*'. Bigger subgraphs will have longer motif specifications.

Note that the presence of a '0' in the motif specification actually denotes what Houbraken et al. (2013) refer to as a "don't care link". This means that there may or may not be an edge between these two nodes; its presence or absence is unimportant. This property of the algorithm was especially designed with biological networks in mind. This has implications for this dissertation, since a '0' edge must indicate the absence of an edge.

Suppose a 3-node subgraph has two edge types: A denotes a one-directional edge and X denotes a bi-directional edge. The motif specification $AA0$ should only represent the subgraph shown in Figure 2.12(a), which has two type A edges from node 1 to 2 and 3, and no edge between node 2 and 3. However, if a '0' in the motif specification represents a "don't care link", $AA0$ can also refer to the subgraphs shown in Figures 2.12(b)–2.12(d). Therefore the subgraph count for subgraph $AA0$ will be inflated, as it includes the subgraphs in Figures 2.12(b)–2.12(d). For purposes of this dissertation, the subgraphs with edges between node 2 and 3 will be removed from the results. This is done using Algorithm 14, which is explained in detail in Chapter 3.



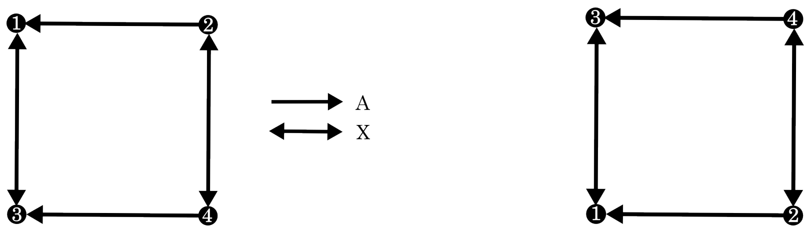
(a) $AA0$. (b) AAA. (c) AAa. (d) AAX.

Figure 2.12: ISMAGS considers Figures 2.12(b)–2.12(d) as $AA0$ subgraphs, but they are actually AAA, AAa, and AAX subgraphs.

Symmetry detection

Before ISMAGS determines subgraph counts, it identifies automorphisms of the subgraph. Automorphisms can be determined by looking for symmetric nodes in the subgraph, which have the same number and type of edge. Consider the following example explaining symmetry detection.

Subgraph 1 in Figure 2.13(a) has four nodes and two edge types. Node 1 and 3 both have an outgoing edge of type X and a. Similarly, node 2 and 4 have the same number and type of outgoing edges (X and A). Therefore node 1 and 3 are symmetric to each other, as are node 2 and 4. Since subgraph 2 in Figure 2.13(b) has the same node and edge relations, it is an automorphism of subgraph 1 in Figure 2.13(a). If ISMAGS finds subgraph 1 in a larger network, would be unnecessary for it to find subgraph 2, since these are automorphism of each other.



(a) Subgraph 1. (b) Subgraph 2.

Figure 2.13: Two automorphisms of the same subgraph.

The nodes of subgraph 1 and 2 are permutations of each other, and the subgraph nodes can be mapped to each other as shown in Figure 2.14(a). This mapping can be represented as a *partition*, shown in Figure 2.14(b). This notation maps nodes in the top row of the partition to nodes in the bottom row of the partition. Since nodes 1 and 3 and 2 and 4 are symmetric to each other, each of these pairs are placed in their own *cell*, denoted by |, as shown in Figure 2.14(c)

$$\begin{array}{c}1 \to 3\\ 2 \to 4\\ 3 \to 1\\ 4 \to 2\end{array}$$

(a) A string and one of its permutations.

$$\left\{\begin{array}{cccc}1 & 2 & 3 & 4\\ 3 & 4 & 1 & 2\end{array}\right\}$$

(b) Representing the string and its permutation in a partition.

$$\left\{\begin{array}{c|c|c|c}1 & 2 & 3 & 4\\ 3 & 4 & 1 & 2\end{array}\right\}$$

(c) Splitting the permutation into cells to indicate symmetric numbers.

Figure 2.14: A permutation represented with a partition.

Starting with the first number in the top row of the partition, 1 is symmetric to 3. But 3 (the third number in the top row of the partition) is symmetric to 1. Therefore the symmetry has come ‘full circle’, and a circle is exactly what is used to represent it. This is called an *orbit*, and is shown in Figure 2.15(a). This orbit maps 1 to 3, and 3 to 1, and is written as $\{13\}$. The second orbit, shown in Figure 2.15(b) maps 2 to 4, and 4 to 2. This is written as $\{24\}$. Combining these two orbits results in an *orbit partition* containing two *orbit partition cells*, $O = \{13\}\{24\}$.

(a) The first orbit.

(b) The second orbit.

Figure 2.15: Representing the symmetric numbers as orbits.

These partitions and orbits are used by Houbraken et al. (2013) to identify all symmetries of a subgraph. Whenever a permutation of subgraph nodes is found, and all nodes are symmetric to each other, it means that an automorphism is found. To explain symmetry detection and breaking, the same example used by Houbraken et al. (2013) shown in Figure 2.16, is used to explain these concepts. Symmetry breaking and detection occur simultaneously in ISMAGS, but for illustration purposes are explained separately.

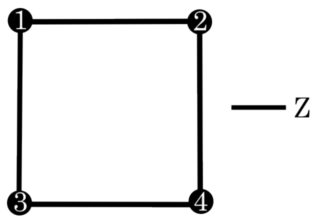


Figure 2.16: The subgraph for which symmetric nodes must be determined.

On inspection of this subgraph, it can be determined that all nodes are symmetric to each other, because they all have two outgoing edges of type Z. Algorithm 7 is used to analyse a subgraph for symmetric nodes, and it calls Algorithm 8 to determine which nodes are symmetric. This algorithm iteratively tries to find nodes in the top of the partition that are symmetrical to nodes in the bottom partition.

Line 8 in Algorithm 9 creates the initial orbit partition for the subgraph by grouping all nodes of the subgraph into one cell, resulting in the partition shown in Figure 2.17. This partition is called an *ordered partition pair* (or *OPP*), denoted by Π, containing the top part of the partition, π_t, and the bottom part of the partition, π_b.

Algorithm 7 Analysing a subgraph.

Input: Subgraph s
Output: List of constraints c

```
1:  function ANALYSESUBGRAPH(subgraph $s$)
2:      empty list of permutations $p$
3:      empty list of constraints $c$
4:      empty list of orbits $o$
5:      for all subgraph node $n$ in $s$ do
6:          add set $\{n\}$ to $o$
7:      end for
8:      Partition $\pi \leftarrow$ create initial partition of subgraph $s$
9:      OPP $opp \leftarrow$ create initial OPP from $\pi$
10:     PROCESSOPP$(opp, p, c, o)$
11:     return $c$
12: end function
```

$$\Pi = \begin{Bmatrix} \pi_t \\ \pi_b \end{Bmatrix} = \begin{Bmatrix} 1234 \\ 1234 \end{Bmatrix}$$

Figure 2.17: The initial orbit partition pair.

The ordered partition pair, and empty sets of permutations p, constraints c, and orbits o are passed to Algorithm 8. The algorithm starts in line 2 by checking whether all symmetric nodes have been identified. Since the ordered partition pair in Figure 2.17 does not contain four cells, all symmetric nodes have not been identified, and the algorithm continues in line 6. It identifies the node n_i in π_t with the smallest node ID, which is node 1 in this example. The cell containing n_i is assigned to the variable T, and the cell in the bottom partition corresponding to T, is assigned to B (lines 7–8). At this moment, both T and B are equal to $\{1234\}$. All nodes in B are ordered in ascending order in line 9.

Lines 10–14 iteratively maps nodes in T to all possible nodes in B. Starting with the node in B with the smallest node ID, 1 is *coupled* to 1. This creates a new cell in the partition, as shown in Figure 2.18(a).

$$\begin{Bmatrix} \pi_t \\ \pi_b \end{Bmatrix} = \left\{ \begin{array}{c|ccc} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{array} \right\}$$

(a) The first coupling operation results in the partition being split into two cells.

$$\begin{Bmatrix} \pi_t \\ \pi_b \end{Bmatrix} = \left\{ \begin{array}{c|cc|c} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{array} \right\}$$

(b) The partition now contains three cells after performing refinement.

Figure 2.18: Coupling node 1 in π_t to node 1 in π_b and performing refinement on the ordered partition pair.

Line 12 *refines* the ordered partition pair by ensuring that all nodes in the second cell in T (2, 3, 4) have incoming edges from the node in the first cell in T (1). Refinement is necessary to ensure that all edge relations and node degrees are preserved, as explained by Katebi et al. (2012). Nodes 2 and 3 have an edge from node 1, and these nodes are placed in the second cell, as shown in Figure 2.18(b). Since node 4 does not have an incoming edge from node 1, it is placed in its own cell.

The PROCESSOPP algorithm is then called recursively until all symmetric nodes have been identified.

Algorithm 8 Processing the ordered partition pair.

```
Input: Ordered partition pair $opp$, permutations $p$, constraints $c$, orbit $o$
 1: procedure PROCESSOPP(ordered partition pair $opp$, permutations $p$, constraints $c$, orbit $o$)
 2:     if all nodes are mapped then
 3:         add current mapping to $p$
 4:         update $o$
 5:     else
 6:         $n_i \leftarrow$ subgraph node with lowest ID among unmapped nodes
 7:         $T \leftarrow$ cell in top partition which contains $n_i$
 8:         $B \leftarrow$ cell in bottom partition corresponding to $T$
 9:         Sort $B$ by increasing ID: $[I_1, I_2, I_3, \ldots, I_B]$
10:         for all $j = 1 \ldots b$ do
11:             couple $n_i$ to $I_j$
12:             $opp_{\text{new}} \leftarrow$ refine $opp$
13:             PROCESSOPP($opp_{\text{new}}$, $p$, $c$, $o$)
14:         end for
15:         if $opp$ maps $n_k \leftarrow n_k \forall k \leq i$ then
16:             $C_i \leftarrow$ orbit of $n_i$
17:             for all $n_i \in C_i, i \neq t$ do
18:                 add $ID_i \leq ID_t$ to $c$
19:             end for
20:         end if
21:     end if
22: end procedure
```

Symmetry breaking

The set of permutations generated in the symmetry detection phase can be applied to a valid subgraph instance to identify another subgraph instance that is symmetric to it. To avoid exporting these symmetric instances, the symmetry of a subgraph instance must be broken, hence the term *symmetry breaking*. By breaking symmetries, ISMAGS returns the minimal set of subgraph instances, since a subgraph instance will only be exported once and its symmetric subgraphs will not be exported.

For subgraph '*XX00XX*' in Figure 2.16, many permutations of the subgraph nodes exist, collectively referred to as G. A *stabiliser* of node 1 is a permutation that maps node 1 to itself, for example, $P = (1)(2)(34)$. But there exists more permutations that maps node 1 to itself, and the collection of these stabilisers is called the *stabiliser chain* of node 1, written as $G_1 = \{(1)(2)(3)(4), (1)(23)(4), (1)(24)(3), (1)(234), (1)(2)(34)\}$.

To get the stabiliser chain for node 2, all permutations in G_1 that map node 2 to itself are extracted, therefore $G_2 = \{(1)(2)(3)(4), (1)(2)(34)\}$. Therefore G_2 is a subset of G_1 and contains permutations that map node 1 and 2 to itself. Again, the next stabiliser chain, G_3, is determined to be the permutation(s) in G_2 that maps node 3 to itself, therefore $G_3 = \{(1)(2)(3)(4)\}$.

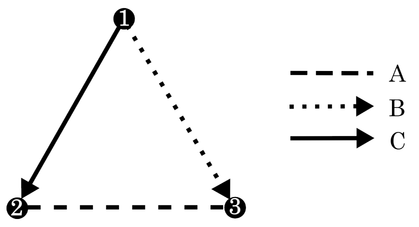
Based on the permutations in G_1, node 2 can be mapped to itself, to node 3, and to node 4. These are called the *coset representatives* of node 2, denoted by $C_2 = \{2, 3, 4\}$. The coset representatives are used to generate the symmetry breaking constraints for each node. The symmetry breaking constraint for subgraph node 2 uses the nodes in its coset representative, $C_2 = \{2, 3\}$. The constraint forces the ID of the graph node mapped to subgraph node 3 to be higher that the ID of the graph node mapped to subgraph node 2. Determining symmetry breaking constraints is done in Algorithm 8 in lines 15–20. The mapping of graph nodes to subgraph nodes is explained in detail in the next section.

Algorithm 9 Initialising ISMAGS.

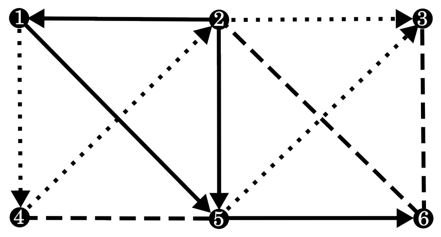
```
Input: Graph $g$, subgraph $s$
 1: constraints $c \leftarrow$ ANALYSESUBGRAPH($sg$)
 2: for all subgraph node $n$ in $s$ do
 3:     for all edge $e$ leaving/arriving in $n$ do
 4:         $t \leftarrow$ type of edge
 5:         $S \leftarrow$ list of graph nodes in $g$ that have edge type $t$
 6:         $\boldsymbol{S_n} \leftarrow S + \boldsymbol{S_n}$
 7:     end for
 8: end for
 9: subgraph node $n \leftarrow$ subgraph node with smallest start list $S$
10: node list $\boldsymbol{C_n} \leftarrow$ calculate candidate node list of $n$
11: MAPNODES($n, \boldsymbol{C_n}, c$)
```

Subgraph enumeration

The searching of subgraph instances is explained by means of the example found in Demeyer et al. (2013). Suppose the subgraph in Figure 2.19(a) must be searched for in the network shown in Figure 2.19(b). The subgraph contains three different edge types, A, B, and C.



(a) The subgraph to look for.



(b) The network in which the subgraph must be found.

Figure 2.19: The subgraph and network used to explain subgraph enumeration ISMAGS algorithm.

Finding subgraph instances is based on a *mapping* principle. Network node x can be mapped to subgraph node y if it has the same type of incoming and outgoing edges. Once all subgraph nodes have a network node mapped to it, a subgraph instance is complete and can be exported. The algorithm then continues to find the remaining subgraph instances.

ISMAGS is initialised in Algorithm 9. It starts by determining a set of constraints based on the symmetric properties of the subgraph to be found. This part of the algorithm sets it apart from its predecessor, the Index-Based Subgraph Matching Algorithm (ISMA) (Demeyer et al., 2013). Generating these constraints has been discussed in the previous sections.

In lines 2–8 it determines, for each subgraph node, all network nodes that have the same type of outgoing edges. Starting with subgraph node 1 and edge A, network nodes 1, 2, and 5 are identified as having A as an outgoing edge. Therefore these nodes are added to the start set of subgraph node 1, $\boldsymbol{S_1}$. Subgraph node 1 has another edge of type B, so again it is determined which network nodes have this as an outgoing edge. Therefore network nodes 1, 2, 4, and 5 are added to $\boldsymbol{S_1}$. This results in duplicate network nodes in $\boldsymbol{S_1}$, which will be extracted in the next part of the algorithm to start mapping network nodes to subgraph nodes. The algorithm continues with these steps until all start sets have been determined, resulting in the start sets shown in Table 2.1.

The subgraph node with the smallest start set, determined by its set cardinality, (in this example, $\boldsymbol{S_1}$) is selected (line 9). In line 10, the candidate list for subgraph node is calculated

Table 2.1: Network nodes with the same outgoing edges as subgraph nodes.

Subgraph node	Symbol	Start set	Set cardinality
1	$\boldsymbol{S_1}$	{1, 2, 5, 1, 2, 4, 5}	7
2	$\boldsymbol{S_2}$	{1, 5, 6, 2, 3, 4, 5, 6}	8
3	$\boldsymbol{S_3}$	{2, 3, 4, 2, 3, 4, 5, 6}	8

Algorithm 10 Mapping subgraph nodes to network nodes.

```
Input: Subgraph node $n$, candidate node list $\boldsymbol{C_n}$, constraints $c$
 1: procedure MAPNODES(subgraph node $n$, candidate node list $\boldsymbol{C_n}$, constraints $c$)
 2:     for all Node $u$ in $\boldsymbol{C_n}$ do
 3:         map $u$ to $n$
 4:         if subgraph instance complete then
 5:             export instance
 6:         else
 7:             for all Edge $e$ arriving/leaving $n$ do
 8:                 $t \leftarrow$ type of $e$
 9:                 $q \leftarrow$ origin/destination of $e$
10:                 $n_e \leftarrow$ neighbours of $n$ by type $t$
11:                 addNeighbourList($\boldsymbol{C_n}[q], n_e$)
12:             end for
13:             next subgraph node to map $n_n \leftarrow$ DETERMINENEXTSUBGRAPHNODE($\boldsymbol{C_n}, c$)
14:             MAPNODES($n_n, \boldsymbol{C_n}, c$)
15:         end if
16:         unmap $u$ from $n$
17:     end for
18: end procedure
```

to be all duplicate nodes in $\boldsymbol{S_1}$, resulting in $\boldsymbol{C_1} = \{1, 2, 5\}$. All network nodes in the candidate set of a subgraph node can be mapped to that subgraph node because it has the same type of outgoing edges. All candidate sets can be seen in Table 2.2.

Table 2.2: Candidate network nodes that can be mapped to subgraph nodes.

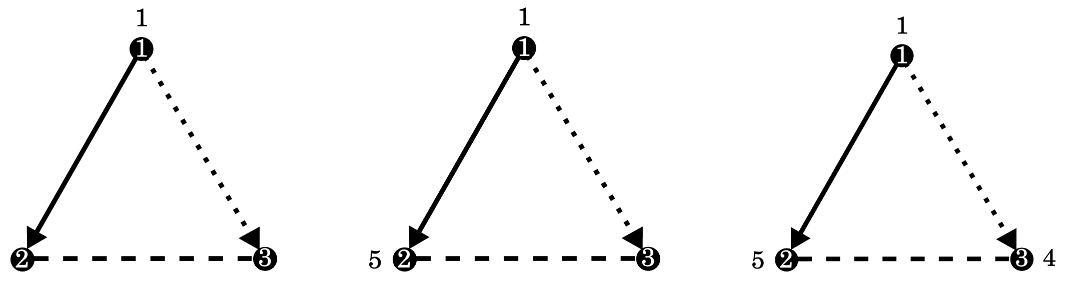
Subgraph node	Symbol	Candidate set	Set cardinality
1	$\boldsymbol{C_1}$	{1, 2, 5}	3
2	$\boldsymbol{C_2}$	{5, 6}	2
3	$\boldsymbol{C_3}$	{2, 3, 4}	3

In line 11, the subgraph node (1), candidate set ($\boldsymbol{C_1}$), and set of constraints for this subgraph node are passed to a function called MAPNODES. This function is detailed in Algorithm 10.

Algorithm 10 starts by comparing the candidate list to the constraints of that subgraph node. From this, a new candidate set is determined, which lists all network nodes that can be mapped to this subgraph node, while complying with the symmetry constraints of the subgraph node.

Suppose the new candidate set for subgraph node 1 remains unchanged after considering the constraints. The first network node in this set (1) is mapped to subgraph node 1 in line 3, resulting in the partially mapped subgraph instance shown in Figure 2.20(a). Subgraph nodes are labelled on nodes, with mapped network nodes shown on the outside of the subgraph.

All possible neighbours of network node 1 must then be determined (lines 7–12). The possible



(a) Mapping the first network node to a subgraph node.

(b) Mapping the second network node to a subgraph node.

(c) Mapping the third network node to a subgraph node. The subgraph instance is now complete and can be exported.

Figure 2.20: Mapping network nodes iteratively to subgraph nodes to determine the first subgraph instance.

neighbouring nodes are those that are connected to network node 1 with the same edge types that are found between subgraph node 1 and its neighbours (therefore, edges A and B).

For each edge connected to subgraph node 1, its edge type (A and B) is determined and is assigned to a new variable, t (line 8). The subgraph node connected to subgraph node 1 via this edge type is assigned to a variable q in line 9. In the network, all neighbours of subgraph node 1 connected by edge type t are searched for. So, for edge type A, the neighbouring set is $\boldsymbol{N_{1,A}} = \{5\}$. The neighbouring set for edge type B is $\boldsymbol{N_{1,B}} = \{4\}$.

All possible neighbours for the already mapped network node is determined, and the next node to map is determined from these node in line 13. Suppose this is chosen to be subgraph node 2. Therefore network node 5 is mapped to subgraph node 2 resulting in the incomplete subgraph instance shown in Figure 2.20(b). Lines 7 - 12 are repeated for subgraph node 2, and the completed subgraph instance shown in Figure 2.20(c) is identified as a result. The subgraph instance is exported as a list of network node IDs: `(1, 5, 4)`. The algorithm continues in this fashion until all subgraph instances have been identified.

2.4 Conclusion

Many factors have to be considered when performing motif discovery, such as how to generate random networks, and which motif discovery algorithm to use. The latter of these considerations is discussed in the next chapter by comparing the performance of G-Tries and ISMAGS on a number of small complex networks.

Chapter 3

Identifying a suitable motif discovery algorithm

Since computational complexity is an issue in motif discovery algorithms, the two chosen motif discovery algorithms are tested on networks that are smaller than the proposed freight complex network. The Barabási-Albert algorithm by Barabási and Albert (1999) is used to generate sample random networks that follow a power law distribution. The sample random networks are generated in R (R Development Core Team, 2008) using the `barabasi.game` function of the *igraph* package, which was developed by Csardi and Nepusz (2006).

Four network sizes, relative to the size of the proposed freight complex network, are used. The sizes range from 15% to 60% relative to the size of the freight network, which has 2,594 nodes. Therefore the size of the first network is $15\% \times 2,594 = 390$; the size of the second network is $30\% \times 2,594 = 779$; the size of the third network is $45\% \times 2,594 = 1,168$; and the size of the last network is $60\% \times 2,594 = 1,557$. For each network size, 10 random networks are generated, resulting in 40 sample random networks.

For each network, each algorithm, and each subgraph size, a distribution of memory, run time, and the number of subgraphs identified are obtained. The experiments are performed on a Dell PowerEdge R910 with 4x Intel Xeon E7540 Processors (2.00 GHz) and 128 GB Memory ($8 \times$ 16GB) running Linux Ubuntu version 14.04.3 LTS (Long term support) operating system. All results are shown as a panel plot, each panel representing the results for the subgraphs of that particular size. The first panel shows the results when the algorithms searched for 3-node subgraphs; the second panel shows the results when the algorithms searched for 4-node subgraphs; and the third panel shows the results when the algorithms searched for subgraphs of size 5. In this chapter, the same y-scale is used in each panel to illustrate how the distributions increase as the subgraph size increases. Appendix B contains panel plots on separate y-axes.

3.1 Quantitative comparison of ISMAGS and G-Tries

ISMAGS and G-Tries are compared quantitatively according to run time, memory usage, and the number of subgraph instances identified. In these experiments the focus is only on subgraph enumeration and not on motif discovery (determining which subgraphs are statistically significant). Therefore in this chapter only *subgraph* instances are referred to and not *motif* instances. Before the experiments can be performed, some pre-processing must be done.

3.1.1 Pre-processing for G-Tries

To run the G-Tries algorithm, the executable *gtrieScanner* application is used, the source files of which are freely available online (Ribeiro, 2015). It was written in *C++* for Linux operating systems, therefore requires a *C++* compiler, such as the *GNU Compiler Collection* (*gcc*) (Free

Software Foundation, 2015), as well as the *GNU make* utility (Free Software Foundation, 2015), which generates executable applications from binary files. After downloading and unzipping the source files, running `make` compiles the executable files.

The *gtrieScanner* is then able to perform various functions, such as generating g-tries from adjacency matrices, performing subgraph enumeration, and determining statistically significant motifs. For this experiment, three g-tries are required, each representing all directed subgraphs of size 3, 4, and 5. The subgraph lists for directed 3-, 4-, and 5-node subgraphs, found on the *gtrieScanner* website (Ribeiro, 2015), are used to generate the respective g-tries for these experiments.

3.1.2 Pre-processing for ISMAGS

Before performing the experiments using ISMAGS, two steps are required. First the motif specifications have to be generated for all directed 3-, 4-, and 5-node subgraphs. Secondly, the edge lists must be split into one- and bi-directional edge lists.

Generating motif specifications requires defining the edge types and converting adjacency matrices to motif specifications. One-directional edges are denoted by an A and a (when edge type A is incoming to the node), and bi-directional edges are denoted by an X. When no edge exists between two nodes, it is denoted by a '0'. Using these newly defined edge type definitions, Algorithm 11 is implemented in *Java* to generate the motif specifications for all directed 3-, 4-, and 5-node subgraphs.

Algorithm 11 Generating motif specifications.

```
Input: Adjacency matrix $a$
Output: Motif specification $s$
 1: function GETMOTIFSPECIFICATION(adjacency matrix $a$)
 2:     $m$, $n \leftarrow$ number of nodes in $a$
 3:     Create empty matrix $[n, m] \leftarrow$ motif specification matrix $m$
 4:     for all positions $[i, j]$ in $a$ where $i = [0 : n)$ and $j = [i : m)$ do
 5:         value $v \leftarrow$ value in position $[i, j]$
 6:         inverse value $v' \leftarrow$ value in inverse position $[j, i]$
 7:         if $v = 0 \wedge v' = 0$ then
 8:             place 0 in position $[i, j]$ in $m$
 9:         else if $v = 1 \wedge v' = 0$ then
10:             place $A$ in position $[i, j]$ in $m$
11:         else if $v = 0 \wedge v' = 1$ then
12:             place $a$ in position $[i, j]$ in $m$
13:         else if $v = 1 \wedge v' = 1$ then
14:             place $X$ in position $[i, j]$ in $m$
15:         end if
16:     end for
17:     $s \leftarrow$ concatenate the values in all columns in $m$
18:     return $s$
19: end function
```

The algorithm starts in line 3 by creating an empty motif specification matrix which is the same size as the adjacency matrix of the subgraph. The algorithm then iterates through all positions in the upper half of the adjacency matrix (lines 4–16). For each position $[i, j]$ where $i = (0, n]$ and $j = (i, m]$, it determines the value in this position (line 5), as well as the value in the inverse position, namely $[j, i]$ (line 6). Four possibilities exist. If both positions $[i, j]$ and $[j, i]$ contain a '0', no edge exists between nodes i and j, therefore position $[i, j]$ in the motif specification matrix is filled with a '0' (line 8). If position $[i, j]$ is filled with a '1', while position

$[j, i]$ is filled with a '0', an edge A exists from node i to node j, and the motif specification matrix is filled with an A in position $[i, j]$ in line 10. If position $[i, j]$ is filled with a '0' and $[j, i]$ with a '1', an edge of type A exists from node j to node i. Position $[i, j]$ in the motif specification matrix must therefore be filled with an a (line 12), since an edge of type A is incoming to node i. If positions $[i, j]$ and $[j, i]$ both contain a '1', $[i, j]$ in the motif specification matrix is filled with an X (line 14). When all values in the upper half of the adjacency matrix have been iterated over, the algorithm concatenates the columns of the motif specification matrix in line 17 to create the motif specification of the subgraph.

The input of ISMAGS requires separate edge lists, one for each type of edge. Algorithm 12 is implemented in *Java* and run on all 40 sample random networks to split the edges into one-directional and bi-directional edges.

Algorithm 12 Splitting edges into one- and bi-directional edges.

```
Input: Edge list $\boldsymbol{E}$
Output: One-directional edge list $\boldsymbol{O}$, bi-directional edge list $\boldsymbol{B}$
 1: function splitEdges(edge list $\boldsymbol{E}$)
 2:     create empty one-directional edge list $\boldsymbol{O}$
 3:     create empty bi-directional edge list $\boldsymbol{B}$
 4:     copy all edges in $\boldsymbol{E}$ to $\boldsymbol{O}$
 5:     for all edges $e \in \boldsymbol{E}$ do
 6:         origin $o \leftarrow$ origin node of $e$
 7:         destination $d \leftarrow$ destination node of $e$
 8:         inverse edge $i \leftarrow (d, o)$
 9:         for all edges $e'$ in $\boldsymbol{E} \neq e$ do
10:             origin $o' \leftarrow$ origin node of $e'$
11:             destination $d' \leftarrow$ destination node of $e'$
12:             if $o' = d \wedge d' = o$ then
13:                 bi-directional edge found
14:                 remove $e$ from $\boldsymbol{O}$
15:                 remove $e'$ from $\boldsymbol{O}$
16:                 add $e$ to $\boldsymbol{B}$
17:             end if
18:         end for
19:     end for
20:     return $\boldsymbol{O}$ and $\boldsymbol{B}$
21: end function
```

The algorithm creates empty one-and bi-directional edge lists, $\boldsymbol{O}$ and $\boldsymbol{B}$, in lines 2–3. The network's edge list is copied to the one-directional edge list in line 4. All edges in $\boldsymbol{E}$ are iterated over in lines 5–19. The inverse i of edge e is determined by switching the origin and destination nodes in line 8. All edges (except e) in $\boldsymbol{E}$ are again iterated over (lines 9–18) to determine if the inverse edge exists. If it does, a bi-directional edge has been found (line 13). Edge e and its inverse e' are removed from one-directional edge list $\boldsymbol{O}$, and e is added to bi-directional edge list $\boldsymbol{B}$. Note that the inverse edge e' can also be added to $\boldsymbol{B}$ instead of e, since a bi-directional edge from node x to node y is equivalent to a bi-directional edge from node y to node x.

Algorithm 13 is implemented in *Java* to iteratively call ISMAGS for all 40 sample random networks and all motif specifications generated by Algorithm 11. The input to the algorithm is three lists. $\boldsymbol{O}$ is a list of the one-directional edges of a sample random network, $\boldsymbol{B}$ is a list of bi-directional edges in that sample random network, and $\boldsymbol{S}$ is a list of all motif specifications to look for. The algorithm calls ISMAGS in line 4 for each sample random network (line 2) and each motif specification (line 3). ISMAGS then identifies all instances of this motif specification s in this sample random network, and returns the list $\boldsymbol{I}$ of subgraph instances in line 21.

Algorithm 13 Iteratively calling ISMAGS.

```
Input: One-directional edge list $\boldsymbol{O}$, bi-directional edge list $\boldsymbol{B}$, list of motif specifications $\boldsymbol{S}$
Output: List of subgraph instances $\boldsymbol{I}$
1: procedure callISMAGS(list $\boldsymbol{O}$, list $\boldsymbol{B}$, list $\boldsymbol{S}$)
2:     for all one-directional edge list $o \in \boldsymbol{O} \wedge$ bi-directional edge list $b \in \boldsymbol{B}$ do
3:         for all motif specification $s \in \boldsymbol{S}$ do
4:             function ISMAGS($o$, $b$, $s$)
5:                 return $\boldsymbol{I}$, list of all subgraph instances of type $s$
6:             end function
7:         end for
8:     end for
9: end procedure
```

It should be noted that, for each iteration (motif specification), ISMAGS reads the network into memory. This results in the network being read multiple times, whereas G-Tries only reads the network in once. This bias is acknowledged, but it is argued that the time it takes to read a network into memory is negligible when compared to the time it takes to search for the subgraphs.

3.1.3 Comparing by memory usage

To capture data on memory usage, a tool called *Valgrind* (Nethercote and Seward, 2007) is used. *Valgrind* is used to debug and profile programs in Linux operating systems, and was released under the GNU GPL license.

The *gtrieScanner* and ISMAGS are run with the prefix `valgrind --tool=massif`, which calls the `massif` method of Valgrind. After the run is complete, *Valgrind* outputs a file detailing the memory usage of the program at different intervals. It also specifies the interval in which the peak memory usage was obtained, and these peak values are reported on in this dissertation. *Valgrind* uses bytes as the unit for measuring memory, but the results are converted to kilobytes using the conversion of 1 kilobyte = 1,024 bytes.

It is expected that G-Tries will have a higher memory usage that ISMAGS, because Ribeiro and Silva (2010) mention that G-Tries sacrifices memory to significantly decrease run time. This sacrifice in memory is not evident when comparing the distribution of memory usage of G-Tries and ISMAGS. Figure 3.1 shows that G-Tries performs better than ISMAGS in terms of memory usage. G-Tries uses an increasing amount of memory as the size of network increases, while the memory usage of ISMAGS remains relatively similar for differing sizes of subgraphs and networks. ISMAGS uses, on average, 150 kilobytes to search for subgraph instances. While this is significantly higher than the memory usage of G-Tries, it is negligible if measured against hardware standards that are currently available.

3.1.4 Comparing by run time

The run time is taken as the total run time, including reading in the network, performing the subgraph search, and writing the output file(s). Any pre- and post-processing performed is not included in these results. Since using *Valgrind* increases the run time of the program, all experiments were rerun to obtain run times. The *gtrieScanner* automatically outputs the time elapsed, and ISMAGS is programmed to return the total run time. These values are captured for each algorithm, each subgraph, and each network, and the distribution of run time (in seconds) is shown in Figure 3.2.

It is expected that G-Tries will have a shorter run time than ISMAGS, since the former is implemented in *C++* and the latter in *Java*. While run time greatly depends on the program-

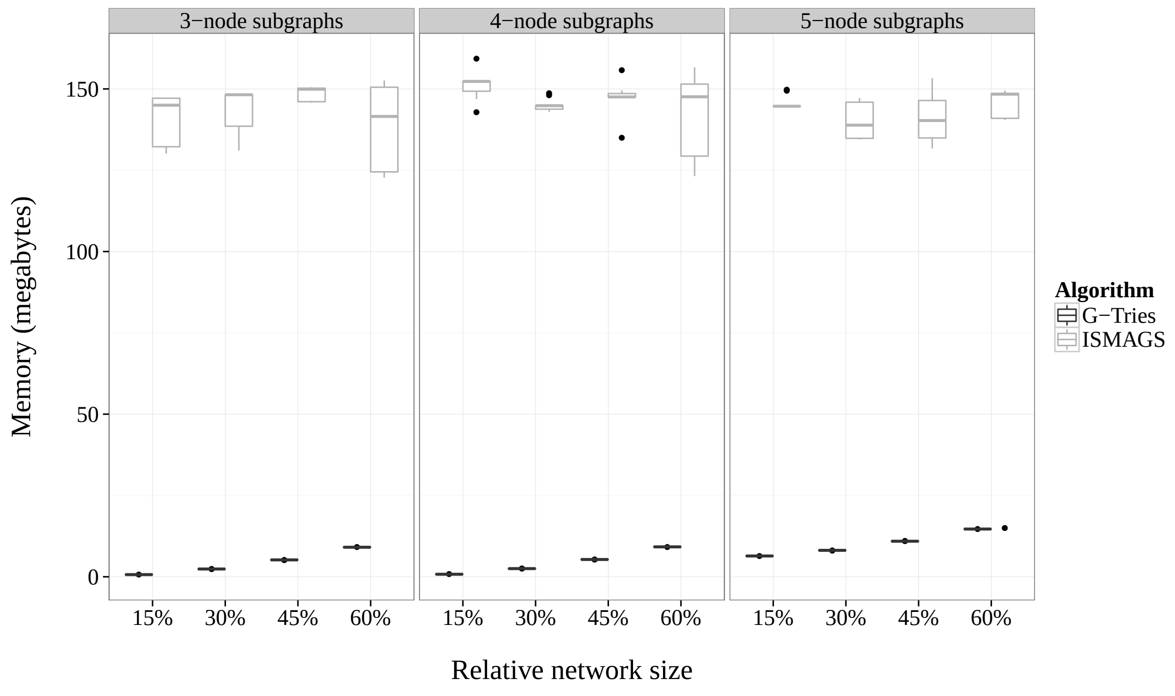


Figure 3.1: The memory usage (in kilobytes) of ISMAGS and G-Tries while searching for subgraphs of increasing size.

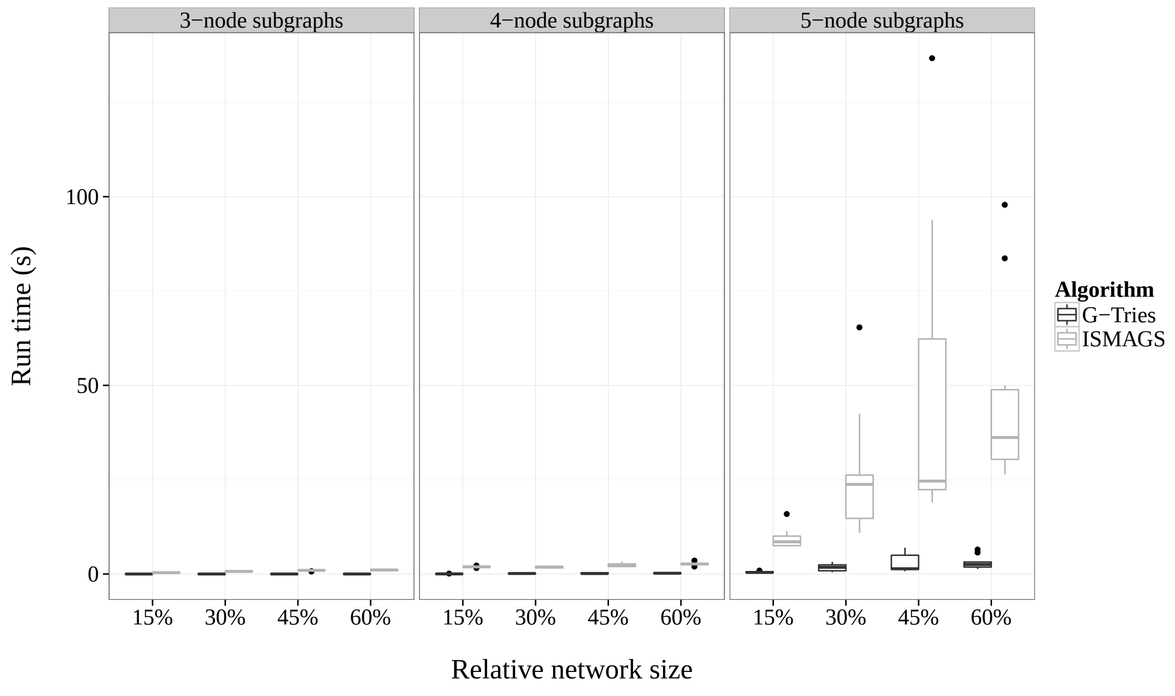


Figure 3.2: Comparing the run time (s) of ISMAGS and G-Tries as the subgraph size increases from 3 to 5 nodes.

mer's ability to optimise the code, there is a general understanding in online communities that *C++* programs run faster than *Java* programs. This may be attributed to the fact that *C++* is written in objective code, which can be directly read by the computer. *Java*, on the other hand, is written in byte code, which needs to be translated by a Java Virtual Machine into objective code for the computer to run it.

As expected, the run time increases as the network size increases, but also as the size of the subgraph increases (since there exists more 5-node subgraphs than 3-node subgraphs). There is greater variability in the run time when looking for subgraphs of size 5, and ISMAGS generally has a slightly longer run time than G-Tries. While ISMAGS performs worse than G-Tries, its longest run time is only 140 seconds when identifying 5-node subgraphs in a network. In comparing ISMAGS to a number of algorithms in terms of run time, Houbraken et al. (2013) also find that G-Tries has, on average, a shorter run time than ISMAGS.

3.1.5 Comparing by the number of subgraphs instances identified

This analysis compares the number of subgraph instances identified by each algorithm. As the size of the subgraph increases, so does the number of subgraphs, because by increasing the number of nodes in a subgraph, the possible edge configurations are also increased. Kashtan et al. (2004) note that there exists 13 directed 3-node subgraphs, 199 directed 4-node subgraphs, and 9364 directed 5-node subgraphs. This analysis is important since the maximum number of motif instances must be identified in the freight complex network. Analyses will be performed on the firms in the statistically significant motifs, and if some motif instances are not identified by the chosen algorithm, the results may not be realistic.

Recall from Chapter 2 that ISMAGS considers a '0' in a motif specification as a "don't care link", which means that there may be an edge in the '0' position. For purposes of this dissertation, a '0' in the motif specification must indicate that the edge does not exist. Therefore Algorithm 14 is implemented in *Java* to remove subgraph instances that contain an edge where there should not be one. It should be noted that the run time of Algorithm 14 is not included in the run times reported on in Section 3.1.4, since any pre- and post-processing performed does not form part of the quantitative results of these experiments.

Suppose *X0X* is identified as a motif specification to filter (line 2). Line 5 identifies the '0' in the motif specification representing the edge between subgraph node 1 and 3 as a "don't care link": there may or may not be an edge between these two nodes. Take, for example, the *X0X* subgraph instance (`23`, `5`, `10`). Network node `23` is mapped to subgraph node 1, network node `5` is mapped to subgraph node 2, and network node `10` is mapped to subgraph node 3 (lines 11–12). Therefore a check must be performed to ensure that there does not exist an edge between network nodes `23` and `10` (line 14). The edges of the network are iterated over in lines 13–18 to determine if any edge (one- or bi-directional) exists between these two nodes. If the edge exists, it is removed from the list of motif instances in line 17, and the resulting list of motif instances is returned in line 21.

Figure 3.3 shows that, for all subgraph sizes, ISMAGS identifies more instances than G-Tries. One would expect that the same number of subgraph instances would be identified by both algorithms, and the fact that they differ either indicates that ISMAGS identifies subgraph instances that actually do not exist, or that G-Tries does not identify all possible subgraph instances.

To validate the number of subgraph instances identified, a third motif discovery algorithm is used. *Fanmod* is a software tool with a graphical user interface for fast network motif detection (Wernicke and Rasche, 2006). It is used to identify all 3- and 4-node subgraphs in all 40 sample random networks. It is unable to identify the 5-node subgraphs in the sample random randoms since the amount of memory the program uses increases exponentially and causes the program to freeze. Figure 3.4 shows the number of subgraph instances identified by all three algorithms (G-Tries, ISMAGS, and *Fanmod*) for subgraphs of size 3 (Figure 3.4(a)) and 4 (Figure 3.4(b)). The figures illustrate that ISMAGS and *Fanmod* identifies exactly the same number of 3-node subgraph instances, and that of 4-node subgraph instances varies slightly. Therefore in both cases, G-Tries does not identify all the subgraph instances present in the sample random networks.

Algorithm 14 Filtering motif instances with '0' edges.

```
Input: Lists of motif instances $\boldsymbol{I}$, list of motif specifications $\boldsymbol{S}$, network edge list $\boldsymbol{E}$
Output: List of filtered motif instances $\boldsymbol{L}$
1: function FILTERMOTIFINSTANCES(list $\boldsymbol{I}$, list $\boldsymbol{S}$, list $\boldsymbol{E}$)
2:     zeroMotifs $z \leftarrow$ motif specifications in $\boldsymbol{S}$ that contain a '0'
3:     for all motif specification in $z$ do
4:         $m \leftarrow$ this motif specification
5:         edge $e \leftarrow$ '0' edge in $m$
6:         node $o \leftarrow$ origin node of $e$
7:         node $d \leftarrow$ destination node of $e$
8:         $\boldsymbol{L} \leftarrow$ list of all $m$ type motif instances in $\boldsymbol{L}$
9:         for all motif instances $i \in L$ do
10:            motif instance $i \leftarrow$ this motif instance
11:            origin network node $o_N \leftarrow$ node $o$ of $i$
12:            destination network node $d_N \leftarrow$ node $d$ of $i$
13:            get edges $(o, d)$ and $(d, o)$ from $\boldsymbol{E}$
14:            if edges $(o, d)$ or $(d, o)$ does not exist then
15:                continue
16:            else if edge $(o, d)$ and/or $(d, o)$ exist then
17:                remove $i$ from $\boldsymbol{L}$
18:            end if
19:        end for
20:    end for
21:    return $\boldsymbol{L}$
22: end function
```

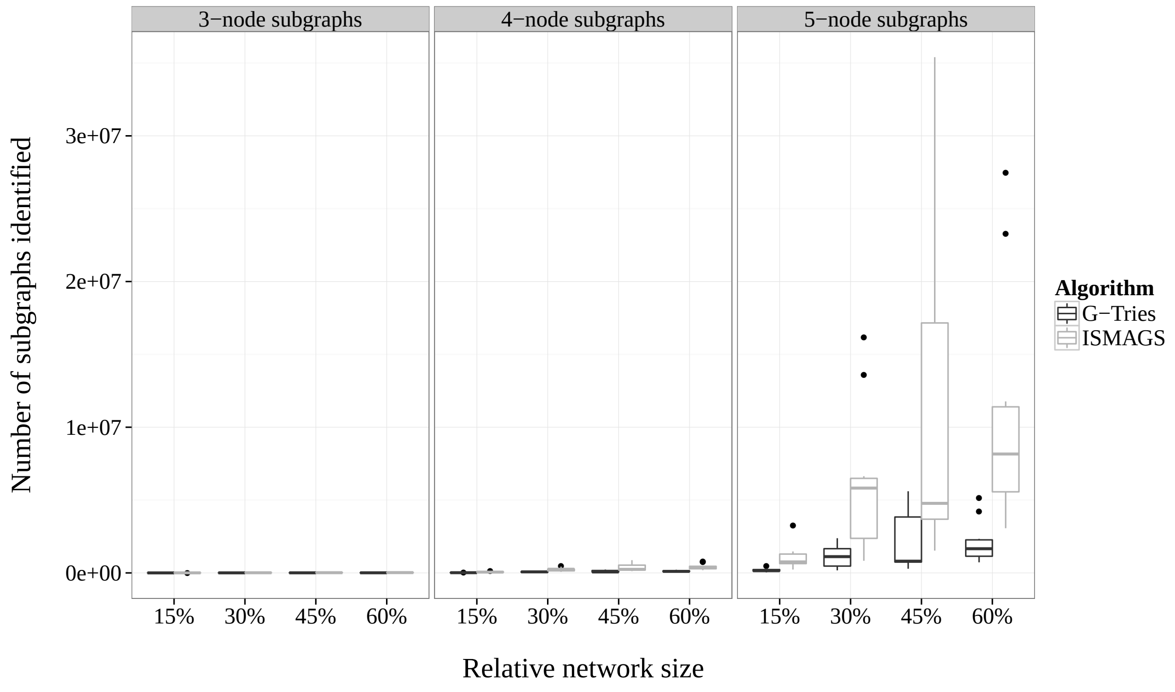
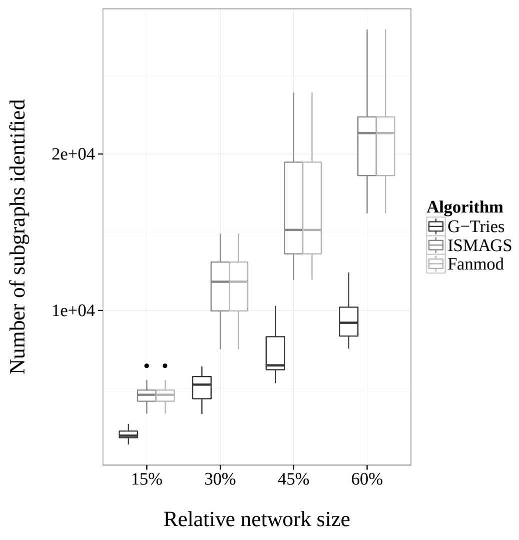
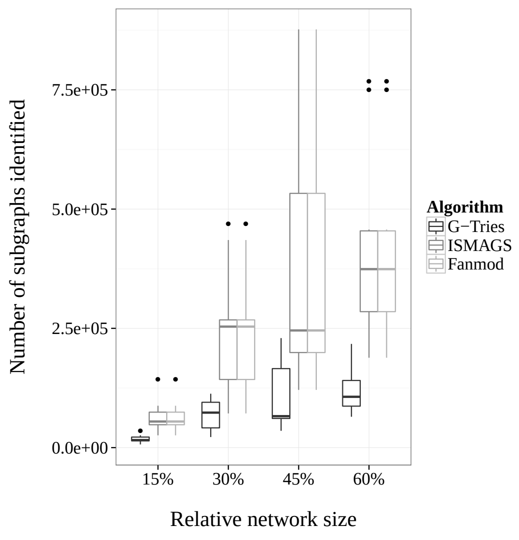


Figure 3.3: The number of subgraph instances identified by ISMAGS and G-Tries.

One possible explanation is that G-Tries does not generate g-tries that represent all directed subgraphs. The g-tries used in the experiments are generated from subgraph lists found on the



(a) The number of 3-node subgraph instances identified.



(b) The number of 4-node subgraph instances identified.

Figure 3.4: Using a third algorithm (*Fanmod*) to validate the number of subgraph instances identified by ISMAGS and G-Tries.

gtrieScanner website (Ribeiro, 2015). The subgraph lists contain exactly the correct number of adjacency matrices for 3-, 4-, and 5-node subgraphs (13, 199, and 9364 respectively). Therefore the problem does not lie with the subgraph lists. The g-trie data files generated by the *gtrieScanner* are not in a human readable format (it contains lists of symbols and numbers), and there is no explanation of what these symbols and numbers in the g-trie data files mean. Therefore one of two conclusions can be made about G-Tries. The first is that the g-tries generated by the *gtrieScanner* does not accurately represent all subgraphs given as input when generating the g-tries. The second is that, even though the g-tries correctly represent all subgraph instances to find, the algorithm is unable to identify all of the subgraph instances.

3.2 Qualitative comparison of ISMAGS and G-Tries

While the quantitative comparisons provide insight into the performance of the algorithms, the qualitative comparisons are deemed more important. This study will be used as a building block for future work, and therefore extendability of the motif discovery algorithm is extremely important.

G-Tries is able to perform motif discovery by performing subgraph enumeration on the original network, generating the specified number of random networks, enumerating subgraphs in the random network, and identifying statistically significant motifs. It is not limited to subgraph enumeration, as is the case with ISMAGS. However, as mentioned by Houbraken et al. (2013), it is possible to integrate your own code with ISMAGS to discover all statistically significant motifs in a network.

While Ribeiro and Silva (2010) state that node labels can start at 0, an error occurs when this algorithm is run on a network that has node labels starting at 0. The freight complex network has node labels that are labelled from 0, therefore if G-Tries was to be used on this large-scale network, all node labels would have to be incremented by one to prevent errors. Before analysing all subgraph instances identified, the node labels would have to be decremented by one to ensure that the analyses are performed on the correct nodes. ISMAGS is able to read networks with nodes that start at 0.

G-Tries writes all subgraph instances to one output file. Each line in the file contains the adjacency matrix of a subgraph, followed by the nodes that occur in that subgraph instance. Using this output file requires that it be split into separate files for each unique group of subgraph instances.

While ISMAGS only looks for one subgraph at a time (specified by the user), it can be programmed to iteratively be called for all subgraphs that the user is interested in identifying. For every subgraph looked for, ISMAGS writes all subgraph instances found to separate output files. The ISMAGS output, however, indicates the edge types in each subgraph instance. Since edge types are not of particular interest in this dissertation, these letters have to be removed from the output files before analysing the instances.

3.3 Conclusion

The quantitative comparisons provide insight into the relative performance of the two algorithms, and G-Tries performs better than ISMAGS in terms of memory and run time. Since the hardware available for this dissertation is not a constraint, the performance of the algorithms does not have to be the deciding factor. Rather, the qualitative comparisons are used to make a decision about which algorithm to use to identify motifs in the freight complex network.

This chapter illustrates that ISMAGS can easily be integrated with custom written *Java* classes, since three different algorithms (11, 12 and 14) are implemented to ensure ISMAGS is customised to suit the needs of these experiments. Therefore it is decided to use ISMAGS to identify statistically significant motifs in the freight complex network, which is built in the next chapter.

Chapter 4

Building the complex network

To build a complex network from raw GPS data requires a number of steps. Activity chains are extracted from raw GPS data traces, which are used to identify frequently visited firm locations. Activities are associated with firms where they were performed, and if two consecutive activities in the activity chains were performed at an identified firm, the freight trip is included in the complex network as two nodes (firms), with a directed edge (direct freight trip) between them. These steps are based on the methodology of Joubert and Axhausen (2011), Joubert and Axhausen (2013) and Joubert and Meintjes (2015a).

4.1 Extracting activity chains

The GPS traces of over 40,000 freight vehicles travelling in South Africa over a six-month period were provided by *Digicore Fleet Management*. From these data, Joubert and Axhausen (2011) extract activity chains, which describe the location, duration, and sequence of activities performed by freight vehicles. Ignition off- and on-signals are used to indicate the start and end of activities performed by freight vehicles. If a freight vehicle's engine is switched off for more than five hours, it is assumed to be a depot-like activity. These are called *major* activities and represent the start and end of the activity chain. *Minor* activities are those that last less than five hours, and make up the rest of the activity chain.

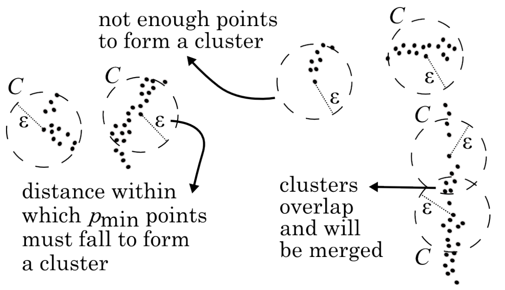
4.2 Identifying the firms where activities were performed

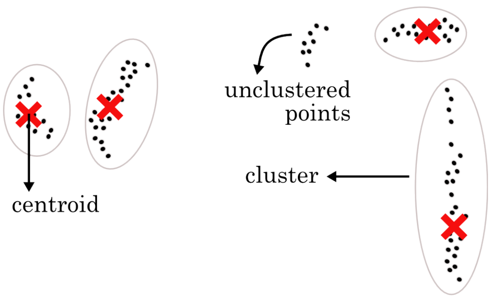
GPS data are noisy and can be inaccurate: when two freight vehicles stop at the same firm to perform an activity, each will have a unique GPS location. A high density of activities will therefore occur at or near firms where activities are frequently performed. The focus is on the more frequented firm locations, because these have a greater logistics impact (Joubert and Axhausen, 2013).

It is argued that a *firm* does not have to be confined to the four walls of a business: businesses can be grouped into firms based on function or location. For example, all freight activities at a small shopping centre can be grouped into one firm. Individually, the small shops at such a shopping centre do not have a large logistics impact because they receive/make deliveries once every few weeks. However, by aggregating the freight activities of all these shops, the shopping centre itself becomes an interesting firm. This is an example of grouping businesses into a firm by location. Facilities may also form based on function: a large retailer may receive deliveries from large suppliers through their back door, while smaller suppliers may deliver goods through the front door. These two types of deliveries will often differ from each other, representing suppliers of different sizes and deliveries of different volumes. Therefore the front- and back-door can be modelled as two separate firms. Another example is a manufacturing plant where there are dedicated shipping and receiving bays. These are often separate in manufacturing plants, and high densities of shipping and receiving activities will be performed at the respective bays. Even though all shipping and receiving activities are performed at the same plant, the bays can be modelled as separate firms, each with their own function.

Joubert and Axhausen (2013) use a density-based clustering algorithm, namely the *DJ-Cluster* approach, developed by Zhou et al. (2004), which groups the GPS points (activities) into clusters. Two input clustering parameters, which determine the minimum number of activities that need to be performed within a given radius from each other to be considered a cluster, control which activities are included in clusters. Different combinations of clustering parameters can result in clusters of different shapes and sizes. To understand the *DJ-Cluster* approach, consider the following explanation along with the clustering example in Figure 4.1.

(a) Activities are superimposed on an aerial image (source: South African National Geospatial Institute (NGI) imagery available on *OpenStreetMap* at location 26.114435 S, 27.894554 E).


not enough points to form a cluster
C
ε
distance within which $p_{\min}$ points must fall to form a cluster
clusters overlap and will be merged
(b) Of the six neighbourhoods that are being calculated, five are identified as clusters, each denoted by C.



(c) Four clusters are identified, and each cluster's centroid represents the location of the firm.

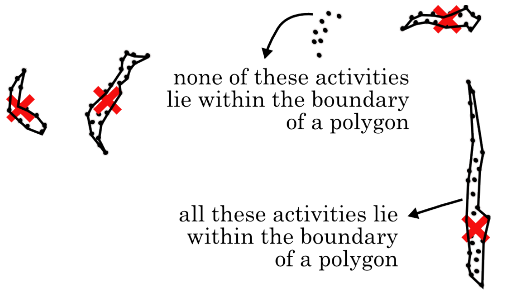

none of these activities lie within the boundary of a polygon
all these activities lie within the boundary of a polygon
(d) A concave hull is generated for each cluster. The polygon encloses all the points that form part of the cluster.

Figure 4.1: The clustering of activities using the *DJ-Cluster* approach proposed by Zhou et al. (2004).

Figure 4.1(a) shows freight activities that have been superimposed on an aerial image of five firms where freight activities are performed. The *DJ-Cluster* approach is applied to six activities in Figure 4.1(b). The neighbourhood, N, of each activity, p, is calculated as all the activities that lie within a specified radius, ϵ, from activity p. If an activity's neighbourhood contains a minimum number of points, $p_{\min}$, it is considered to be a cluster, C. The clustering parameters in this example are chosen to be $\epsilon = 4.8$ units and $p_{\min} = 10$ points. Changing the clustering parameters results in different clusters being identified.

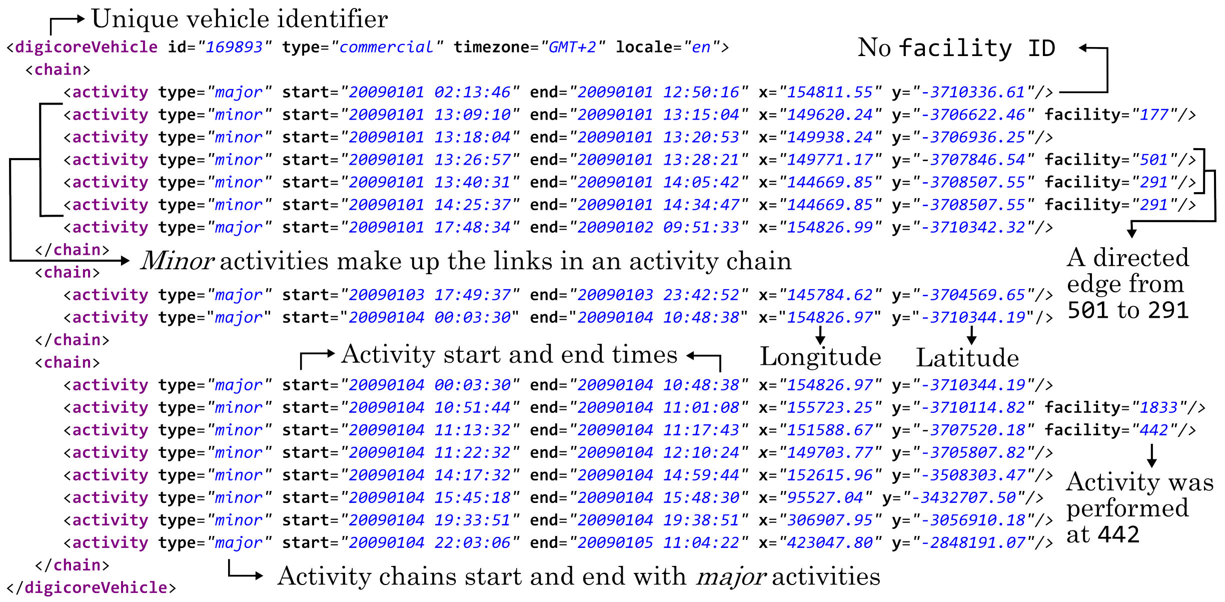
If no cluster exists around an activity, the activity and its neighbourhood are ignored because they do not create a big enough logistics impact (Joubert and Axhausen, 2013). If two clusters contain one or more of the same activities, the clusters are merged into a new cluster.

Each cluster represents an interesting firm, and the activities that form part of the cluster were performed at that firm. The centroids of all identified clusters are calculated in Figure 4.1(c), which is assumed to be the location of the firm. A unique identifier, the `facility ID`, is assigned to each centroid and keeps track of all interesting firms.

4.3 Adapting activity locations

Millions of points are considered during the clustering step, and it is a computational burden to keep record of every single point that forms part of a cluster. Therefore Joubert and Meintjes (2015a) generate a concave hull polygon for each cluster, as shown in Figure 4.1(d). The implementation of the concave hull algorithm forms part of a mini-dissertation for an undergraduate study (Meintjes, 2013). It is easy to determine whether an activity is performed at an interesting firm: simply verify whether it is covered by a concave hull. If it is, the `facility ID` of the concave hull is assigned to the activity, and the activity's location is adapted to that of the centroid.

Executing this step results in adapted activity chains, where some of the activities have new locations and `facility IDs` allocated to them. Figure 4.2 shows three of the adapted activity chains of `digicoreVehicle 169893`.


Unique vehicle identifier
No facility ID
Minor activities make up the links in an activity chain
A directed edge from 501 to 291
Activity start and end times
Longitude
Latitude
Activity was performed at 442
Activity chains start and end with *major* activities

```
<digicoreVehicle id="169893" type="commercial" timezone="GMT+2" locale="en">
  <chain>
      <activity type="major" start="20090101 02:13:46" end="20090101 12:50:16" x="154811.55" y="-3710336.61"/>
      <activity type="minor" start="20090101 13:09:10" end="20090101 13:15:04" x="149620.24" y="-3706622.46" facility="177"/>
      <activity type="minor" start="20090101 13:18:04" end="20090101 13:20:53" x="149938.24" y="-3706936.25"/>
      <activity type="minor" start="20090101 13:26:57" end="20090101 13:28:21" x="149771.17" y="-3707846.54" facility="501"/>
      <activity type="minor" start="20090101 13:40:31" end="20090101 14:05:42" x="144669.85" y="-3708507.55" facility="291"/>
      <activity type="minor" start="20090101 14:25:37" end="20090101 14:34:47" x="144669.85" y="-3708507.55" facility="291"/>
      <activity type="major" start="20090101 17:48:34" end="20090102 09:51:33" x="154826.99" y="-3710342.32"/>
   </chain>
   <chain>
      <activity type="major" start="20090103 17:49:37" end="20090103 23:42:52" x="145784.62" y="-3704569.65"/>
      <activity type="major" start="20090104 00:03:30" end="20090104 10:48:38" x="154826.97" y="-3710344.19"/>
   </chain>
   <chain>
      <activity type="major" start="20090104 00:03:30" end="20090104 10:48:38" x="154826.97" y="-3710344.19"/>
      <activity type="minor" start="20090104 10:51:44" end="20090104 11:01:08" x="155723.25" y="-3710114.82" facility="1833"/>
      <activity type="minor" start="20090104 11:13:32" end="20090104 11:17:43" x="151588.67" y="-3707520.18" facility="442"/>
      <activity type="minor" start="20090104 11:22:32" end="20090104 12:10:24" x="149703.77" y="-3705807.82"/>
      <activity type="minor" start="20090104 14:17:32" end="20090104 14:59:44" x="152615.96" y="-3508303.47"/>
      <activity type="minor" start="20090104 15:45:18" end="20090104 15:48:30" x="95527.04" y="-3432707.50"/>
      <activity type="minor" start="20090104 19:33:51" end="20090104 19:38:51" x="306907.95" y="-3056910.18"/>
      <activity type="major" start="20090104 22:03:06" end="20090105 11:04:22" x="423047.80" y="-2848191.07"/>
   </chain>
</digicoreVehicle>
```

Figure 4.2: Vehicle activity chains after modification, indicating with a unique `facility ID` whether an activity was performed at a clustered firm.

4.4 Identifying the edges in the complex network

During the final step, only direct freight trips made between two interesting clustered firms are included as edges in the complex network. To identify these trips, consecutive activities in the adapted activity chains are considered.

The first activity chain in Figure 4.2 has two major and five minor activities. The first pair of activities in this activity chain will not be included in the complex network because only one of them has a `facility ID`. The same applies to the next two pairs of activities.

The fourth activity's `facility ID` is `501` and the fifth activity's `facility ID` is `291`. This direct freight trip must be included in the complex network. Two new nodes, labelled as `501` and `291` respectively, are added to the complex network, and a directed edge is created from the former to the latter. If another trip occurs from `501` to `291`, the weight of the edge will

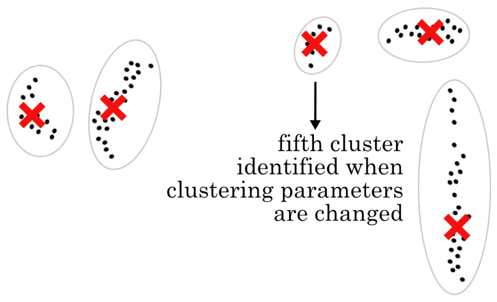


Figure 4.3: Changing the clustering parameters results in a fifth cluster being identified.

be incremented by one, because this edge already exists in the complex network. If a reverse movement occurs between these two firms (`291` to `501`), a bi-directional edge will be created between these two nodes.

The next freight trip in the activity chain starts and ends at the same facility, `291`. The origin and destination of this trip is the same, which often happens when freight vehicles have to wait in a queue to deliver their goods. These queues may take hours to dissipate, therefore freight vehicles switch off their engines while waiting. This trip is included in the complex network by a self-edge starting and ending at node `291`. The building of the complex network continues in this way until all activity chains of all vehicles have been considered.

4.5 Determining clustering parameters

The shape, size and the number of clusters generated are determined by the two clustering parameters, $p_{\min}$ and ϵ, which must be set by the user. Figure 4.3 illustrates this fact, since an additional cluster is identified by simply decreasing $p_{\min}$ to 5 points. Therefore more, or fewer, clusters can be identified by changing the configuration of clustering parameter values. The choice of clustering parameters are extremely important for subsequent analyses: complex network theory provides a vast number of metrics that can be used to study the connectivity of actors (firms) in the network. Nodes should be identified as accurately as possible so that later analyses will prove useful for firms that are visited by the freight vehicles.

Joubert and Axhausen (2013) choose the clustering parameters by testing a variety of clustering parameter values on 10 sample areas in the Gauteng province in South Africa. Based on the underlying land-use and visual inspection, they calculate a penalty score for each configuration of clustering parameters in all 10 study areas: the lower the score, the more visually accurate the clusters appear. The penalty scores are a metric for visual accuracy, measuring how well the clusters compare to what experts define as a *gold standard*. Visual accuracy of clusters are therefore subject to the opinion of the experts analysing the clusters. While this may be subjective, the *gold standard* approach is found to be repeatable and reproducible by Joubert and Meintjes (2015b). This means that similar results are obtained when different experts evaluate the clusters, as well as when the same expert evaluates the same clusters more than once. Visually accurate clusters intuitively make sense to the human eye: it does not group the activities at two firms on opposite sides of a road, but may possibly group the activities at neighbouring shops at a shopping centre. Based on Joubert and Axhausen's expert opinion, clustering parameter configuration, $\gamma = (\epsilon, p_{\min}) = (30, 15)$, is identified as the combination that results in visually accurate clusters. This means that, for a group of points to be included in the complex network as a facility (node), a minimum of 15 activities have to be performed withing a 30 metre radius of each other.

Joubert and Meintjes (2015a) argue that the visual accuracy of the clusters should not be the only deciding factor when choosing clustering parameters. Network completeness (the percentage of activities that are included in the complex network as nodes) and the computational complexity of building the complex network both need to be considered when choosing the clustering parameters. They perform multi-objective optimisation and determine that a much different clustering parameter configuration, $\gamma = (\epsilon, p_{\text{min}}) = (1, 2)$, maximises the completeness and minimises the computational complexity of the complex network.

The configuration (30, 15) often merges multiple firms into single large clusters (firms), while the combination (1, 2) splits single firms into many smaller clusters. These two extremes have to be reconciled, therefore another iteration of multi-objective optimisation is done, this time trading off completeness with visual accuracy.

To collect visual accuracy data for this round of optimisation, the *gold standard* approach, used by Joubert and Axhausen (2013), is followed. Ten new study areas are identified in the Nelson Mandela Bay Metropolitan area. For each study area, all clustering configurations containing radii $\epsilon \in \boldsymbol{E} = \{1, 5, 10, 15, 20, 25, 30, 35, 40\}$ and minimum point thresholds $p_{\text{min}} \in \boldsymbol{P} = \{1, 5, 10, 15, 20, 25\}$, the penalty scores are calculated. This results in $6 \times 9 \times 10 = 540$ penalty scores, 10 scores for each configuration. This is aggregated into one score per configuration by taking the average weighted sum of scores for each configuration, resulting in 54 penalty scores.

The completeness data are collected by determining, for each clustering parameter configuration, the percentage of activities that are (1) associated with a cluster, and (2) the percentage of these included in the complex network. The percentage completeness is determined for each configuration of clustering parameters, resulting in $6 \times 9 = 54$ completeness values.

The efficient frontier for visual accuracy and completeness is shown in Figure 4.4. Two points, (10, 10) and (20, 20), lie on the efficient frontier. When using either of these clustering parameter combinations, the resulting complex network will contain activities that are accurately associated with firms.

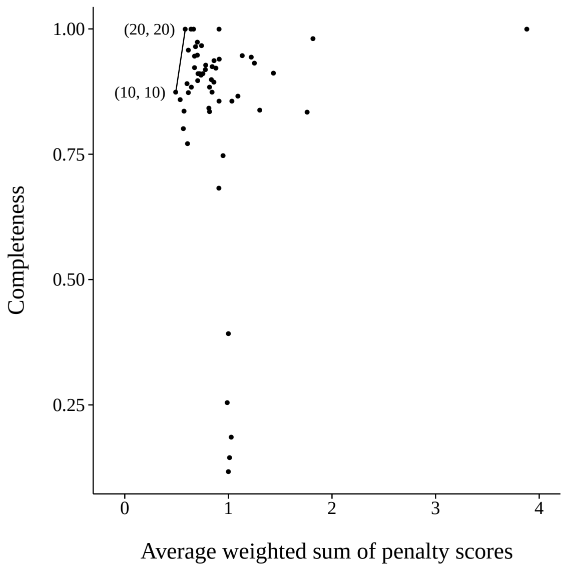


Figure 4.4: The efficient frontier when visual accuracy of the clusters and completeness of the complex network are traded off against each other. The efficient frontier is limited to two points.

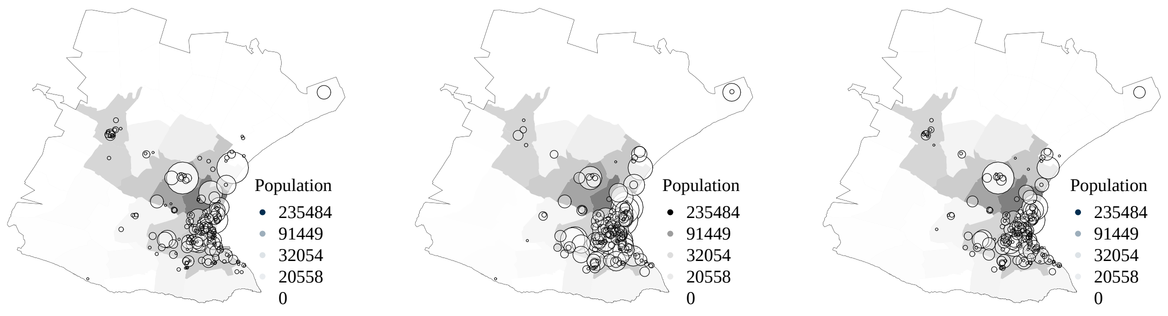
Following the above-mentioned steps, a complex network is built with the clustering parameter configuration (20, 20) using activity chains that have at least 60% of their activities performed in the Nelson Mandela Bay Metropolitan.

4.6 Preliminary analysis of the complex network

The freight complex network is analysed according to the standard network analytic techniques discussed in Chapter 2. This provides initial insights into the connectivity of the network, who the central firms are according to centrality scores, and where these central firms are located.

4.6.1 Node-level analysis

Three centrality scores are determined for the nodes in the complex network, namely betweenness, eigenvector, and degree centrality. The 200 nodes with the highest centrality scores are identified and plotted on a population density map of the Nelson Mandela Bay Metropolitan, as shown in Figure 4.5. Each centrality score is located at the location of the firm, and bigger circles indicate higher centrality scores.



(a) Betweenness centrality. (b) Eigenvector centrality. (c) Weighted degree centrality.

Figure 4.5: Mapping centrality scores on a population density map of Nelson Mandela Bay Metropolitan.

While the distribution of central actors differ between the different centrality scores, most central actors are located in the areas with higher density. This is similar to the results of Joubert and Axhausen (2013) and Joubert and Meintjes (2015a). Joubert and Axhausen (2013) note that this finding differs to developed countries, where the most central actors are located on the periphery of cities, due to the unique socioeconomic and spatial context of South Africa.

The three most central actors according to each centrality score is shown in Table 4.1. Firm `2314`, a brewery, is the most central actor according to betweenness and degree centrality scores. Firm `1351` (a group of distribution centres) and firm `1309` (a truck stop) both have an outgoing edge to firm `2314`, which makes them central by association and therefore the most and second most central actors according to eigenvector centrality. The second most central actor according to betweenness centrality is firm `899`, which is a fuel station and truck stop. Firm `1309` is the second most and third most central actor according to degree and betweenness centrality respectively. Lastly, the third most central actor according to eigenvector centrality is firm `630`, which has outgoing edges to both firm `899` and `1309`.

While truck stops may not intuitively be considered as important firms in terms of supply chain management, the nodes with the highest centrality scores reveal that they play a big role in the movement of freight vehicles. There is an incredible amount of movement to and from truck stops, and this finding can be exploited by decision makers. Suppose a new weigh bridge must be installed. It would make sense to install it at a location that is frequently visited by freight vehicles, therefore the central truck stops can be considered as possible locations. Another example is the implementation of new freight vehicle regulations. By frequently visiting central truck stops, the degree of compliance to the new regulations can be determined.

Table 4.1: The three most central actors according to the different centrality scores.

Ranking	Betweenness centrality	Eigenvector centrality	Degree centrality
1	2314	1351	2314
2	899	1309	1309
3	1309	630	1351

An alternative to studying centrality scores in isolation is to study combinations of centrality scores. By comparing eigenvector and betweenness centralities, Conway (2015) explains that *gatekeepers* and actors with *unique access* can be identified. Gatekeepers control the flow between other nodes, while nodes with unique access have exactly that: unique access to nodes that cannot easily be reached through other nodes in the network. Figure 4.6 shows the relationship between these two centrality scores. Eigenvector centrality and betweenness centrality are expected to have a linear relationship, and should therefore produce a linear distribution. The residual values for all data points are calculated to determine how far the point deviates from the expected linear relationship, and the residual values are indicated with different grades of transparency. Darker points indicate higher residual values, and are therefore outliers. The top three firms with the highest residual values are firms 899, 2314, and 1309 and are labelled in the figure. These three firms are gatekeepers since their betweenness centrality scores are higher than their eigenvector centrality scores. When decision makers must ensure that information should (or should not) reach other firms, firms 899, 2314, and 1309 can be targeted to ensure (or prevent) the spread of information.

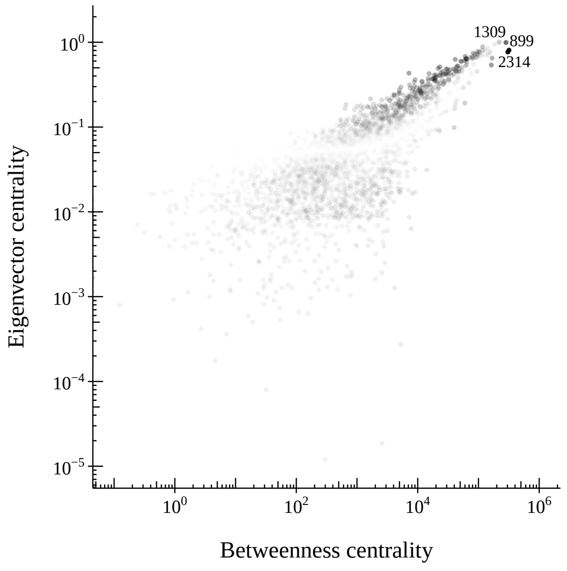
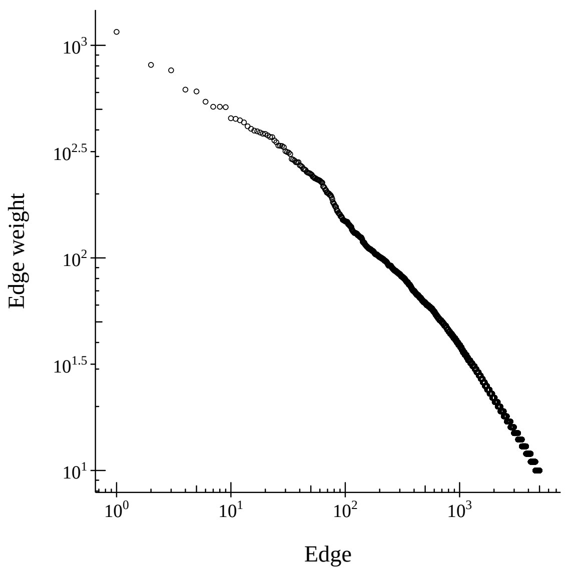


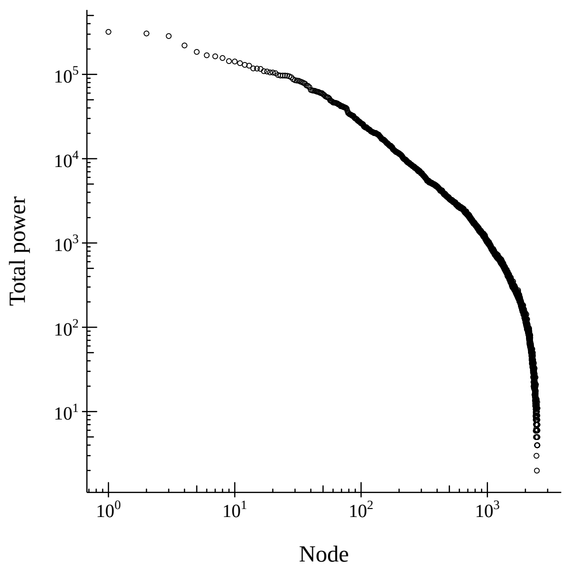
Figure 4.6: Mapping the eigenvector centrality against the betweenness centrality scores of firms.

4.6.2 Edge-level analysis

All edges in the complex network represent the same type of relation: direct freight vehicle trips between firms. Each edge has a weight associated with it, and can be used as a proxy of the strength of the relationship between two firms. The distribution of edge weights plotted on a log-log scale can be seen in Figure 4.7(a). Even though this distribution appears to follow a power law distribution, a *goodness-of-fit* test results in a p-value of 0, indicating that it does not follow a power law.



(a) Edge weight distribution on a log-log scale.



(b) The distribution of power on a log-log scale.

Figure 4.7: The distribution edge weight and power of the complex network.

4.6.3 Network-level analysis

The distribution of power in the freight network is shown in Figure 4.7(b). The centrality scores are added to get the total power of each node, and by performing the *goodness-of-fit* test, a *p*-value of 1 is obtained, indicating that it follows a power law distribution.

The remaining network-level properties are compared to three other complex networks from different fields of study, namely air traffic and animal behaviour. Data for these complex networks are obtained on the website of the Koblenz Network Collection project collected by the Institute of Web Science and Technologies at the University of Koblenz-Landau, Germany (University of Koblenz-Landau, 2015).

Table 4.2 summarises key network properties of four complex networks, including the freight complex network used in this dissertation. The airport network reports on airplane movements (directed and weighted edges) between American airports (nodes) in 2010. The monkey network reports on the grooming habits of free roaming Rhesus monkeys during a two month study conducted in 1963. A directed edge from one monkey to another shows that the former groomed the latter, and a bi-directional edge indicates that the grooming was reciprocated. The bison network represents the display of dominance between male bison in 1972 on the National Bison Range in Montana, USA. A directed edge from one bison to another indicates that the former exhibited dominant behaviour toward the latter, while edge weights indicate the number of times this behaviour was observed.

The freight complex network contains 2,594 nodes and 77,989 edges, making it sparse with a network density of only 1.122%. The airport network has approximately half the number of nodes and edges as the freight complex network, and has a similar network density (0.0114) because the relationship between number of nodes and edges is similar to that of the freight network. The monkey network has a very high density (0.4625), because it has many edges in relation to the number of nodes. Similar to this, the bison network has a density of 0.483.

All nodes are connected to each other by paths (combinations of nodes and edges). The shortest path between two nodes is the path with the least number of edges between the two

Table 4.2: Network properties of various complex networks.

Network property	**Freight**	**Airport**	**Monkey**	**Bison**
Number nodes	2,954	1,574	16	26
Number edges	77,989	28,236	111	314
Density	0.01122	0.0114	0.4625	0.4830
Diameter	6	9	4	4
Average clustering coefficient	0.1775	0.3841	0.6706	0.7887
Reciprocity	0.4927	0.7806	0.7568	0.3877

nodes. The diameter is the longest shortest path in a network, and gives an indication of how easily nodes can interact with one another. The freight network has a diameter of 6, which is slightly smaller than the diameter of the airport network (9). The monkey and bison networks have the smallest diameter of all four networks (4).

The average clustering coefficient in the freight network is 0.1775, which means that if two nodes are connected to a third mutual actor, there is a 17% chance that they are also connected to each other. Mutually connected nodes in the airport and monkey networks have a much higher probability of being connected to each other, while the bison network's average clustering coefficient indicates that there is a 78% chance of two bison, who have been dominated by a third bison, exhibit the same dominating behaviour on each other.

The reciprocity of the freight network, which is a measure of the number of edges with reciprocal relationships, is 0.4927. Therefore approximately half of the relationships in the network are reciprocal. If there was a truck trip from a to b, there is almost a 50% chance that another truck trip will occur from b to a. It may be that, after a truck has travelled from a to b, it completely unloads its load and returns to a with an empty truck. Therefore the reciprocated movement occured as a result of the first movement. Alternatively, one truck can travel from a to b, and at another time, independent of the first trip, another truck travels from b to a. Therefore the two trips are independent of each other. While the reason of the truck trips are unknown, the complex network proves that there exists a reciprocal relationship between facility a and b.

The airport network has the highest reciprocity (0.7806), which makes sense since there are usually numerous flights in both directions between two airports. The monkey network also has a high reciprocity (0.7568), which indicates that the Rhesus monkeys are a tightly knit group. The bison network has the lowest reciprocity (0.3877), which makes sense since the animal that exhibits dominant behaviour is often stronger than those who are being dominated, therefore the weaker bison will not reciprocate the behaviour.

4.7 Conclusion

Node centrality scores indicate that `2314`, `1351`, `899`, `1309` and `630` are the most central firms in the network. Decision makers can target these firms when policies or information must be disseminated quickly through the network. The firms in the statistically significant motifs can also be considered central in the complex network, because they form part of the building blocks of the network. These firms are identified in the next chapter.

Chapter 5

Identifying and analysing motifs in the complex network of freight movements

In this chapter the two research questions are addressed. To answer the first research question, ISMAGS is used to determine whether there are statistically significant motifs in the freight complex network built in the previous chapter. The second research question, identifying firms that frequently appear in motif instances, is answered by performing various analyses on the motif instances.

5.1 Identifying statistically significant motifs

The *original* freight complex network refers to the empirical complex network built from real GPS traces of freight vehicles travelling in the Nelson Mandela Bay Metropolitan. The *random* networks refer to the randomly generated networks that have the same degree distribution as the original network. Two methods are used to identify statistically significant motifs, namely comparing the original subgraph frequency to the mean subgraph frequency of the random networks, and using the significance profile. These two methods may result in two unique sets of statistically significant motifs, and the intersection of these sets will be chosen for further analysis.

5.1.1 Determining subgraph counts

A number of steps are performed to determine the subgraph counts in the original freight network and the random networks. These steps are described as follows:

1. **Determining motif specifications** of all directed 3-node subgraphs. This is done in Chapter 3 for subgraphs of size 3, 4 and 5. All 13 directed 3-node subgraphs, along with their motif specifications, are shown in Figure 5.1.

2. **Generating random networks** by implementing the matching algorithm in *Java* (Algorithm 2). To determine the number of random networks to generate, similar work by other authors are considered. Kavurucu (2015) simply states that a *sufficient* number of random networks should be generated. Ribeiro and Silva (2010) only generate five random networks, and Wernicke (2006) suggests that at least a thousand random networks be generated. In the seminal work done by Milo et al. (2002), 100 random networks are generated. Following the example of this seminal work, 100 random networks are generated in this dissertation.

3. **Splitting the edge lists** into one- and bi-directional edges is required by ISMAGS. Therefore Algorithm 12 in Chapter 3 is run on the freight complex network, as well as all on 100 random networks.

4. **Running ISMAGS** by using Algorithm 13 on the original freight complex network, as well as on all 100 random networks. This results in 13 subgraph instance lists for each network.

5. **Filtering subgraph instances** is required to ensure that the number of subgraph instances with a '0' in their motif specification is not inflated. As in Chapter 3, Algorithm 14 is run on the original freight network as well as on all 100 random networks.

6. **Determining subgraph counts** for each network by counting the number of lines in each output file (since each line contains one subgraph instance).

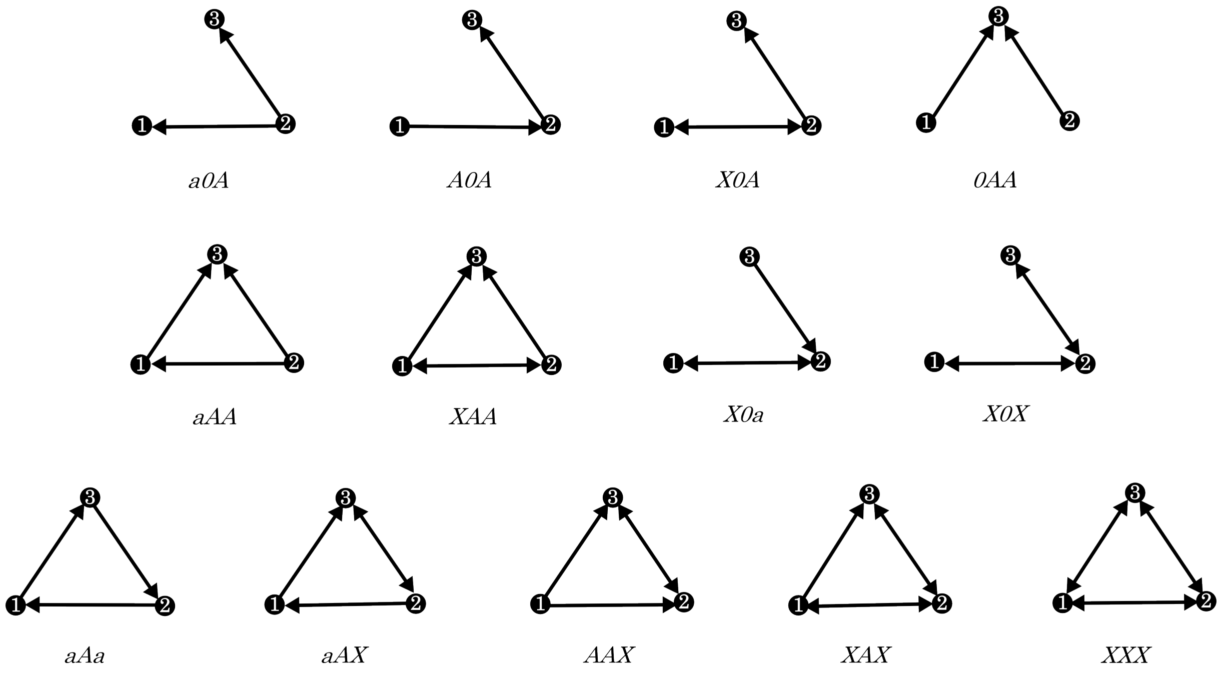
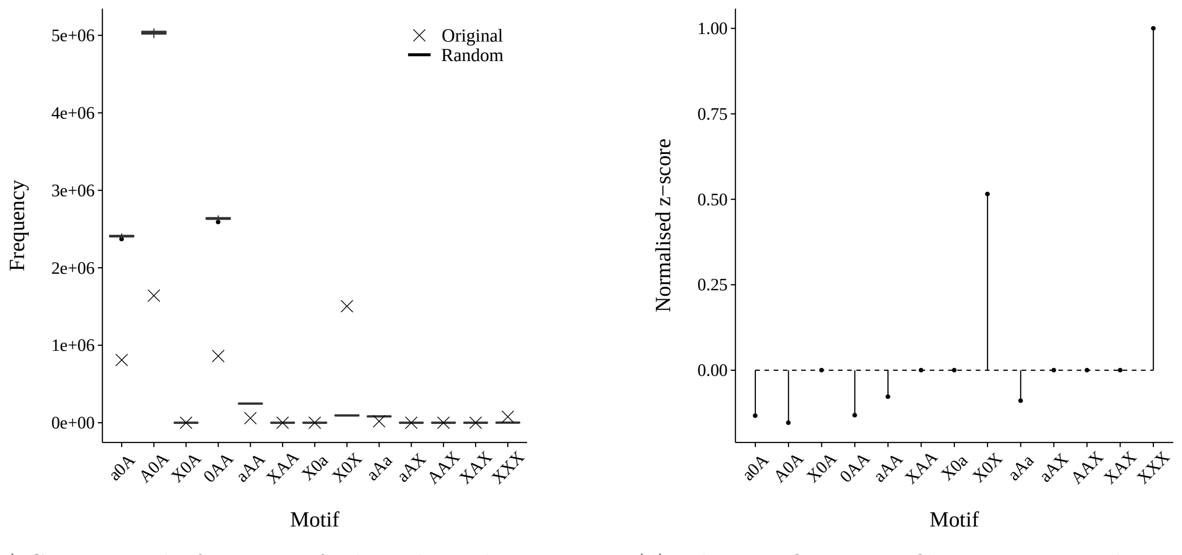


Figure 5.1: The 13 directed 3-node subgraphs with their motif specifications.

5.1.2 Determining statistically significant motifs

The original frequency and random frequencies for the 3-node subgraphs can be seen in Figure 5.2(a). The data for the random networks are plotted as boxplots. The random frequencies have very thin boxes due to the small variation, with little to no outlier points. For subgraphs *X0A*, *XAA*, *X0a*, *aAX*, *AAX* and *XAX* there are no occurrences in either the original or random networks. Subgraphs *0AA*, *a0A*, *A0A*, *aAa*, and *aAA* occur, on average, less frequently in the original network than in the random networks. The fact that these subgraphs are under-represented in the original network is in itself significant, and these subgraphs are called *anti-motifs*. While the term is introduced by Milo et al. (2004) and is subsequently mentioned by a number of authors (Frankenstein et al., 2006; Jackups et al., 2006; Berka, 2012), no literature discussing the expected behaviour of anti-motifs could be found. The only two statistically significant motifs are *X0X* and *XXX*, since the original frequency is greater than the random frequencies. Motif *X0X* occurs approximately 16 times more in the original network than in the random networks, and *XXX* occurs approximately 57 times more in the original network than in the random networks.



(a) Comparing the frequency of subgraphs in the original network to the distribution of frequencies of subgraphs in the random networks.

(b) The significance profile representing the normalised z-scores of each subgraph.

Figure 5.2: Determining statistically significant motifs. In both figures, subgraphs *X0X* and *XXX* are statistically significant motifs.

The significance profile represents the normalised z-scores of each 3-node subgraph, shown in Figure 5.2(b). Each normalised z-score is plotted as a circle, with a solid vertical line connecting it to a dashed line. The dashed line is plotted at 0, which is the threshold for determining statistical significance. Subgraphs that have a normalised z-score of 0 lie on the dashed line, and this indicates that there are none of these occurrences in either the original or the random networks. This is the case for subgraphs *X0A*, *XAA*, *X0a*, *aAX*, *AAX* and *XAX*. Subgraphs *0AA*, *a0A*, *A0A*, *aAa*, and *aAA* have z-scores less than 0 because they occur less frequently in the original network than on average in the random networks, and therefore are not statistically significant. Subgraphs that have normalised z-scores greater that 0 are statistically significant motifs, because they occur more in the original network than on average in the random networks. Subgraphs *X0X* and *XXX* are therefore statistically significant motifs. *XXX* is more significant than *X0X* due to its high normalised z-score.

Therefore both methods result in the same set of statistically significant motifs: *X0X* and *XXX*. This means that when three firms have two or three reciprocal relationships between them, these patterns of relationships occur significantly more in the freight complex network than any other pattern of firm relationships. The analyses are continued on all instances of these *X0X* and *XXX* motifs.

5.2 Analysing statistically significant motifs

In this dissertation, a motif instance is made up of relationships (edges), and each relationship exists between two firms (nodes). Following this hierarchical structure, analyses are performed on the motif instances at a motif-level, then at an edge-level (relationships), and finally at a node-level (firms).

5.2.1 Motif-level analysis

The 40,000 freight vehicles in the *Digicore* data set travelled thousands of kilometres over the six-month period that the data were collected. Each freight vehicle has their own agenda: one may be delivering spare parts and another may be a courier. While freight vehicles make direct trips between firms to perform their freight-related activities, they unknowingly create significant patterns in the freight complex network. Figure 5.3 shows how the activity chains of different freight vehicles, represented by different line types, can create motifs. In Figure 5.3(a) an *X0X* motif is highlighted with two thick transparent grey lines, while Figure 5.3(b) highlights an *XXX* motif in a similar way.

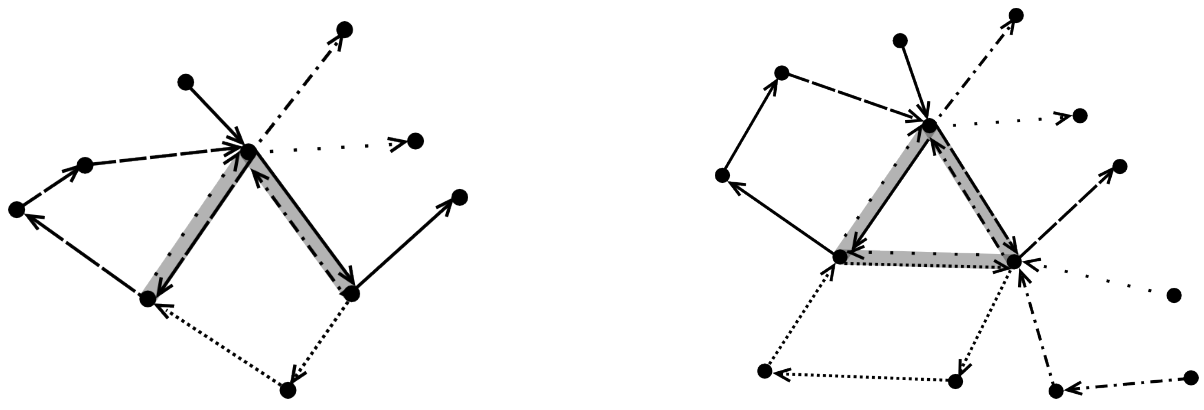
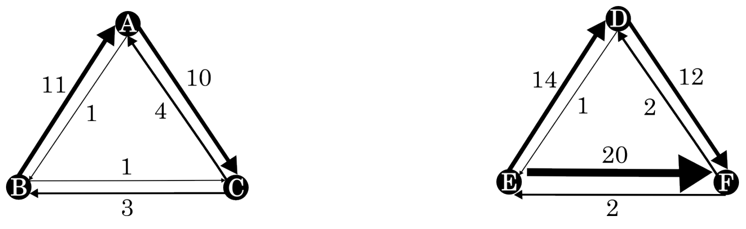

(a) An *X0X* motif. (b) An *XXX* motif.

Figure 5.3: Multiple activity chains, represented by different line types, forming motifs.

Ranking motifs by total freight trips

The relative importance of nodes in a complex network can be determined by calculating node centrality scores. Can the principle of *central actors* in a network be extended to *central motifs* in a network? Are some motif instances more important than others?

One way to quantify a motif's importance is by considering the number of direct freight trips that occur between the firms in the motif instance. Consider, for example, the two *XXX* motif instances shown in Figure 5.4. Each edge in the motif has an edge weight associated with it, which represents the number of direct freight trips between the two firms. A total of $11+10+1+1+4+1+3=30$ direct freight trips occurred between firm `A`, `B`, and `C` in the motif in Figure 5.4(a). More direct freight trips occurred between firm `D`, `E`, and `F` in the motif in Figure 5.4(b), since it has a total edge weight of $14+12+1+2+20+2=51$. Therefore, if a motif's importance is quantified by total edge weight, firms `D`, `E`, and `F` form the more important firm relationships.



(a) An *XXX* motif (`A`, `B`, `C`) with a total edge weight of 30. (b) An *XXX* motif (`D`, `E`, `F`) with a total edge weight of 51.

Figure 5.4: Comparing the importance of motifs using total edge weight.

To rank the motif instances of the freight complex network, the total edge weight of each motif instance is determined. This is done by implementing Algorithm 15 in *Java*. The algorithm requires a list of motif instances and a weighted network edge list, which is a list of network edges along with their edge weights. The algorithm iterates through all motif instances in lines 2–11, and determines the weight of each edge in the motif instance (line 4). There are four edges to iterate over in *X0X* motif instances, and six in *XXX* motif instances. The total edge weight thus far is calculated in line 8, and after all edges in the motif instance have been iterated over, the motif instance and its total edge weight are added to the weighted motif list in line 10.

Algorithm 15 Determining edge weights of motif instances.

```
Input: List of motif instances $\boldsymbol{I}$, Weighted network edge list $\boldsymbol{E_W}$
Output: List of weighted motif instances $\boldsymbol{I_W}$
 1: procedure determineTotalMotifEdgeWeight(list $\boldsymbol{I}$, list $\boldsymbol{E_W}$)
 2:     for all motif instances $i \in \boldsymbol{I}$ do
 3:         total weight $w \leftarrow 0$
 4:         for all edges $e \in i$ do
 5:             $o \leftarrow$ origin node of $e$
 6:             $d \leftarrow$ destination node of $e$
 7:             $w_1 \leftarrow$ weight of $(o, d)$ in $\boldsymbol{E_W}$
 8:             $w \leftarrow w + w_1$
 9:         end for
10:         add $i$ and $w$ to $\boldsymbol{I_W}$
11:     end for
12:     return list of weighted motif instances $\boldsymbol{I_W}$
13: end procedure
```

It is expected that there will be a few motif instances with a large number of direct freight trips between all three firms in the motif instance, therefore all edges in the motif will have a high edge weight. These motif instances can then be deemed more important than the other instances because all three firms contribute relatively equally to the strong relationships in the motif instance. However, the results reveal that only one extremely strong relationship is required in a motif instance for it to be considered important.

The weighted motif instance list generated by Algorithm 15 is sorted in descending order according to total edge weight. The 10 *X0X* and *XXX* motif instances with the highest total edge weights are shown in Table 5.1. The highlighted cells show repeating edges between firm 1716 and 2314. The top 10 *X0X* and *XXX* motif instances contain these two edges because of their high edge weights. The edge weight distribution is shown in Figure 5.5, with edge (2314, 1716) labelled as the edge with the highest edge weight (1,157) in the entire network, and edge (1716, 2314) labelled as the edge with the fourth highest edge weight (619) in the entire network.

The first *X0X* motif instance in Table 5.1, (2314, 1716, 1294), is the most important in terms of total freight trips. But this motif instance is important because of two extremely strong relationships between firm 2314 and 1716, not because of the relationships with firm 1294. Only 98 and 13 direct freight trips were made to and from firm 1294, as labelled on Figure 5.5, resulting in weak relationships with firm 1294 in comparison to the rest of the relationships in the motif instance. Yet, firm 1294 is part of the most important *X0X* motif instance. Therefore a firm does not need to have extremely strong relationships in order for it to be considered part of an important motif instance. Instead, it can form part of an important motif instance by having relatively weak relationships with firms that have extremely strong relationships with other firms. Therefore firm 1294 is important *by association*. This principle is similar to eigenvector centrality, where a node is central merely by being associated with a central node.

For both *X0X* and *XXX*, the most central motifs are those that contain direct freight trips between a brewery (firm 2314) and a truck stop (firm 1716). There exists a bi-directional

Table 5.1: Top 10 *X0X* and *XXX* motif instances according to total edge weight.

Ranking	***X0X* motifs**			***XXX* motifs**		
	Node 1	Node 2	Node 3	Node 1	Node 2	Node 3
1	2314	1716	1294	1768	1794	1764
2	1716	2314	863	899	2314	1716
3	1716	2314	2026	1351	2314	1716
4	1716	2314	2412	2314	1716	1320
5	2314	1716	1593	2314	1716	1747
6	1716	2314	2342	2314	1716	2459
7	1478	1716	2314	2314	1716	729
8	1716	2314	1142	1309	2314	1716
9	1716	2314	2388	1616	92	1416
10	1716	2314	2438	2314	2265	1716

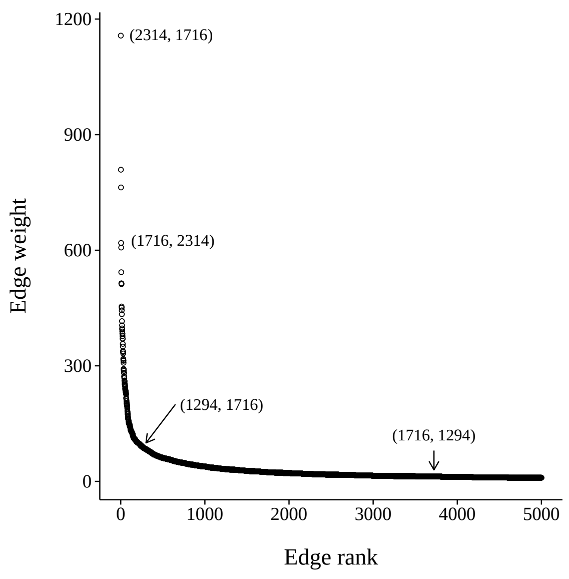
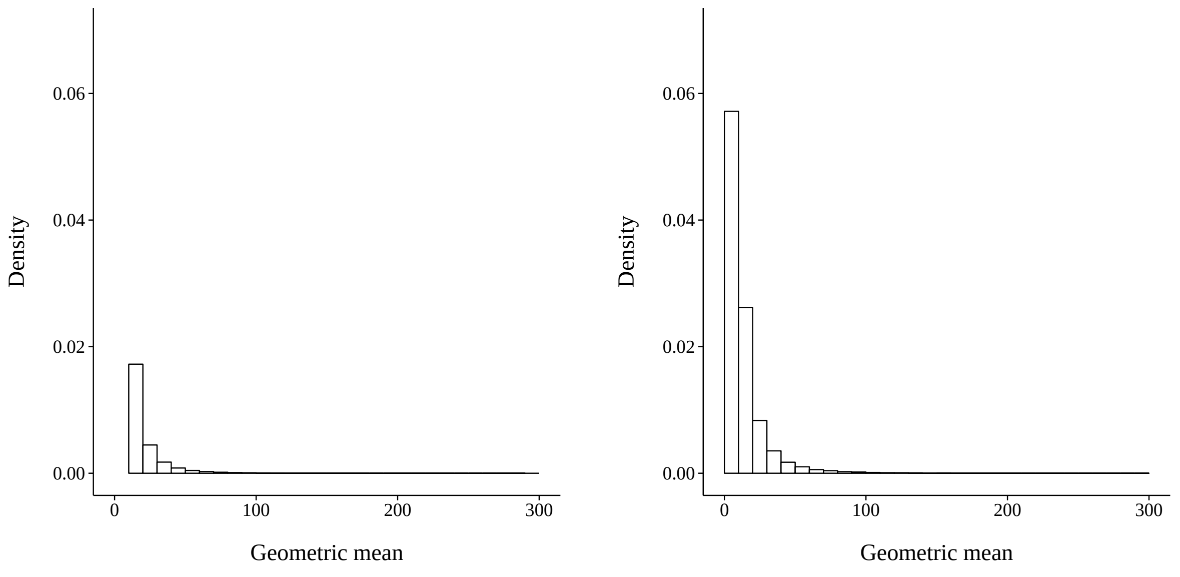


Figure 5.5: The edge weight distribution of the freight complex network. Only the 5,000 highest edge weights are shown.

edge between these two firms, therefore freight trips occur in both directions. It makes sense that direct freight trips occur from the truck stop to the brewery: a long haul freight vehicle travelling to the brewery could arrive late at night, make an overnight stop at the truck stop, and deliver the goods in the morning. Or perhaps the brewery has dedicated delivery hours and the freight vehicle arrives ahead of its scheduled delivery. But why would so many direct freight trips occur from the brewery to the truck stop? There is a fuel station on the premises of the truck stop, and it would make sense if the brewery's freight vehicles always fill up before making their deliveries. The ranking of motif instances according to total freight trips therefore provides interesting insights into the brewery's supply chain and firm behaviour, but a lot more insight can be gained by conducting more detailed studies with the cooperation of the brewery.

Internal variation of edge weights

The previous analysis has shown that some of the motif instances have a dominating relationship with a very large edge weight. Calculating the arithmetic mean of the edge weights therefore skews the average towards the dominating edge, not giving any indication of the edge weights of the remaining edges. Therefore this analysis looks at the internal edge weight difference by using the geometric mean of edge weights. In this way, the central tendency of edge weights for both *X0X* and *XXX* motif instances can be determined. Figure 5.6 illustrates the histogram of the geometric mean of edge weights for *X0X* and *XXX* motif instances respectively. Both histograms have a bin width of 10, with the same x- and y-scale.



(a) *X0X* motifs. (b) *XXX* motifs.

Figure 5.6: Geometric mean of motif edge weights.

The results indicate that there are more *XXX* motif instances with smaller internal edge weight differences.

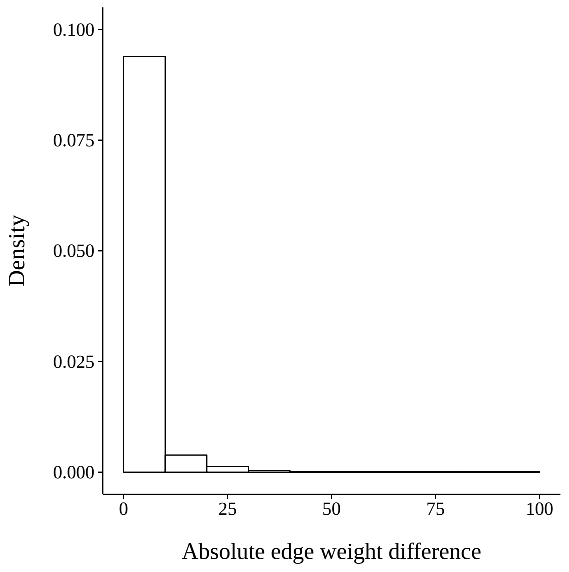
Difference in strength of motif relationships

Similar to Joubert and Axhausen (2013), the assumption is made in this dissertation that a direct freight trip between two firms indicates the existence of a relationship between them. Therefore the higher the number of direct freight trips between two firms, the stronger the relationship between them. In all *X0X* motif instances, there are four relationships, and in *XXX* motif instances, there are six relationships. As seen in the previous analysis, edge weights can differ greatly in a single motif instance. The question is, do all motif instances follow this behaviour? Or do some motif instances have relationships that are of similar strength?

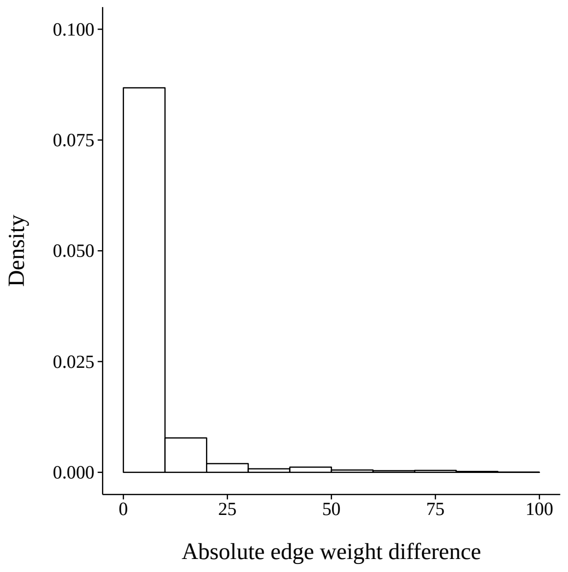
To answer this question, the absolute difference between the edge weights in all motif instances is considered. The minimum edge weight in each motif instance is subtracted from the maximum edge weight in that motif instance. There may be some bias in this analysis in that *XXX* motif instances have two more edges than *X0X* motif instances, and therefore the absolute difference may be inherently larger. Therefore, for *X0X* motifs, the absolute difference is divided by four, and for *XXX* motifs, the absolute difference is divided by six. The absolute differences in edges weights are shown as histograms in Figure 5.7. Both histograms have a bin width of 10.

The *X0X* and *XXX* motif instances with the highest difference in edge weight are those that have an edge from firm `2314` to `1716`, since this edge has the highest edge weight in the entire network. The previous analysis has already shown that motif instances with these two firms have a large difference in edge weights. The purpose of this analysis is to focus on the absolute edge weight differences in the remaining motif instances. Therefore the x-axis limits in Figure

5.7 only continue to 100. Both Figures 5.7(a) and 5.7(b) are plotted on the same y-scale for easier comparison.



(a) *X0X* motifs.



(b) *XXX* motifs.

Figure 5.7: Absolute differences in motif edge weight.

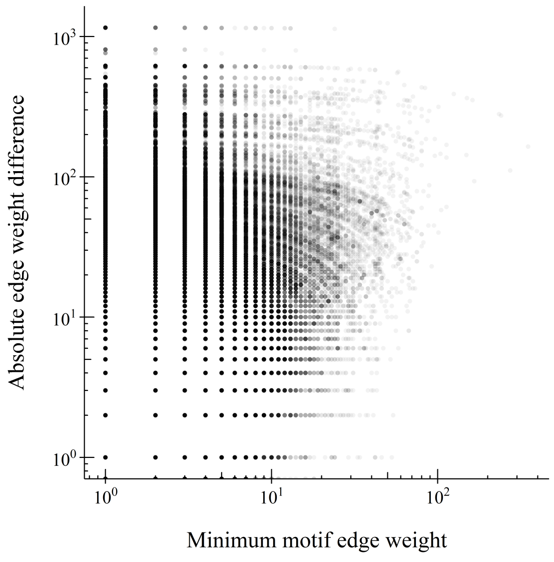
The results indicate that *X0X* and *XXX* motif instances have similar absolute edge weight differences. Approximately 9% of all *X0X* and *XXX* motif instances have an absolute edge weight difference between 1 and 10. Therefore the majority of *X0X* and *XXX* motif instances have relationships of similar strength. This could indicate that firms in *XXX* and *X0X* motif instances have similar functions in their supply chains. This can possibly be investigated further in future work.

Relationship between absolute difference and minimum edge weights

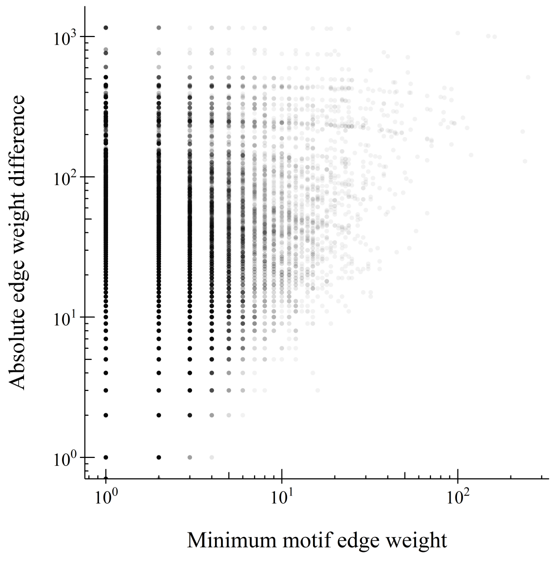
When a motif instance has an absolute edge weight difference between 1 and 10, it gives no indication of the value of the minimum edge weight: the minimum may be 1 or it may be 1,000. It may be that the smaller the absolute edge weight difference, the lower the minimum edge weight of the motif instance. There may also be an inverse relationship, or none at all.

The purpose of this analysis is to determine the relationship between the absolute difference in edge weight and the minimum edge weight of motif instances. For each motif instance, its absolute difference in edge weight is plotted against its minimum edge weight on a log-log scale in Figure 5.8. For both *X0X* motif instances (Figure 5.8(a)) and *XXX* motif instances (Figure 5.8(b)), there exist many instances with low minimum edge weights with various absolute edge weight differences. As the minimum edge weight of *X0X* motif instances increases, it seems that most of the motif instances have an absolute edge weight difference between 10 and 100. As the minimum edge weight of the *XXX* motif instances increase, most of the motifs gravitate towards a larger absolute edge weight difference.

There would be no motif instances identified in the freight complex network if there were no relationships and freight movements between the firms in the network. Therefore the firm relationships play an important role in motif formation, and these relationships are studied next.



(a) *X0X* motifs.



(b) *XXX* motifs.

Figure 5.8: Motif absolute edge weight difference and minimum edge weight on a log-log scale.

5.2.2 Edge-level analysis

Transport cost is one of the major expenditures when it comes to supply chain management. There are many ways to decrease transport costs, such as load consolidation or milk runs. Another way, albeit more extreme, is to decrease the distance that freight vehicles have to travel by moving closer to important suppliers or customers or by using intermediary storage facilities.

Freight trip frequency and distance

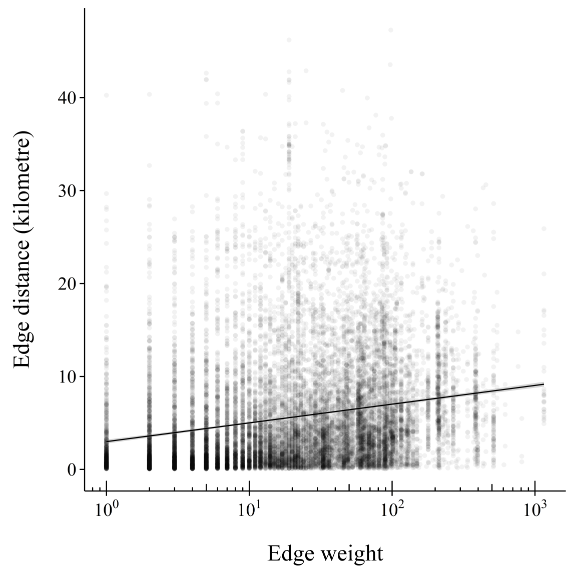
When two firms have strong relationships with each other, there will be a lot of freight traffic between them, increasing the transport costs. Therefore it would make sense for these firms to be located close to each other. Firms with weaker relationships will not have as much freight traffic between them, and can therefore afford to travel longer distances. Therefore the stronger the relationship between firms, the more crucial the distance between them.

This edge-level analysis determines the relationship between the number of direct freight trips and the distance travelled by freight vehicles. For each edge in an *X0X* or *XXX* instance, its weight and distance are determined. Edge distance is calculated as the Euclidean distance between the firms of that edge. The edge weights and edge distances are plotted in Figures 5.9(a) and 5.9(b). The x-axes are plotted on log scales to improve the readability of the graphs, and each data point is shown as a slightly transparent circle, therefore darker circles indicate that multiple edges have similar edge weights and distances.

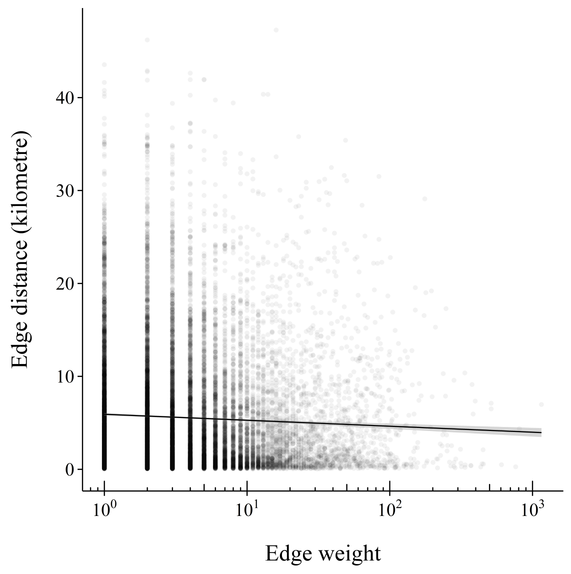
For *X0X* motif instances, most edge weights lie between 10 and 1,000, with most of the distances less than 10 kilometres. This translates to between one direct freight trip every second day, and 15 direct freight trips per day. 15 direct freight trips per day is an incredible amount of freight movement, adding up to at least 1,500 kilometres travelled per day (if return trips are not taken into account).

For *XXX* motif instances, there are many edges with low edge weights (less than 10, which is approximately one direct freight trip every 10 days). Since the number of freight trips are very low, it is not crucial for the distance travelled to be low. For the rest of the edges, the distances seem to decrease slightly as the edge weight increases.

Trend lines are fitted to Figures 5.9(a) and 5.9(b). The *X0X* edge weights increase slightly as edge distances increase, increasing transport costs. Firms in XXX motif instances are located closer to each other when there are more freight trips between them. Therefore firms in XXX motif instances apply the concept of decreasing the distance travelled when there is a lot of freight movement between them, reducing the total transport cost. There are, however, some XXX motif instances that are outliers: they have high edge weights and long edge distances. The highest edge weight is 1,157 with an edge distance of 25.9 kilometres. Travelling this far so frequently indicates areas for improvements. These outlier firms can be used by research teams that wish to study firms who's freight vehicles often travel long distances, or when implementing policies regarding excessive freight trips.



(a) $X0X$ motifs.



(b) XXX motifs.

Figure 5.9: Motif instance edge weights versus distances.

The edges studied in these analyses have two firms connected to their endpoints. To gain more detailed insight into the motif instances, the next analyses will focus on these firms as individuals.

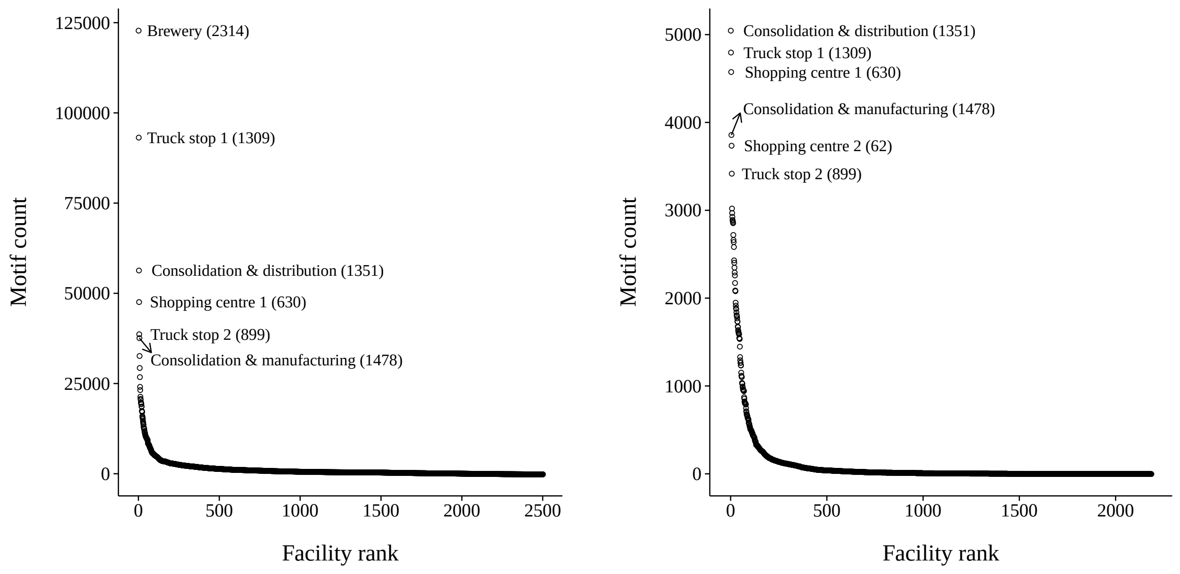
5.2.3 Node-level analysis

In this dissertation degree centrality measures the number of relationships a firm has with other firms. If a firm has many suppliers and customers, it can be considered important, because it can interact with many firms. This concept can be extended to motif instances: if a firm appears in many motif instances, it can be considered important because it forms part of multiple tightly bounded groups of firms. It is postulated that a firm with a high degree centrality will also occur in many motif instances, because the more connections a firm has, the more motif instances it can appear in.

Ranking firms by motif counts

One way to quantify the importance of a firm is by counting the number of motif instances it appears in, referred to as its *motif count*. The motif counts are determined for each firm and are ranked from highest to lowest. The distribution of $X0X$ motif counts are shown in Figure 5.10(a), and that of XXX are shown in Figure 5.10(b). The six most important firms are labelled in each figure. The highest motif count for $X0X$ is almost 125,000 for a brewery, while that of

XXX is approximately 5,000 for a group of consolidation and distribution centres. The top 10 firms, along with their motif counts, are listed in Table 5.2.



(a) *X0X* motifs.

(b) *XXX* motifs.

Figure 5.10: The number of motifs each firm appears in.

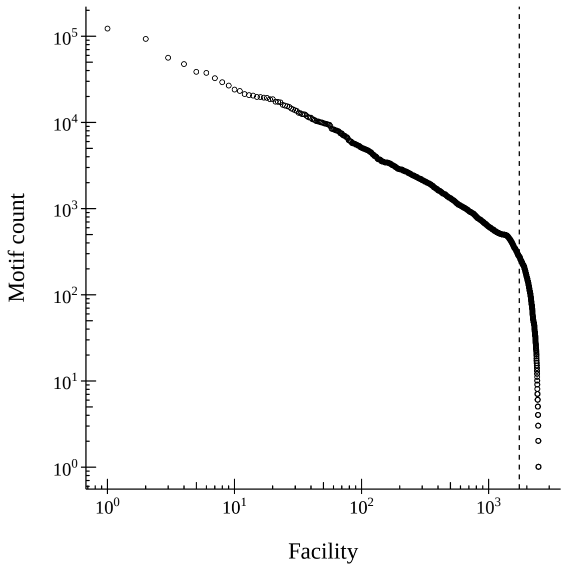
Table 5.2: The firms that occur in the most *X0X* and *XXX* motif instances.

Ranking	**X0X motifs**			**XXX motifs**		
	Firm	Motif count	Degree ranking	Firm	Motif count	Degree ranking
1	2314	122,785	2	1351	5,045	3
2	1309	93,149	1	1309	4,796	1
3	1351	56,324	3	630	4,574	4
4	630	47,567	4	1478	3,857	7
5	899	38,650	5	62	3,735	8
6	1478	37,581	7	899	3,417	5
7	1636	32,625	6	269	3,020	19
8	62	29,304	8	2141	2,969	9
9	1645	26,777	12	1636	2,926	6
10	1695	24,071	10	2314	2,891	2

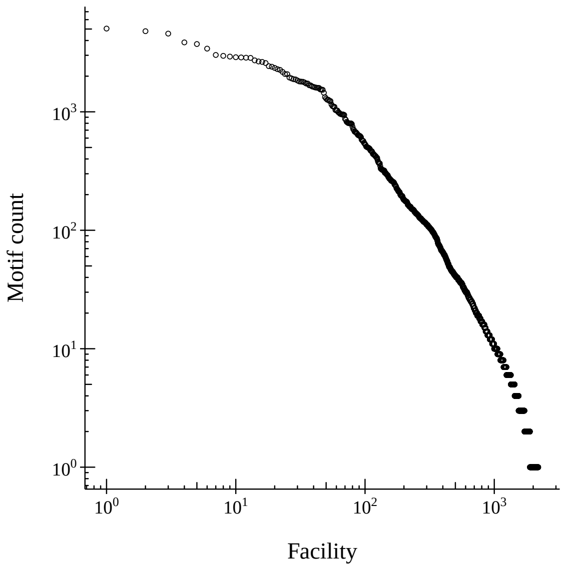
One intuitively knows that firms such as shopping centres, truck stops, and consolidation centres will attract a lot of freight movement. The results indicate these type of firms are important to the network as a whole, since they occur in the most number of motifs. If something were to happen to one of these firms, thousands of other firms will be impacted.

It is interesting to see that five of the top six firms are the same for both *X0X* and *XXX* motif instances. These are two truck stops (1309 and 899), a shopping centre (630), a group of consolidation and distribution centres (1351), as well as a group of consolidation centres and manufacturing plants (1478). Therefore if a firm appears in a lot of *X0X* motif instances, it will likely also appear in a lot of *XXX* motif instances.

The motif count distributions are shown on log-log scales in Figure 5.11. The null hypothesis is that these distributions follow a power law (with few firms that have very high motif counts, and many firms that have less motif counts). The *goodness-of-fit* method proposed by Clauset et al. (2009) is used to test this hypothesis. When a distribution's p-value is greater than 0.1 it is considered to follow a power law. The goodness-of-fit test is performed on both distributions, and it is found that $p_{X0X} = 0.16$ and $p_{XXX} = 0$. Therefore only the *X0X* distribution in Figure 5.11(a) follows a power law. The dashed line indicates that $x_{\text{min}} = 1,744$, which is the value above which the *X0X* motif counts follow a power law.



(a) *X0X* motifs.



(b) *XXX* motifs.

Figure 5.11: The number of motifs each firm appears in on a log-log scale.

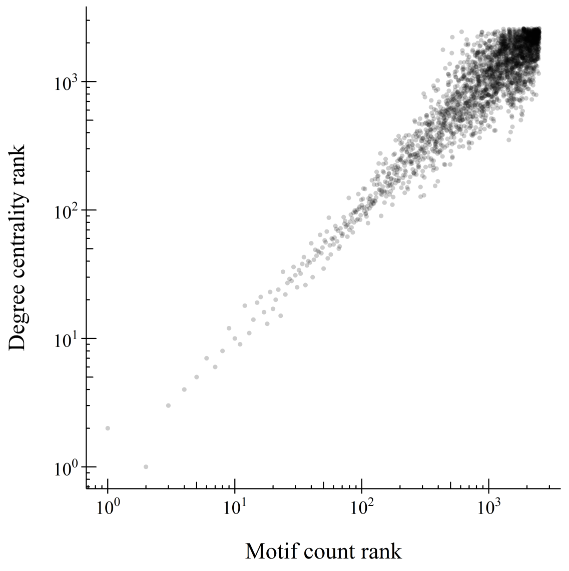
Similar to node centrality, which determines the importance of nodes in terms of their connectivity to other nodes, the node-level analysis determines the importance of firms in the motif instances. If a firm appears in many motif instances, it has many tightly bounded relationships with groups of firms and can be considered a *central firm*. If something happens to a central firm, many firms will be impacted, therefore it is important to determine who the central firms are.

Relating degree centrality to motif count

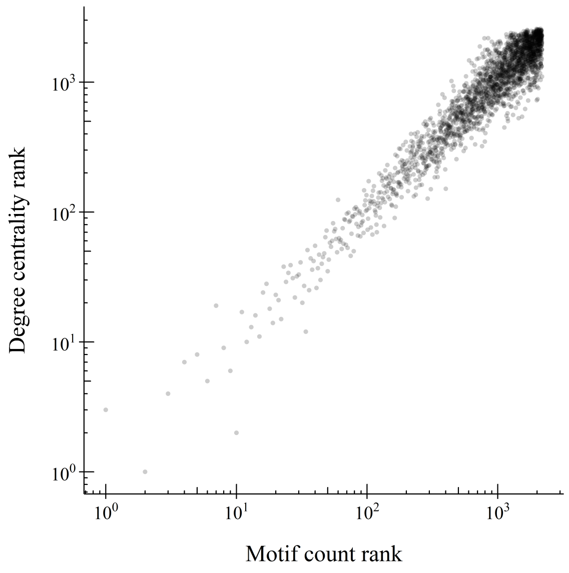
The ranking of firms according to degree centrality is listed in Table 5.2, and it indicates that ranking firm importance by motif counts and degree centrality gives the same results. Figure 5.12 shows the motif count ranking of each firm plotted against its ranking according to its degree centrality on a log-log scale. For both *X0X* and *XXX*, the degree centrality of firms is related to the number of motifs they appear in: the higher the degree centrality, the higher the motif count of a firm. There is a direct relationship between degree centrality and motif counts. Therefore motif counts do not have to be calculated to determine firm importance; calculating degree centrality scores will suffice.

Firm rankings according to number of freight trips

The number of direct freight trips to and from firms are also used to weigh the importance of firms in motif instances. When a firm receives or makes many deliveries, it is considered to have a big logistics impact, and is therefore an important actor in the network. This analysis determines whether considering the different frequencies of freight movements to and from a firm influences the importance of the firm.



(a) *X0X* motifs.



(b) *XXX* motifs.

Figure 5.12: The relationship between ranking firms using motif counts and degree centrality scores.

The importance of firms can also be studied by ranking them according to the minimum, maximum, and total direct freight trips to and from the firm. Therefore firms are weighed according to their minimum, maximum, and total edge weight. The purpose of this analysis is to determine whether the number of freight trips to and from a firm have an influence on the firm rankings. This may be useful for future studies where edge weights are not available or are difficult to determine.

Consider the *XXX* motif with weighted edges in Figure 5.13. Each firm receives freight trips from two different firms, and makes freight trips towards these two firms. There are a different number of freight trips made to and from each firm, and therefore each edge is weighted differently. Each firm in the motif is iterated over and the following is determined:

1. The minimum edge weight connected to this firm in this motif instance.
2. The maximum edge weight connected to this firm in this motif instance.
3. The total edge weight connected to this firm in this motif instance.

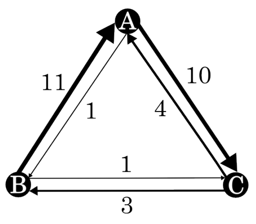


Figure 5.13: An *XXX* motif with edge weights included.

For example, firm `A` has a minimum score of 1, a maximum score of 11, and a total score of 26, while firm `B` has a minimum score of 1, a maximum score of 11, and a total score of 16. This is repeated for all motif instances and each firm score is incremented according to the minimum, maximum, and total edge weight values.

It is expected that the minimum, maximum, and total edge weight rankings will not differ for *X0X* firms, since these motifs have relatively similar edge weights. However, the absolute difference in *XXX* motif edge weights differ a lot, therefore the rankings according to minimum, maximum, and total edge weights would probably differ. These suppositions are confirmed in the results shown in Table 5.3. The unweighted rankings (column 'Unw.') is the ranking of firms according to motif counts, shown previously in Table 5.2.

Table 5.3: Facility rankings according to edge weights.

Ranking	***X0X* motifs**				***XXX* motifs**			
	Min.	Max.	Tot.	Unw.	Min.	Max.	Tot.	Unw.
1	1309	2314	2314	2314	1351	1351	1351	1351
2	2314	1309	1309	1309	1309	1309	1309	1309
3	1351	1351	1351	1351	1478	899	899	630
4	1478	1478	1478	630	899	1716	1716	1478
5	899	899	899	899	1716	1478	1478	62
6	1716	1716	1716	1478	1695	2314	2314	899
7	630	630	630	1636	630	630	630	269
8	1636	1636	1636	62	2314	1695	1695	2141
9	1645	1645	1645	1645	62	62	62	1636
10	1695	1695	1695	1695	738	286	286	2314

For *X0X* motifs, using minimum, maximum, and total edge weights to rank firms has little effect on the order of the top 10 firms. The order of the unweighted ranking is very similar to that of the weighted rankings. Therefore the same rankings will be obtained by both methods.

The minimum, maximum, and total weighted rankings for *XXX* motifs contain the same top nine firms in a slightly different order. The maximum and total rankings are exactly the same, with the minimum ranking differing slightly. When comparing the weighted rankings to the unweighted rankings, only seven of the top ten weighted firms appear in the unweighted ranking. Therefore the two different methods of ranking *XXX* firms result in slightly different rankings.

For *X0X* both *XXX* motif instances, it does not matter which method of ranking firms is used: whether firms are weighed according to the freight trips to and from them, or whether they are weighed according to the number of motifs they occur in, the rankings remain very similar.

5.3 Conclusion

Some motif instances are much more important that others as a result of the large amount of freight traffic that occur in them. The more freight trips that occur in a motif instance, the stronger the relationships between the firms. The strength of relationships in *X0X* motif instances are very similar, and all firms in *X0X* motif instances contribute relatively equally to the amount of freight traffic between them. *XXX* motif instances, on the other hand, have large differences in relationship strengths. Therefore *XXX* motif instances usually have two dominant firms (between which the strong relationship exists), and the remaining weaker firm. As the relationship strength increases in *XXX* motif instances, the distance between the firms decrease, but the same cannot be said for *X0X* motif instances.

The ranking of firms can be done in a number of different ways: by counting the number of motif instances a firm appears in, by weighing the importance of a firm according to the number of freight trips to and from the firm, or by simply using degree centrality. These three methods result in very similar firm rankings, since motif counts are shown to be directly related to motif counts. The contribution of these results are discussed in the next chapter, along with recommendations for future work.

Chapter 6

Conclusion

In this dissertation the presence of 3-node motifs in a complex network of freight movements was investigated. A motif is a subgraph that occurs significantly more in the original network than it does on average in random networks with the same degree distribution. The research questions posed in this dissertations were as follows:

1. *Can statistically significant motifs be identified in a complex network of freight movements?*
2. *Which firms frequent these patterns of interconnections?*

This dissertations showed that it is possible to identify statistically significant motifs in a complex network of freight movements. The most significant relationships that occur between three firms are two and three reciprocal relationships. The firms that often frequent these significant patterns were also identified, and a brewery and truck stop frequently appeared in the results. This has implications for future work, which is discussed in Section 6.2.

6.1 Contribution

To address the research questions, the optimal clustering parameter configuration was determined, and this is the first contribution of this dissertation. The second contribution is the implementation of a number of algorithms in order to use ISMAGS to identify motifs in the freight complex network. These *Java* classes can be used in future work where ISMAGS is used to identify motifs. A number of insights were gained through the results and indicated different avenues of future research. These contributions are discussed further in the following sections.

6.1.1 Determining optimal clustering parameters

The clustering parameters have an impact on how firms are identified as nodes in the complex network, therefore it is crucial to use clustering parameters that are visually accurate, but also result in a complete complex network. Therefore the multi-objective optimisation performed in this dissertation is of critical importance. From this analysis, it was determined that using either clustering parameter configuration $\gamma = (10, 10)$ or $\gamma = (20, 20)$ results in an optimal complex network. Therefore either of these clustering parameter configurations can be used in future work.

6.1.2 Algorithms

A number of algorithms, listed in Table 6.1, were implemented in *Java* to use ISMAGS in this dissertation. The matching algorithm was implemented in Algorithm 2 to generate random networks with the same degree sequence as the freight complex network. The edge lists of the freight complex network and all random networks were split into one- and bi-directional edges by implementing Algorithm 12. The motif specifications, where an A represents a one-directional edge, and an X represents a bi-directional edge, were generated from adjacency matrices using Algorithm 11.

When ISMAGS is called, it only takes one subgraph as input, and can only look for this subgraph in the run. Therefore Algorithm 13 was implemented so that ISMAGS could be called iteratively for a list of subgraphs.

Before subgraph counts were determined, some of the motif instances had to be filtered. ISMAGS handles a '0' edge in a motif specification as an edge that may or may not exist, but in this dissertation a '0' edge must indicate the absence of an edge. Therefore Algorithm 14 was implemented to ensure that all subgraph instances with a '0' edge in their motif specification did not contain an edge in the '0' position.

Table 6.1: Algorithms implemented in *Java* for this dissertation.

Algorithm	**Purpose**
Algorithm 2	Generate random networks using the matching algorithm (used in Chapter 5).
Algorithm 11	Generate custom motif specifications from adjacency matrices.
Algorithm 12	Split network edges into one- and bi-directional edges and write to separate files.
Algorithm 13	Iteratively call ISMAGS for different motif specifications and different networks.
Algorithm 14	Ensure that motif instances containing a '0' in their motif specifications do not have an edge in the '0' position.
Algorithm 15	Assign edge weights to all motif instances.

6.1.3 Significance of results

It was determined that three firms with two ($X0X$) or three (XXX) reciprocal relationships between them occur significantly more in the freight complex network than on average in the random networks. There are more one-directional than bi-directional edges in the freight complex network, but despite this fact, the bi-directional edges form the significant patterns. Therefore these relationships should be focused on when studying firm relationships in the freight complex network.

If a research study must be conducted on groups of firms that have similar relationship strengths between them, firms in $X0X$ motif instances should be studied. On the other hand, if the research must focus on groups of firms with one dominant relationship, firms in XXX motif instances should be studied.

$X0X$ firms that have strong relationships with each other, but are located far from each other, can be identified from the results of this dissertation. These firms incur a lot of transport costs which may be avoided if dedicated supply chain improvements are implemented.

The central firms that were identified using motif counts are extremely important to the health of the entire complex network. If new policies or regulations are being considered, these firms should be included in the discussions, because they will be impacted greatly by any major decisions. The central firms have immense power in the network, and can get the remaining firms on board regarding the policies or regulations. The central firms can also be included in the decision-making process when it comes to new policies and regulations. They can represent the interests of the firms in the network, and bring a different perspective to discussions.

Central firms in the network can disseminate information fast through the network since they are connected to many other firms. It was found that central firms also form part of central motifs. These relationships can be exploited by disseminating new information through these central firms, who will spread the information through their relationships with other firms.

6.2 Future work

The results in this dissertation are by no means the silver bullet when it comes to analysing motifs in a supply chain context. It has revealed further areas of interest that can prove useful for supply chain researchers.

6.2.1 Imposing edge weight criteria

According to the definition of the freight complex network, an edge is created when a direct freight trip occurs between two firms. Since the purpose of the freight trip is unknown, some of the infrequent trips may have occurred without an actual relationship existing between the two firms. Also, if the edge weight in one direction is 500, and the reciprocal edge only has an edge weight of 10, should this edge really be considered as bi-directional? Future work can therefore address the effect that different definitions of bi-directional edges have on the resulting network. For example, bi-directional edges must have a minimum absolute difference of 10, otherwise it is considered one-directional.

6.2.2 Studying the relationships between firm rankings and node centrality scores

The results show that there is a relationship between motif counts and degree centrality scores of firms. In the proposed future work, this relationship can be studied rigorously and systematically. There may also be relationships with other centrality scores which can also be investigated. This may have a big impact on how network analyses are performed in a supply chain context in subsequent studies.

6.2.3 Determining the industries to which firms belong

Firms in motif instance can be grouped into different industry classes. If these industries are determined, the *X0X* and *XXX* motif instances can be grouped further according to the industries they form part of, and analyses can be performed to see if some motif instances behave differently when they form part of different industries.

6.2.4 Determining commodity flow

The reciprocal relationships between firms in motifs can be studied in much greater detail. The commodity flow between firms can be determined when relationships are reciprocated. One would expect a large number of empty returning freight trips, and this could indicate areas of improvement in supply chains, since empty truck trips are a wasteful expenditure. It is also possible that there is commodity flow in both directions between firms in motifs, and that these commodities could differ greatly from each other. The data collected on commodity flow can then be used in freight models, contributing significantly to the literature surrounding freight modeling.

6.2.5 Comparing the supply chains of *X0X* and *XXX* firms

The supply chain structure of firms in similar motifs can be studied to determine if there is any overlap. For example, do firms in *X0X* motifs use third party logistics companies to transport their goods, while firms in *XXX* motifs use their own fleets? Or do firms in *X0X* use larger freight vehicles, while firms in *XXX* use smaller freight vehicles? To answer these question will require a great deal of surveying, since there are thousands of firms that form part of the *X0X* and *XXX* motif instances. To limit the scope of this study, the most central *X0X* and *XXX* firms, as identified by the results of this dissertation, can be used as subjects for this proposed study.

Bibliography

Barabási, A.-L. (2007). The architecture of complexity. *Control Systems, IEEE*, 27(4):33–42.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

Battini, D., Persona, A., and Allesina, S. (2007). Towards a use of network analysis: quantifying the complexity of Supply Chain Networks. *International Journal of Electronic Customer Relationship Management*, 1(1):75–90.

Bellamy, M. and Basole, R. (2013). Network analysis of supply chain systems: A systematic review and future research. *Systems Engineering*, 16(2):235–249.

Berka, T. (2012). The Generalized Feed-forward Loop Motif: Definition, Detection and Statistical Significance. In *Proceedings of the* 3*rd* *International Conference on Computational Systems-Biology and Bioinformatics*, volume 11 of *Procedia Computer Science*, pages 75 – 87.

Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., and Hwang, D.-U. (2006). Complex networks: Structure and dynamics. *Physics Reports*, 424(4–5):175–308.

Borgatti, S. P. and Li, X. (2009). On social network analysis in a supply chain context. *Journal of Supply Chain Management*, 45(2):5–22.

Breitkreutz, A., Choi, H., Sharom, J. R., Boucher, L., Neduva, V., Larsen, B., Lin, Z.-Y., Breitkreutz, B.-J., Stark, C., Liu, G., et al. (2010). A global protein kinase and phosphatase interaction network in yeast. *Science*, 328(5981):1043–1046.

Castro, N. C. and Azevedo, P. J. (2012). Significant motifs in time series. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(1):35–53.

Chen, L., Qu, X., Cao, M., Zhou, Y., Li, W., Liang, B., Li, W., He, W., Feng, C., Jia, X., and He, Y. (2013). Identification of breast cancer patients based on human signaling network motifs. *Scientific reports*, 3:3368. doi: 10.1038/srep03368.

Choi, T. Y., Dooley, K. J., and Rungtusanatham, M. (2001). Supply networks and complex adaptive systems: Control versus emergence. *Journal of Operations Management*, 19:351–366.

Choi, T. Y. and Hong, Y. (2002). Unveiling the structure of supply networks: Case studies in Honda, Acura, and DaimlerChrysler. *Journal of Operations Management*, 20:469–493.

Clauset, A., Shalizi, C. R., and Newman, M. E. (2009). Power-law distributions in empirical data. *SIAM review*, 51(4):661–703.

Conway, D. (2015). Social network analysis in R. New York City R User Group Meetup Presentation. Supporting code available online at `https://github.com/drewconway/ZIA/tree/master/R/SNA%20in%20R%20Talk%20-%20Supporting%20Code`. (Accessed 8 August).

Csardi, G. and Nepusz, T. (2006). The *igraph* software package for complex network research. *InterJournal*, Complex Systems:1695. Available online at `http://igraph.org`. (Accessed 20 May 2015).

Demeyer, S., Michoel, T., Fostier, J., Audenaert, P., Pickavet, M., and Demeester, P. (2013). The Index-Based Subgraph Matching Algorithm (ISMA): Fast Subgraph Enumeration in Large Networks Using Optimized Search Trees. *PloS one*, 8(4):e61183.

Fortin, S. (1996). The graph isomorphism problem. Technical Report 96-20, University of Alberta, Edomonton, Alberta, Canada.

Frankenstein, Z., Alon, U., Cohen, I. R., et al. (2006). The immune-body cytokine network defines a social architecture of cell interactions. *Biology Direct*, 1(32):1–15. Available online at `http://www.biology-direct.com/content/1/1/32`. (Accessed 30 August 2015).

Free Software Foundation (2015). GCC, the GNU Compiler Collection. Available online at `https://gcc.gnu.org/`. (Accessed 1 June).

Free Software Foundation (2015). *The GNU Make Manual*, 0.73 edition. Available online at `https://www.gnu.org/software/make/manual/`. (Accessed 10 August).

Galaskiewicz, J. (2011). Studying supply chains from a social network perspective. *Journal of Supply Chain Management*, 47(1):4–8.

Houbraken, M. (2015). ISMAGS. *GitHub* repository. Available online at `https://github.com/mhoubraken/ISMAGS`. (Accessed 20 February).

Houbraken, M., Demeyer, S., Michoel, T., Audenaert, P., Colle, D., and Pickavet, M. (2013). The Index-Based Subgraph Matching Algorithm with General Symmetries (ISMAGS): Exploiting symmetry for faster subgraph enumeration. *PloS one*, 9(5):e97896.

Iturria-Medina, Y. (2013). Anatomical brain networks on the prediction of abnormal brain states. *Brain Connectivity*, 3(1):1–21.

Jackups, R., Cheng, S., and Liang, J. (2006). Sequence Motifs and Antimotifs in β-Barrel Membrane Proteins from a Genome-Wide Analysis: The Ala-Tyr Dichotomy and Chaperone Binding Motifs. *Journal of Molecular Biology*, 363(2):611–623.

Joubert, J. W. and Axhausen, K. W. (2011). Inferring commercial vehicle activities in Gauteng, South Africa. *Journal of Transport Geography*, 19:115–124.

Joubert, J. W. and Axhausen, K. W. (2013). A complex network approach to understand commercial vehicle movement. *Transportation*, 40(3):729–750.

Joubert, J. W. and Meintjes, S. (2015a). Computational considerations in building inter-firm networks. *Transportation*, 42(5):857–878.

Joubert, J. W. and Meintjes, S. (2015b). Repeatability & reproducibility: Implications of using GPS data for freight activity chains. *Transportation Research Part B: Methodological*, 76:81–92.

Kashtan, N., Itzkovitz, S., Milo, R., and Alon, U. (2004). Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758.

Katebi, H., Sakallah, K. A., and Markov, I. L. (2012). Graph Symmetry Detection and Canonical Labeling: Differences and Synergies. In Voronkov, A., editor, *Turing-100. The Alan Turing Centenary*, volume 10 of *EPIC*, pages 181–195, Manchester, UK.

Kavurucu, Y. (2015). A comparative study on network motif discovery algorithms. *International Journal of Data Mining and Bioinformatics*, 11(2):180–204.

Kim, Y., Choi, T. Y., Yan, T., and Dooley, K. (2011). Structural investigation of supply networks: A social network analysis approach. *Journal of Operations Management*, 29(3):194–211.

Liu, X. (2011). *Applications of complex network science*. PhD thesis, The Hong Kong Polytechnic University. Available online from `http://hdl.handle.net/10397/5704`. (Accessed 3 July 2015).

Mari, S. I., Lee, Y. H., and Memon, M. S. (2015). Complex network theory-based approach for designing resilient supply chain networks. *International Journal of Logistics Systems and Management*, 21(3):365–384.

McKay, B. D. (1981). Practical graph isomorphism. *Congressus Numerantium*, 30:45–87.

Meintjes, S. (2013). Multi-objective optimisation of a commercial vehicle complex network. Mini-Dissertation (BEng), University of Pretoria, South Africa. Available online at `http://hdl.handle.net/2263/33499`. (Accessed 15 August).

Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., Sheffer, M., and Alon, U. (2004). Superfamilies of evolved and designed networks. *Science*, 303(5663):1538–1542.

Milo, R., Kashtan, N., Itzkovitz, S., Newman, M., and Alon, U. (2014). On the uniform generation of random graphs with prescribed degree sequences. Available online at `http://0-arxiv.org.innopac.up.ac.za/pdf/cond-mat/0312028.pdf`. (Accessed 6 June 2015).

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. (2002). Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827.

Min, H. and Zhou, G. (2002). Supply chain modeling: Past, present and future. *Computers & Industrial Engineering*, 43(1):231–249.

Nethercote, N. and Seward, J. (2007). *Valgrind*: A framework for heavyweight dynamic binary instrumentation. In *ACM Sigplan notices*, volume 42, pages 89–100. ACM.

Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM Review*, 45:167–256.

Newman, M. E. J. (2008). The mathematics of networks. *The New Palgrave Encyclopedia of Economics*, 2:1–12.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Ribeiro, P. (2011). *Efficient and scalable algorithms for network motifs discovery*. PhD thesis, Universidade do Porto. Available online at `http://www.dcc.fc.up.pt/~pribeiro/publications/pribeiro-phd-2011.pdf`. (Accessed 22 April 2015).

Ribeiro, P. (2015.). *gtrieScanner* - Quick Discovery of Network Motifs. Available online at `http://www.dcc.fc.up.pt/gtries/`. (Accessed 5 March).

Ribeiro, P. and Silva, F. (2010). G-tries: An efficient data structure for discovering network motifs. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1559–1566. ACM.

Schilling, M. A. and Phelps, C. C. (2007). Interfirm collaboration networks: The impact of large-scale network structure on firm innovation. *Management Science*, 53(7):1113–1126.

Schreiber, F. and Schwöbbermeyer, H. (2005). Frequency concepts and pattern detection for the analysis of motifs in networks. In *Transactions on Computational Systems Biology III*, volume 3737 of *Lecture Notes in Computer Science*, pages 89–104. Springer Berlin Heidelberg.

Slota, G. M. and Madduri, K. (2014). Complex network analysis using parallel approximate motif counting. In *Parallel and Distributed Processing Symposium, 2014 IEEE* 28*th* *International*, pages 405–414. IEEE.

Strogatz, S. H. (2001). Exploring complex networks. *Nature*, 410(6825):268–276.

Tran, N., Mohan, S., Xu, Z., and Huang, C.-H. (2015). Current innovations and future challenges of network motif detection. *Briefings in Bioinformatics*, 16(3):497–525.

University of Koblenz-Landau (2015). The Koblenz Network Collection. Available online at `http://konect.uni-koblenz.de/`. (Accessed 12 August).

Wasserman, S. and Faust, K. (1994). *Social network analysis: Methods and applications*, volume 8. Cambridge University Press.

Wen, L., Shi, Y., and Wang, L. (2013). The modeling and simulation of supply chain based on directed complex network. *Journal of Information & Computational Science*, 10(18):6085–6092.

Wernicke, S. (2006). Efficient detection of network motifs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 3(4):347–359.

Wernicke, S. and Rasche, F. (2006). *Fanmod*: a tool for fast network motif detection. *Bioinformatics*, 22(9):1152–1153.

Wong, E., Baur, B., Quader, S., and Huang, C.-H. (2012). Biological network motif detection: Principles and practice. *Briefings in Bioinformatics*, 13(2):202–215.

Wu, G., Harrigan, M., and Cunningham, P. (2012). Classifying Wikipedia articles using network motif counts and ratios. In *Proceedings of the* 8*th* *Annual International Symposium on Wikis and Open Collaboration*, page 12, Linz, Austria. ACM.

Zhou, C., Frankowski, D., Ludford, P., Shekhar, S., and Terveen, L. (2004). Discovering Personal Gazetteers: An Interactive Clustering Approach. In *Proceedings of the* 12*th* *Annual ACM International Workshop on Geographic Information Systems*, pages 266–273. ACM.

Appendix A

Discovering three-node subgraphs

Motif discovery is illustrated in Chapter 1 using a network and three-node subgraphs. This appendix illustrates all possible subgraph instances that can be found in the network in Figure 1.2.

Figure A.1 shows all possible $i = 1$ subgraph instances. The six $i = 2$ subgraph instances are shown in Figure A.2. There are seven $i = 3$ subgraph instances, which are shown in Figure A.3. The two $i = 4$ subgraph instances are shown in Figure A.4. There are no $i = 5$ subgraph instances in this network, but two $i = 6$ subgraphs are shown in Figure A.5. The last two subgraph instances are that of $i = 7$, and are shown in Figure A.6. There are no other subgraph instances that can be identified in this network.

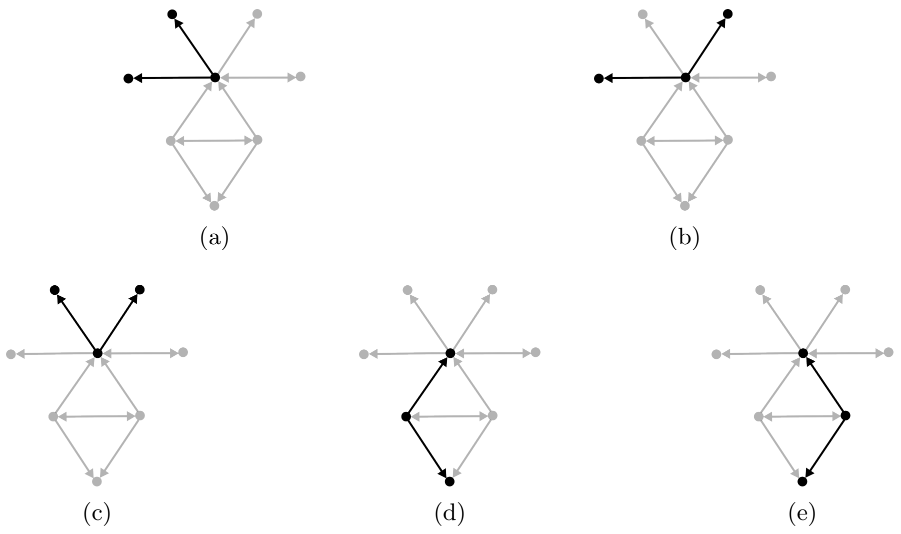


Figure A.1: Identifying all $i = 1$ subgraph instances in a network.

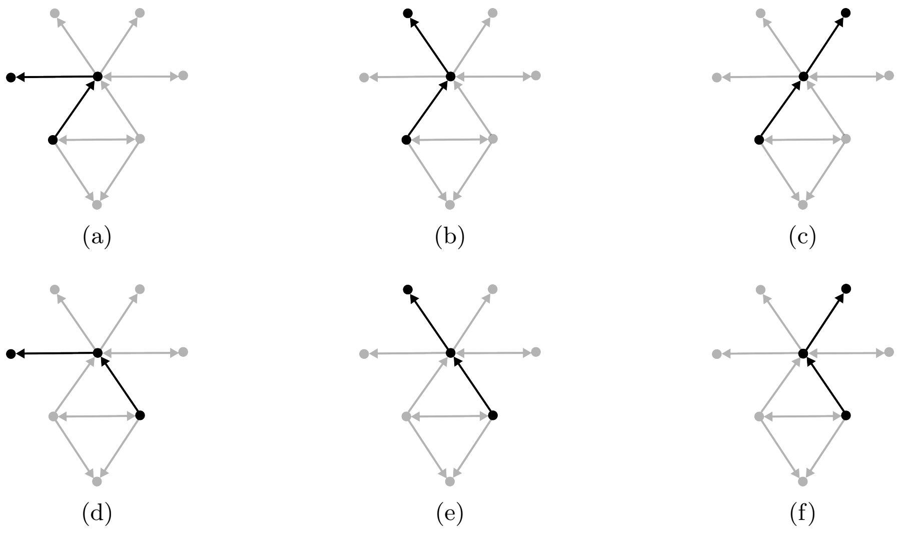


Figure A.2: Identifying all $i = 2$ subgraph instances in a network.

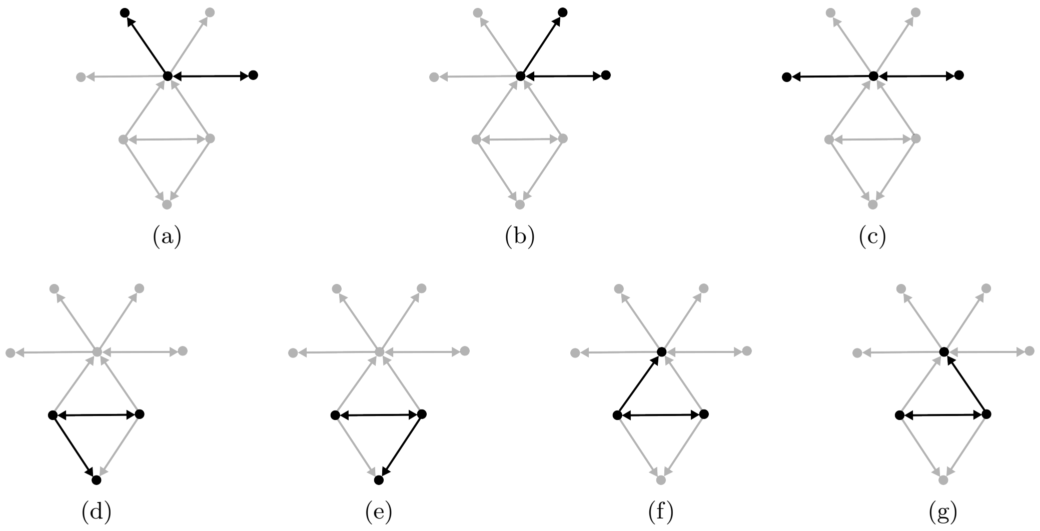


Figure A.3: Identifying all $i = 3$ subgraph instances in a network.

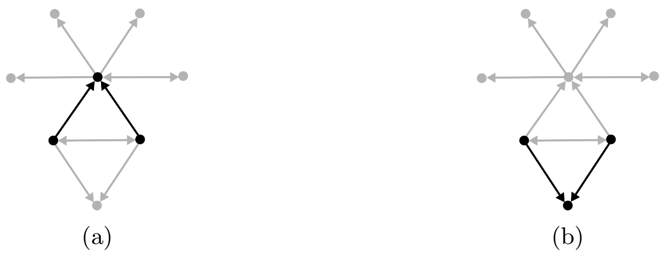


Figure A.4: Identifying all $i = 4$ subgraph instances in a network.

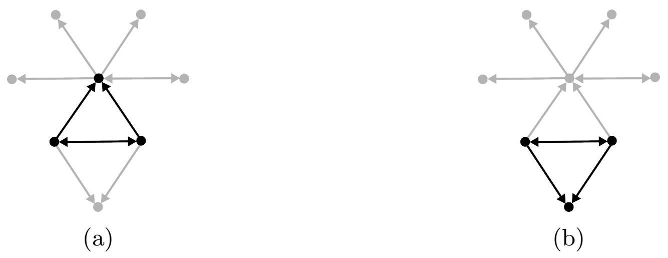


Figure A.5: Identifying all $i = 6$ subgraph instances in a network.

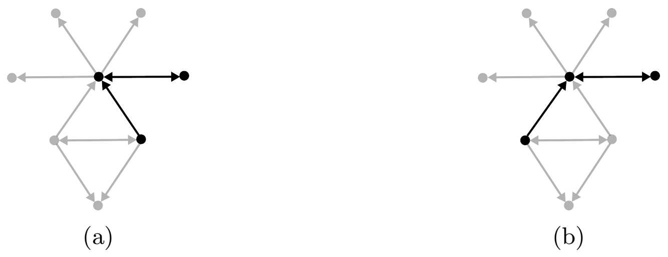


Figure A.6: Identifying all $i = 7$ subgraph instances in a network.

Appendix B

Comparing G-Tries and ISMAGS quantitatively

The following figures show the results from Chapter 3 on separate y-axes. Figure B.1 shows that each algorithm follows the same trend, irrespective of the size of the subgraphs. The memory of ISMAGS remains relatively similar as the size of the networks increase, while the memory usage for G-Tries increases as the size of the networks increases. Figure B.2 shows that the run times for ISMAGS when 3- and 4-node subgraphs are identified is much higher than the run time of G-Tries. The run time of ISMAGS is much closer to that of G-Tries when it searches for 5-node subgraphs. The number of subgraph instances are identified in Figure B.3, and both ISMAGS and G-Tries has an upward trend as the size of the network increases.

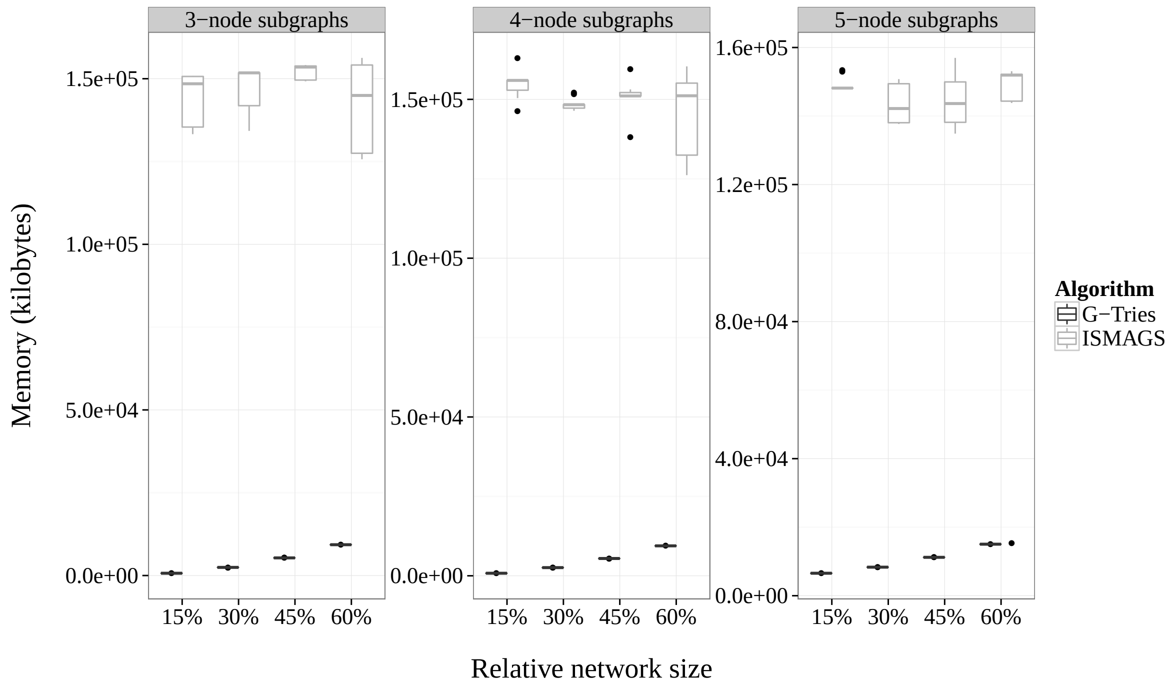


Figure B.1: Comparing the memory usage on separate y-axes.

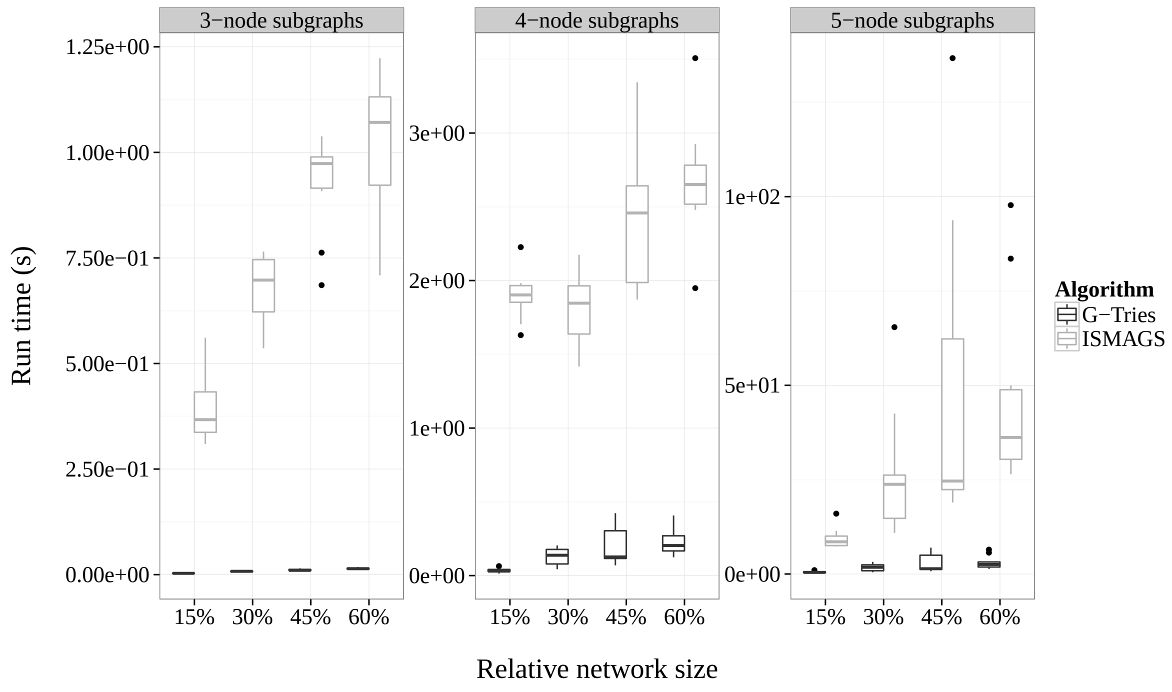


Figure B.2: Comparing the run time on separate y-axes.

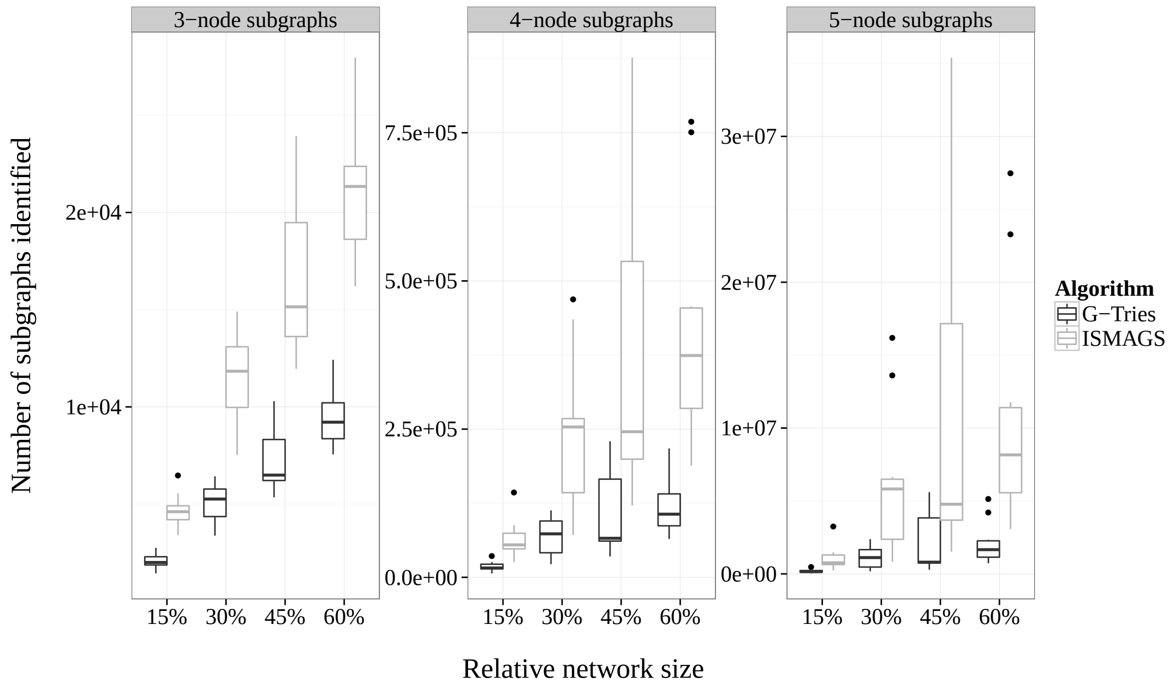


Figure B.3: Comparing the number of subgraph instances identified on separate y-axes.