Evaluation of Visual Network Algorithms on Historical Documents

Khairunnisa Ibrahim

A thesis submitted

In fulfilment of the requirements for the degree of Master of Science

(Computer Science)

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK
2020

DECLARATION

I declare that the work in this thesis was carried out in accordance with the regulations of Universiti Malaysia Sarawak. Except where due acknowledgements have been made, the work is that of the author alone. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

……

Signature

Name: Khairunnisa binti Ibrahim

Matric No.: 15020326

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak

Date: Jan 2020

ACKNOWLEDGEMENTS

Foremost, I would like to offer this endeavour to Allah SWT for the wisdom He bestowed upon me, the strength and also good health to finish my research.

Immeasurable appreciations and deepest gratitude are extended to the following persons for the help and support in making this study possible:

Associate Professor Dr. Bali Ranaivo-Malançon and Dr Stephanie Chua Hui Li, my supervisors, for their commitments and efforts in guiding and assisting me in doing the research. Thank you very much for your unreserved supports and advices.

Associate Professor Dr Cheah Yu-N and Mr Terrin Lim, my co-supervisors, it has been a great honour to work under your co-supervisions.

Ibrahim and Zaharah, my parents, thank you for everything. Without a doubt, I would have been nothing without you both.

Friends, family and everyone that have been taking part in supporting me to completing my research, for the kind endless helps and generous advises.

ABSTRACT

Visual network is a special type of graph representing real life systems where the vertices are accompanied with attributes and the edges represent relationships between them. Network visualisation facilitate comprehension of texts, especially for historical documents, where important events, facts and relationships are recorded. This study proposed a generic framework to perform evaluation of visual network algorithms to find the best network representation of a document. The framework suggests to evaluate both graph layout and clustering algorithm in order to produce a good network. The framework has been used to evaluate three graph layout and three graph clustering algorithms on the historical SAGA dataset. The evaluation found that FA2 algorithm when combined with MC algorithm produce the best network representation for SAGA. The evaluation also demonstrates that the scores given by evaluation metrics can disagree with one another as they each are invented based on different opinions on how to indicate a good cluster. The proposed framework is also applied on Biotext and dBPedia dataset and the findings implied that the performance of an algorithm, be it a layout or a clustering algorithm, actually depends on the structure of the document itself. Therefore, for a new document, evaluation of algorithms is ineluctable. The study also proposed a simple but reliable cluster evaluation metric called NPL-C metric. The metric is able to rate both the internal and external structure of clusters in a given network by using the concept of average path length and conductance.

Keywords: Generic evaluation framework, reliable cluster evaluation metric, network visualisation, historical network, graph algorithm evaluation

Penilaian Algoritma Rangkaian Visual ke atas Dokumen Bersejarah

ABSTRAK

Rangkaian visual adalah sejenis graf khas yang menggambarkan sistem dunia realiti dimana nod disertakan dengan atribut dan pertalian antara nod ditandakan oleh garis penghubung nod. Rangkaian visual membantu pemahaman teks, khasnya bagi dokumen bersejarah, di mana tercatatnya maklumat penting. Kajian ini mencadangkan sebuah rangka kerja yang umum untuk membuat penilaian ke atas algoritma-algoritma rangkaian demi mendapatkan bentuk rangkaian terbaik untuk sesebuah dokumen. Algoritma susun atur rangkaian dan pengelompokan rangkaian dicadangkan untuk dinilai untuk mendapatkan rangkaian terbaik. Rangka kerja tersebut telah digunakan untuk menilai tiga algoritma susun atur dan tiga algoritma pengelompokan ke atas dokumen bersejarah SAGA. Kajian mendapati gabungan algoritma FA2 dan MC menghasilkan rangkaian yang terbaik. Kajian juga mendapati skor metrik-metrik penilaian boleh bercanggah kerana dicipta atas konsep berbeza.. Rangka kerja tersebut juga digunakan ke atas dua lagi dokumen iaitu Biotext dan dBPedia dan kajian mendapati prestasi sesebuah algoritma, samada susun atur atau pengelompokan, sebenarnya bergantung kepada struktur dokumen itu sendiri. Maka, untuk dokumen yang baru, membuat penilaian yang baru tidak dapat dielakkan. Kajian juga mencadangkan satu metrik penilai bagi algoritma pengelompokan yang diberi nama metrik NPL-C. Metrik tersebut boleh menilai keadaan struktur dalaman dan luaran kelompok-kelompok yang ada dalam sesebuah rangkaian dengan menggunakan konsep purata jarak laluan dan pengaliran.

Kata kunci: *Rangka kerja penilaian umum, metrik penilai pengelompokan yang berkesan, rangkaian visual, rangkaian bersejarah, penilaian algoritma rangkaian*

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

LIST OF ABBREVIATIONS

CA	Clustering Algorithm
CPL	Cluster Path Length
CSV	Comma – Separated Values
CW	Chinese Whispers
FA2	Force Atlas 2
GATE	General Architecture for Text Engineering
GN	Girvan Newman
HU	Yifan Hu
LA	Layout Algorithm
MC	Markov Clustering
NEs	Named Entities
NPL-C	Normalised Path Length – Conductance
OO	Open ORD
RC	Research Contributions
RO	Research Objectives
RP	Research Problems
RQ	Research Questions
SAGA	Sarawak Gazette
Q	Modularity
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation

CHAPTER 1

INTRODUCTION

1.1 Overview

The evolution of technologies has made more and more textual data available on the World Wide Web. Historical documents are among the textual data that are continuously digitised and made available online (e.g., the project Visualising Historical Networks[1]). Information visualisation has provided various techniques for converting textual data into visual representation. As brought up by Hearst, the human interpretation is highly receptive to images, and visual representations convey information more rapidly and effectively compared to textual data (Hearst, 2009).

The historical documents, especially, are often comprised of long texts, which made reading them conventionally line by line a hard work. Finding relationships between two historical subjects in long texts are often time consuming, unless the subjects appeared frequently together in the texts. It will be harder if the relationships are explicitly written. Only after a full comprehension of the texts can readers conclude such relationships. If the elements are extracted from the textual data, a visual network of these elements can assist human readers. It helps deliver vast historical information in just one presentation. Apart from that, the analysis of the visual network can reveal hidden patterns, spot trends, intrinsic communities, etc.

However, there are many aspects to be considered when converting a document into a network. One of the aspects is the comprehensibility of the network. This is because not

[1] http://www.fas.harvard.edu/~histecon/visualising/

all networks generated from a document is guaranteed to be decipherable. An example of an undecipherable network is given in Figure 1.1.

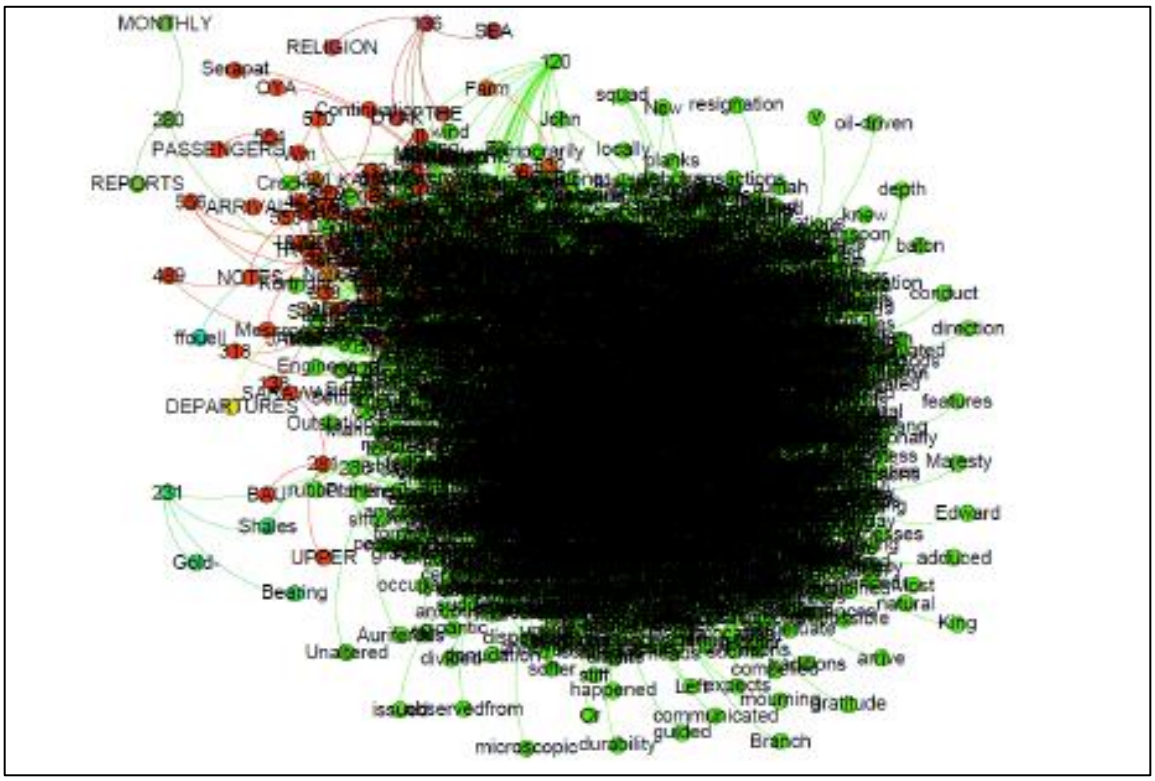


Figure 1.1: An Undecipherable Network

Apart from that, there is also a myriad of visual network algorithms that can be applied to the network. Different network algorithms perform differently on the network. As highlighted by Dunne et al. (2015), there can be variations of incomprehensible and confusing layouts generated by visual network algorithms for a given network. Thus, selecting the right algorithm to be applied for a specific data is a challenge.

Other aspects that need to be considered is on how to improve the information delivered by the network. For example, by clustering the entities in the network, entities that

are closely-related in the documents can be discovered. However, there are also many visual network algorithms that can perform clustering but use different mechanisms.

1.2 Research Problems

Historical documents hold valuable information about people, locations, events, etc. They contain our past and the legacies of the past in the present. For example, if the interest is in understanding world wars, then the World War I Document Archive[2] makes available many historical documents such as conventions, treaties, diaries, memorials, etc. But if the interest is on a specific area, for example, Sarawak (a Malaysian state on Borneo), then the Sarawak State Library[3] has made available for online reading the digitised version of Sarawak Gazette (39 years collections starting from 1907). The size of historical documents is huge making human analysis difficult. Therefore, there is a need to overcome the problem by providing a means in exploring these historical documents. Network visualisation can be this means. The following are the research problems (RP) in the study, each numbered in sequence accordingly:

RP1: The representations of a document into visual networks are sometimes hard to be understood.

Historical documents contain texts that are made of a sequence of words and punctuations. The problem is that all content words (nouns, verbs, adjectives, and adverbs) in the historical documents are meaningful. Content words convey meaning, whereas function words such as ‘and’, ‘but’ and ‘under’ are grammatical words. Taking into

[2] https://wwi.lib.byu.edu/index.php/Main_Page
[3] http://www.pustaka-sarawak.com/gazette/home.php

consideration all content words for the construction of a visual network will bring forth several problems such as the large size of the visual network, the limitation of the computer screen for the display of the network, the varieties of relations that need to be defined and managed, etc. Such choice always results in a network that is unreadable. Besides that, there are so many algorithms to be chosen from which can also affect the appearance of the network generated. Not all algorithms are guaranteed to be able to produce an understandable network for a document.

RP2: Evaluation of network algorithms usually focused on evaluating a single category of algorithm, not to find the best representation for a single document.

There are several literatures which evaluate the effect of applying visual network algorithms on visual networks. However, all of the literature focus on either the evaluation of layout or clustering algorithms on the network at a time. In order to find the best network representation of a single document, the two categories of algorithms could have been applied and evaluated at the same time. The layout algorithms function to produce a readable layout for the network. On the other hand, the clustering algorithms function to produce meaningful groups for the contents of the network. Therefore, a framework needs to be elaborated for the evaluation of both categories of visual network algorithms in order to find the best visual network representation for a document.

RP3: Evaluation of network algorithms is difficult with so many evaluation metrics available.

There is currently a wide variety of evaluation metrics that can be used for the evaluation of visual networks. Choosing which metrics to be used in the evaluation process

require attention. This is because different evaluation metrics measured different aspects of the network. Apart from that, certain metrics can only be used under certain condition. An example would be the scalability metric which is only applicable for large networks. Besides that, several metrics have been used repeatedly without deeper investigation on their reliability, especially cluster evaluation metrics.

1.3 Research Questions

Four research questions (RQ) emerge from the three problems stated in the previous section, each numbered in sequence accordingly:

RQ1: What kinds of information are to be extracted from the historical texts to constitute the data of a visual network?

RQ2: How to find the best representation of a network for a single document?

RQ3: Can the selected algorithms for the historical documents perform the same for other datasets too?

RQ4: Is all the many evaluation metrics used reliable?

1.4 Research Objectives

The study aims to perform evaluation on some visual network algorithms on historical documents to find the best representation of the documents in the form of visual

network. The following are the list of research objectives (RO) of the study, each numbered in sequence accordingly:

RO1: To identify the appropriate historical textual units and their relations to be the data of visual networks.

RO2: To propose a framework to do evaluations of both graph layout and clustering algorithms on a single document to find its best network representation.

RO3: To find the algorithms that can produce better historical visualisation and study their results on other datasets.

RO4: To create a simpler and reliable evaluation metric to simplify evaluation task.

Figure 1.2 shows the overview of the study where the relations between the research problems, research question and research objectives of the study are illustrated.

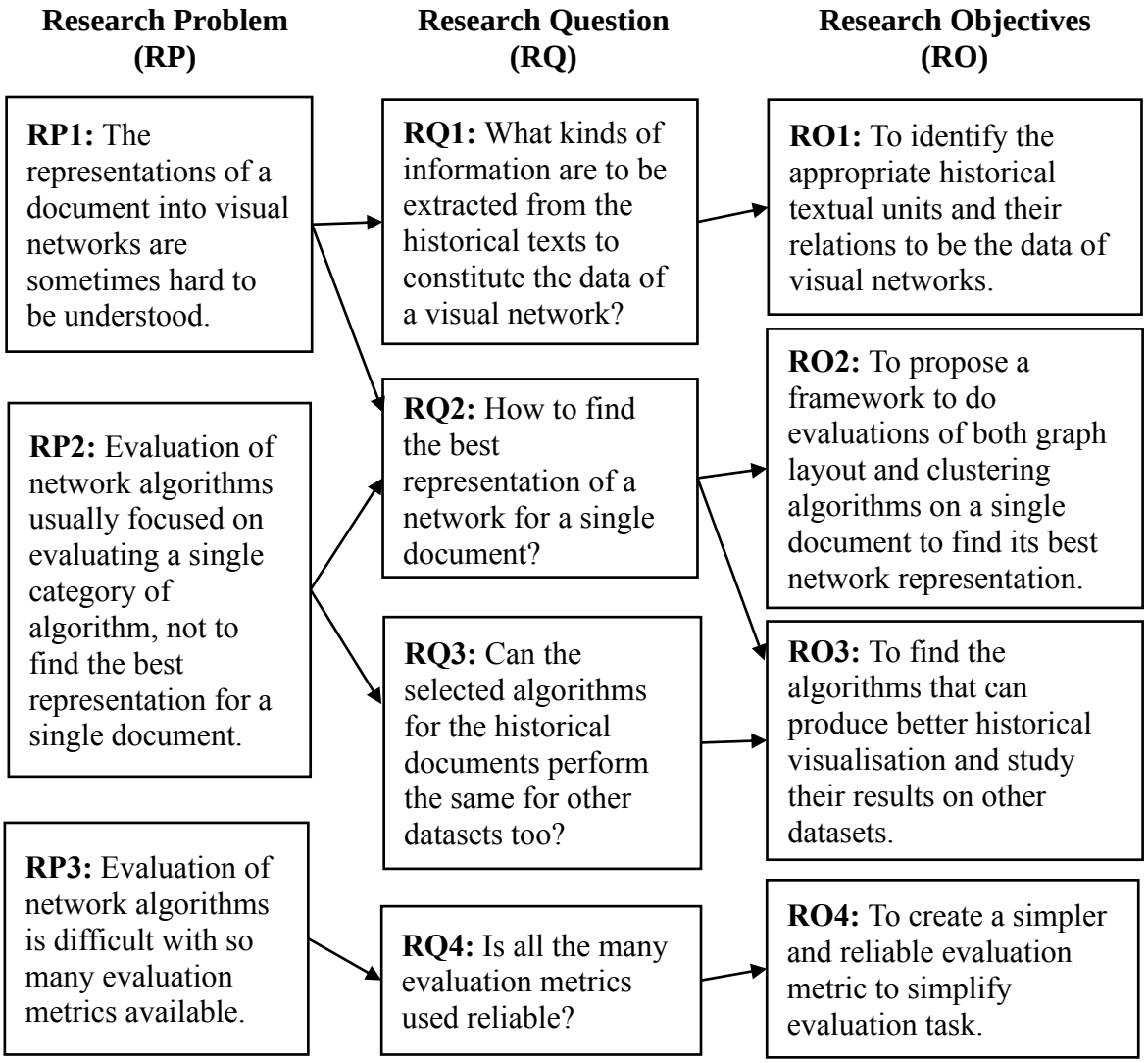


Figure 1.2: Overview of Study

1.5 Research Scope

This study is confined to the following topics:

- Historical documents used in the study is Sarawak Gazette (specifically 1904 January 1st edition);
- Not all visual network algorithms are included in the evaluation but only a few from the category of graph layout algorithms and graph clustering algorithms;
- Not all evaluation metrics for graph algorithms are used but only a few.

1.6 Research Contributions and Deliverables

The research contributions (RC) and deliverables of this study are listed as follows, each numbered in sequence accordingly:

RC1: The study produced a representation of Sarawak Gazette in the form of visual network which has both high-quality layout and clustering.

RC2: The study produced a generic framework to perform evaluation of both graph layout and clustering algorithms to find a good network representation of documents.

RC3: The study proposed a simpler metric to evaluate clustering algorithms.

1.7 Organisation of the Thesis

The dissertation contains five chapters including this introductory chapter. This chapter introduces a brief explanation on the research problems, research questions, research objectives, research scopes and research contributions of the thesis. Chapter 2 starts with the introduction to the historical documents used in this study, which is Sarawak Gazette. The introduction is then followed by the introduction to graph theory, followed by a review on existing visual network algorithms and network visualisation tools, the evaluation metrics currently used in the field, and works done in presenting historical documents into visual network. Chapter 3 explicates the methodology of the study. The proposed general framework to evaluate visual network algorithms is further explained and a simple but reliable cluster evaluation metric is introduced in the chapter. Chapter 4 provides all the visual network outputs generated by the evaluated algorithms on the studied historical documents, and the statistical results obtained from using the evaluation metrics. The analysis of the results is discussed, and an additional comparison of the selected algorithms on the historical document is made on two other data of different domain from historical documents. Further observation on the results are discussed whether the selected algorithms can also perform well on other documents. Chapter 5 concludes the study and highlights the contributions. The research limitations and a number of possible future works are suggested at the end of the chapter.

CHAPTER 2

LITERATURE REVIEW

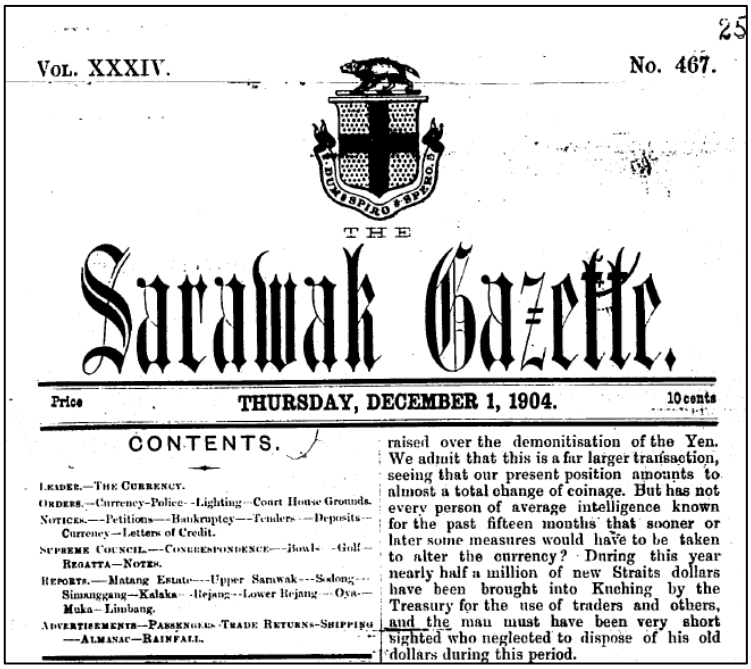
2.1 Introduction

This chapter starts with a section that contains the introduction to the historical documents used in this study. The section is then followed by an explanation of the background of graph theory and the definition of some commonly used terms in the field. The chapter continues with a review on existing visual network algorithms, to find the category of algorithms suitable to be evaluated for the study and the available network visualisation tools that fit the needs of the study. The metrics proposed in the literature to evaluate visual network algorithms are also reviewed. The chapter ends with a section containing a review on works published related to the transformation of textual data into the form of visual network.

2.2 Historical Document Used

In this study, the historical document used is the Sarawak Gazette, henceforth called SAGA. SAGA is one of the oldest newspapers published in Sarawak, a Malaysian state on the island of Borneo. Its first publication was on August 26, 1870[4], and this was during the British reign. Written in British English, the gazette was published monthly and was edited by the Rajah's Civil Service. Since the articles reported everyday life in different parts of Sarawak, it is a very rich source for exploring and understanding the history of Sarawak. Figure 2.1 shows an excerpt of a SAGA newspaper.

[4] http://www.pustaka-sarawak.com/gazette/home.php



25

VOL. XXXIV. No. 467.

THE

Sarawak Gazette.

Price THURSDAY, DECEMBER 1, 1904. 10 cents

CONTENTS.

raised over the demonitisation of the Yen. We admit that this is a far larger transaction, seeing that our present position amounts to almost a total change of coinage. But has not every person of average intelligence known for the past fifteen months that sooner or later some measures would have to be taken to alter the currency? During this year nearly half a million of new Straits dollars have been brought into Kuching by the Treasury for the use of traders and others, and the man must have been very short sighted who neglected to dispose of his old dollars during this period.

Figure 2.1: Excerpt of SAGA newspaper

For the study, we already acquired several editions of SAGA ranging from 1903 to 1909 in collection from previous researches (Wan et al., 2017). However, only a subset of SAGA will be used to perform evaluation on the algorithms in this study, specifically the edition published on 12th January 1904. The reasons why are further explained in Section 3.3. SAGA data used in this thesis are SAGA editions with annotated named entities (NEs). Annotated NEs are real-world objects found in unstructured texts which has been classified into their own predefined categories, such as persons, locations, or organisations. The annotations are done using GATE (General Architecture for Text Engineering) software. GATE is an open-source platform that integrates a named entity recognition (NER) module called ANNIE (A Nearly New Information Extraction system). This thesis does not cover the NEs annotation task, but readers can refer to the work of Wan et al. (2017) that reports the semi-automatic annotation of NEs found in SAGA. In that work, the NEs annotated

gazette is saved in GATE XML format. The GATE XML file stores information that GATE has recorded for the annotation of NEs. However, this information does not include the relations between NEs since they have not been added during the annotation stage. Therefore, to be able to generate a graph from the available information, ad-hoc relations have to be created. This will be further explained in Section 3.3.

At the moment, the current form of SAGA available to the public is in an online website[5]. In the website, SAGA is browsable by editions but to be read, the files need to be downloaded first from the website. The downloaded files are scanned images of SAGA pages in PDF. The drawback is that the texts cannot be searched for since there is no optical character recognition (OCR) applied to the document. Therefore, turning SAGA contents into a network can help give a brief insight to the documents before user can decide to download it or not.

2.3 Background of Graph Theory

In this thesis, the term 'visual network' is used to emphasise that the discussion is on network. A visual network is a special type of graph representing real life systems, where relation between real life objects are illustrated discretely through their links to each other. Therefore, introduction to visual network should start with the introduction to graph theory.

Graph study started to become an interest when Leonhard Euler, a mathematician of the 18th century in Prussia tried to solve the seven Konigsberg bridge problems (Mihalcea & Radev, 2011). All of the seven bridges are connected to an isolated region and needed to be crossed each not more than once. In his attempt to solve the problem, Euler represented them

[5] http://www.pustaka-sarawak.com/gazette/home.php

in the form of graph, thus introducing graph theory. Figure 2.2 shows how the seven bridges connecting each of the isolated region in Konigsberg, indicated as region A, B, C and D were visualised into a graph. Since then, to ease problem solving, graph became a popular method to represent objects in real life.

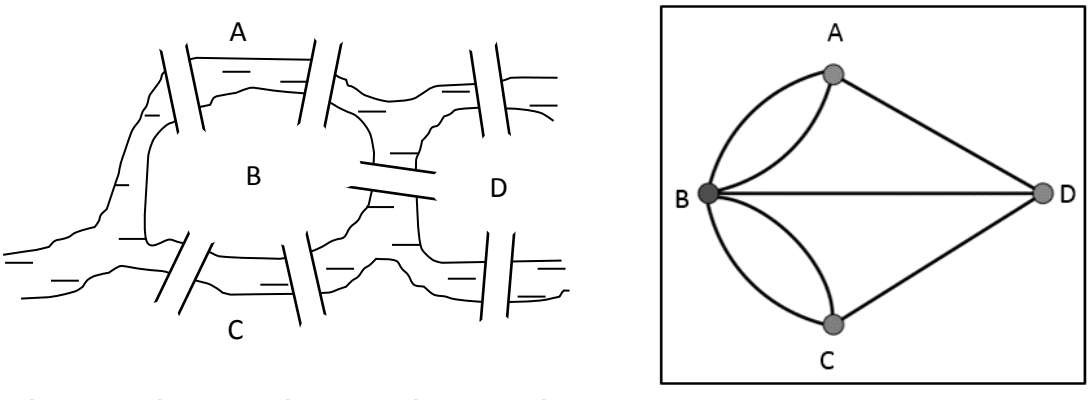


Figure 2.2: Bridges in Konigsberg (Map and Graph)

2.3.1 Some Definitions

A graph is a data structure formed by vertices (or vertices) and edges (or links, arcs or connections). It is defined as a set of $G = (V, E)$, where V is a non-empty set of **vertices** and E is a non-empty set of **edges**, such that an edge $e \in E$ joins a pair of vertices, say (u, v), where $u \in V$, $v \in V$. In this case, (u, v) are called the **end points** of e. The vertices *u* and *v* are said to be **adjacent** to each other. The edge *e* is said to be **incident** to the vertices *u* and *v*. An edge *e* is called a self-loop edge if *u* = *v*.

A **directed graph** (also known as **digraph**) has the direction of its edges defined. A directed edge is represented by a line with an arrow at the end indicating its direction. An **undirected graph** has no direction defined for its edges. An undirected edge is represented by a simple line. It can be traversed from any of its endpoints. Figure 2.3 (a) shows a directed

graph with four vertices (A, B, C and D) and five edges (AB, AC, AD, BC and CD). Figure 2.3 (b) shows an undirected version of it.

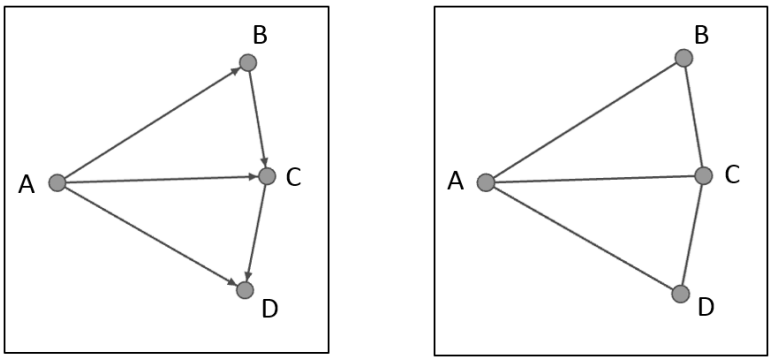


Figure 2.3: Example of Directed (a) and Undirected (b) Graph

A **subgraph** of graph G is a graph that is fully made from only a partial of graph G. For example, for the graph shown in Figure 2.3 (a), a graph that consists of vertices A, B and C along with the edges incident to them is its subgraph. A **cyclic** graph is a directed graph which has a circle in it. Vice versa, an **acyclic** graph is a directed graph which has no circle in it, for example the subgraph containing vertices A, B and C in Figure 2.3 (a).

If there exist multiple edges between any two vertices in the graph, then the graph is called a **multigraph**. The **size** of a graph corresponds to the number of vertices (denoted by $|V|$) in that graph. A **sparse graph** has $|E| << |V|2$, where $|E|$ is the number of edges. A **walk** is a sequence of alternating vertices and edges from a vertex to another. In Figure 2.3 (b), $A \rightarrow B \rightarrow C$ is a walk of length 2 from vertex A to vertex C. A walk is also called a **path** or an **open walk** if each of the vertices in it do not repeat itself. A **cycle** or a **closed walk** is a walk that starts and ends at the same vertex, such as $A \rightarrow B \rightarrow C \rightarrow A$. A graph is called a **connected graph** when any two vertices in it are connected through a path, either of long or

short length. A **strongly connected graph** is a graph that has a path between any two pair of vertices in it. A **disconnected graph** is a graph that consists of more than one separated graph.

The degree of a vertex is the number of edges incident to the vertex. In a directed graph, the degree of a vertex is of two types: the **in-degree**, which is the number of edges that point towards the vertex, and the **out-degree**, which is the number of edges that point out from the vertex. For example in Figure 2.4, vertex C is said to have a degree of three, an in-degree of two, and an outdegree of one.

In a **weighted graph** (Figure 2.4), each edge and/or each vertex is associated with a label (or weight). The weights on the edges may represent frequencies, distances, semantic relations (e.g., "is a" relation as in "a table is a furniture"), etc. In an **unweighted graph**, the edges do not have labels.

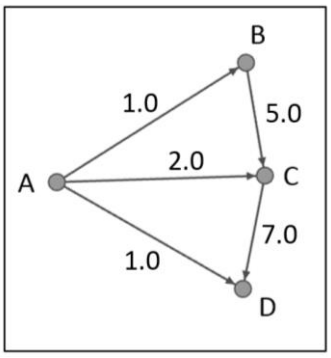


Figure 2.4: Example of a Directed Weighted Graph

The terms defined previously are the basics often used in mathematical graph problems. Throughout time, graphs have been used to represent not just mathematical problems but more real-life events including social networks, telecommunication networks, neural networks, economic networks and many more (Venturini et al., 2014). A visual

network is a term used to define graph which has real life objects labelled as its vertices, for example locations, people, months and etc. The edges in a visual network represent relations between the objects such as “went to”, “was born in” and etc depending on the type of relation harnessed from the real-life events. Figure 2.5 shows an example of a visual network representing “is a friend of” relation where the vertices represent people.

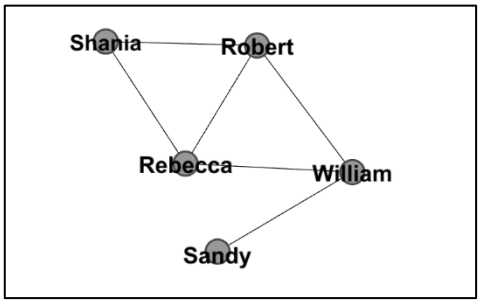


Figure 2.5: Example of a Visual Network

Visual network algorithms, or simply network algorithms, or also known as graph algorithms are algorithms used either to generate, modify, or analyse graphs such as to compare or to traverse. The discussion on the type of visual network algorithms focused in this thesis will be explained in Section 3.3.

2.3.2 Graph Representations

In computers, a graph is usually represented by an adjacency matrix or an adjacency list. Adjacency matrix, or connection matrix of a graph is a matrix with the vertices of the graph as its column and row. The matrix contains 1 (or more than 1 for weighted graphs) or 0 to indicate whether the pair of vertices are adjacent or not in the graph. Adjacency list consists of the list of vertices pair which are next to each other in a graph. For a human view,

the adjacency list is translated into a visualisation of linked vertices, and if the data visualised represent real life objects having certain attributes, it forms a visual network.

2.4 Visual Network Algorithms

Varieties of visual network algorithms have been devised to ease network visualisation task. For instance, automatic placement of vertices and edges in a network following certain rules can be done by using graph layout algorithms. The rules applied aim for the network to be drawn so that they are easy to read and understood. There are also methods developed which make use of the vertices attributes to produce a clustered view of the visual network, namely graph clustering algorithms (Gibson et al., 2014). Graph clustering algorithms improve representation of the data in the network by revealing implicit relationships between them through their clusters. Graph isomorphism algorithms, on the other hand, are invented with the aim to compare and identify exact matching graphs in terms of their number of vertices and the way their edges are connected, as in (Foggia et al., 2001). However, our focus fall on the two previous categories of visual network algorithms, namely graph layout algorithms and graph clustering algorithms because they revolve around the generation and enhancement of the network representation.

2.4.1 Graph Layout Algorithms

To be visualised, a graph needs to be drawn. In graph drawing, the graphic standard is to draw vertices using symbols like circles or boxes. Those vertices symbols are usually connected by edges which are represented using simple open curves (Di Battista et al., 1994). There are many graph drawing conventions but as stated by Di Battista et al. (1998), the

commonly used are *straight-line drawing* (the edges consist of straight line segments), *orthogonal drawing* (the edges are polygonal chains either in horizontal or vertical direction), *polyline drawing* (the edges are polygonal chains), *grid drawing* (integer coordinates are associated with the vertices, edge crossings and bends), *planar drawing* (the edges do not cross each other), and *upward and downward drawings* for directed acyclic graphs (the edges are either all vertically increasing or vertically decreasing (Tamassia, 1998)). Straight-line and orthogonal drawings are polyline drawings but with their own special rules (refer Figure 2.6).

Graph drawing algorithms are developed with the aim to construct the geometric representations of graphs (Di Battista et al., 1998). They are also known as graph layout algorithms (Michailidis, 2008). When drawing a graph for visualisation purposes, several issues need to be considered. One of the most crucial concern is the size of the graph. A graph that is too large faces possibility to not fit in the visualisation tool. Fitting the whole large graph in a screen usually ends up in vertex occlusions, with their labels unreadable as they overlap one another (Gibson et al., 2014). The interactivity of the visualisation tools in use, for example permission to zoom in and scroll might help to track down vertices and edges in a large graph. However, due to the limited resolution of the display devices or network visualisation tools, struggles can be faced in discerning them. A visual network with no edge crossings is preferable to ensure easy comprehension of the relationships portrayed but that is impossible for especially large and complex real-world visual network. Therefore, choosing the right drawing algorithm and visualisation tool matters.

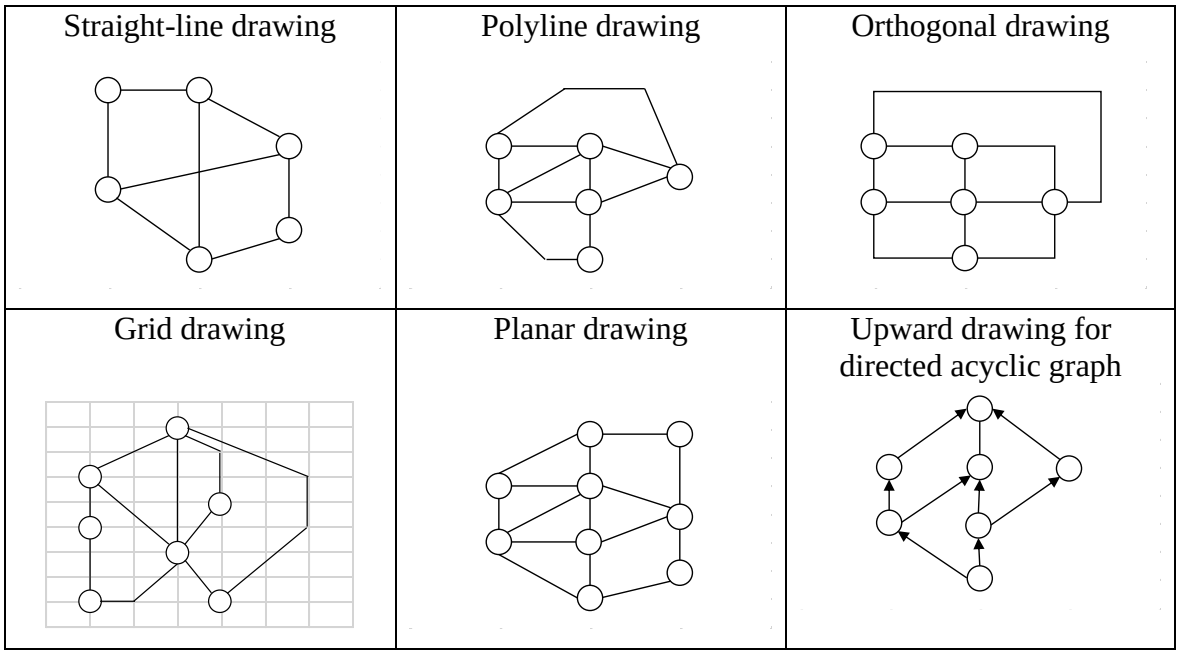


Figure 2.6: Most Commonly Used Drawing Conventions

Kruja et al. (2001) referred to the origination of graph drawing as something that are not well known (Kruja et al., 2001), even though Di Battista et al. (1998) credited the drawing flowcharts of Knuth in 1963 as the first drawing algorithm invented to be used for visualisation (Di Battista et al., 1998). Knuth's algorithm was invented to draw flowcharts to improve the documentation of computer program and has been the starter to the application of network visualisations in many areas (Gibson et al., 2014).

A single collection of data can be visualised into different drawings using different graph drawing algorithms. The literature is proposing different classifications of graph drawing algorithms. For example, Di Battista et al. (1998) suggested six approaches: topology-shape-metrics, hierarchical, visibility, augmentation, force-directed, and divide and conquer. They considered that these approaches were the most popular at that time. Between 1998 and 2018, other classifications have been proposed. For example, Gibson et al. (2014) divided graph drawing algorithms under three approaches: force-directed layouts,

the use of dimension reduction in graph layout (e.g., spectral graph drawing) and computational improvements including multi-level techniques. In 2018, six years later, Di Giacomo et al. (2018), reported three approaches (planarisation, layering, and force directed), which they considered as the most successful techniques for drawing general graphs.

In this review, the graph drawing algorithms are classified under four approaches: planarisation, orthogonal (or topology-shape-metrics), hierarchical (or layering), and force-directed.

i. Planarisation Approach

A *plane graph* is a graph that is drawn on a plane with any two of its edges fulfilling either one of these two conditions: joined at their endpoints or separated (Santanu, 2013), that is, none of the edges are crossing. According to Santanu (2013), a planar graph is a graph that is isomorphic to the plane graph. Two graphs are isomorphic if their vertices and edges show exact congruity. In a planar graph drawing, the regions separated by its non-crossing edges are called *faces* (Eades & Klein, 2015). For example, the graph in Figure 2.7 has eight faces with f_0 included as its *outer face*.

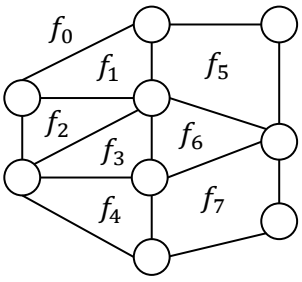


Figure 2.7: Faces in a Planar Graph

In the planarisation approach, algorithms are used to transform non-planar graphs into planar graphs. The transformation goes through two steps: (1) computation of a maximal planar subgraph *G′* of an input graph *G*, and then (2) crossing minimisation, that is, crossing edges in *G* that are not in *G'* are appended to *G'* where dummy vertices are inserted to each crossing (Di Giacomo et al., 2018), making the graph planar as illustrated in Figure 2.8.

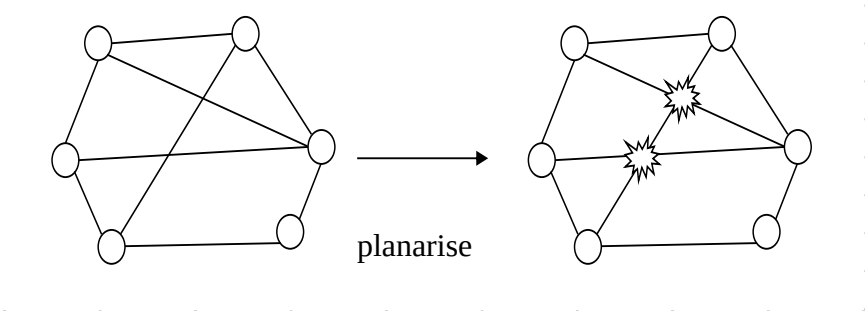


Figure 2.8: Planarisation of Graph

However, the problem of finding the largest subset of vertices that form a planar graph subgraph and minimising the edge crossing with the latter added edges are NP-hard problem, a problem that is computationally intractable. Thus, heuristics are used by planarisation algorithms for solutions (Di Giacomo et al., 2018). For example, heuristics with the aim to solve the computation of maximal planar subgraph have been proposed by Tamassia (1998), Poranen (2006) and Chimani et al. (2019). Gutwenger and Mutzel (2003), He et al. (2005) and Clancy et al. (2019) have also proposed heuristic methods to minimise crossings in graph. However, an NP-hard problem is a problem that is computationally intractable.

Planarisation approach ease the viewers to perceive edges and vertices in the graphs. However, for a large graph it is not always the case that each vertex has exactly four edges. The high number of added dummy vertices to maintain planarity alters the information

displayed by the graphs. It is not always possible to find graphs with originally very few edge-crossings, especially for real-world graphs which have countless entangled relations. Therefore, there is a high possibility that the neat representation of a document using planarisation approach does not maintain the original structure of the document content.

ii. Orthogonal Approach

Drawing graphs using edges that are rectilinear is a very effective strategy to draw graphs with good angular resolution (Duncan & Goodrich, 2013). The orthogonal approach is also known in the literature as the *topology-shape-metrics approach*. The reason is that the approach revolves around those three basic concepts (Di Battista et al., 1998): topology, shape, and metrics.

- Topology – Two graphs have the same topology (Figure 2.9) if one shows homeomorphism to the other (Eades, 2015), in other words, the graphs can be acquired from one another when they are continuously deformed as long as the edges lining the faces in their drawings remain in the same order.

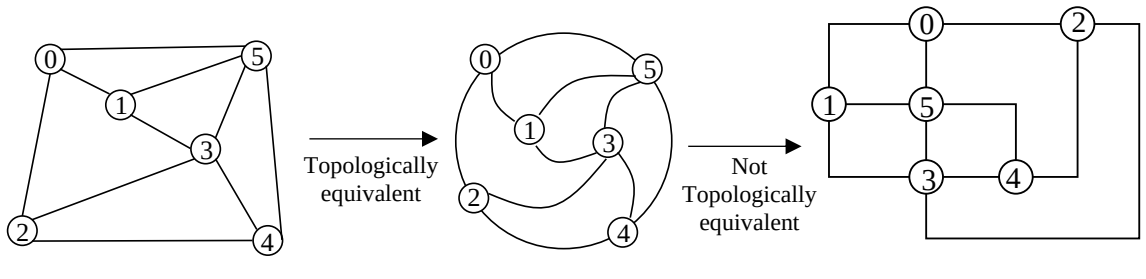


Figure 2.9: Topology Equivalence

- Shape – Two orthogonal graphs are of the same shape (Figure 2.10) if they are topologically equivalent, and they can be acquired from one another when the lengths of their orthogonal chain segments are altered without altering their angles (Di Battista et al., 1998).

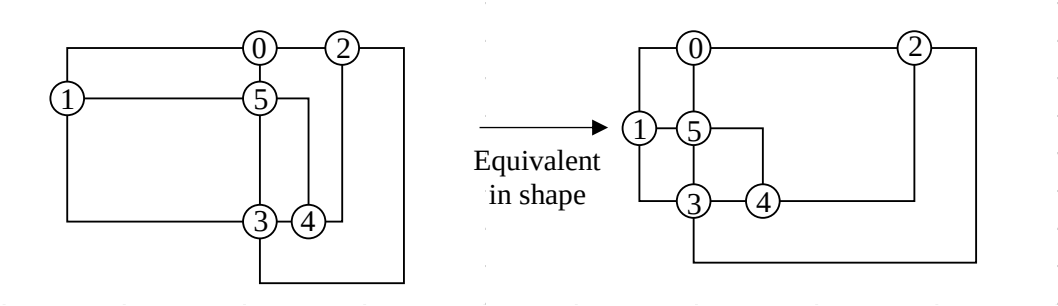


Figure 2.10: Shape Equivalence

- Metrics – Two orthogonal graphs have the same metric when they are identical (Figure 2.11) either after being rotated or translated (Di Battista et al., 1998).

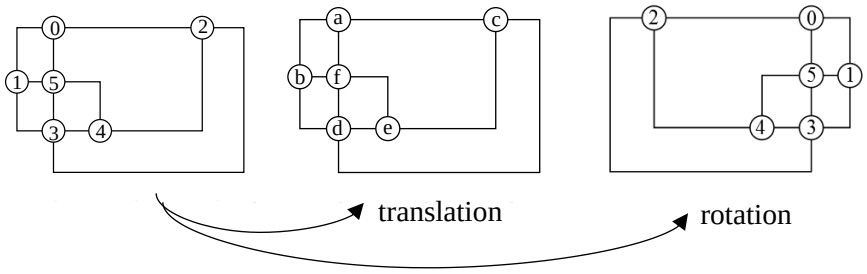


Figure 2.11: Metric Equivalence

The three basic concepts are used in the three general main steps of drawing orthogonal graphs: (1) planarisation, (2) orthogonalisation, and (3) compaction (Di Battista

et al., 1998). In the planarisation step, given an input of non-orthogonal graph *G*, a non-orthogonal graph *G'* with a good topology, which has only few edge crossings, and is topologically equivalent to *G* is first determined (refer Figure 2.12).

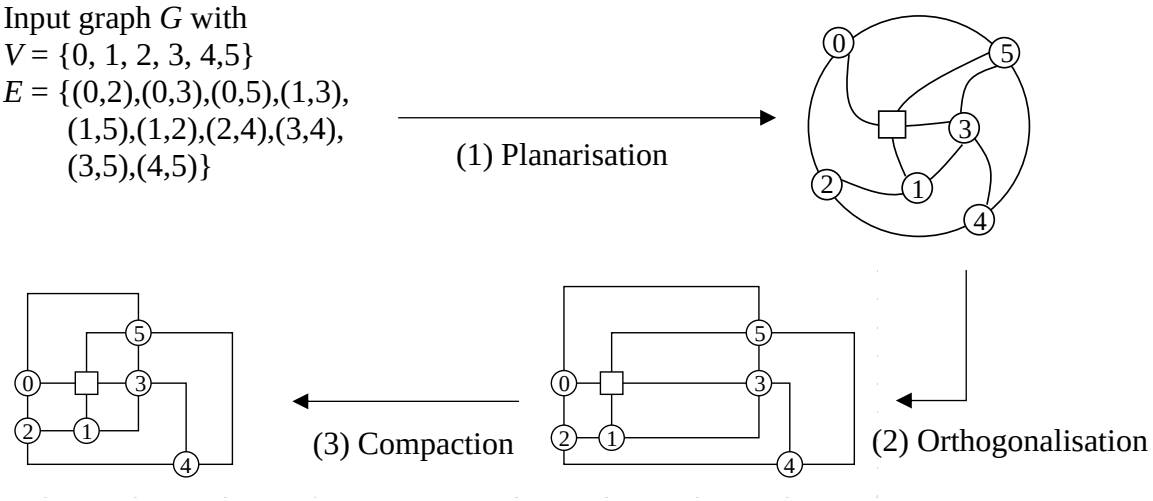


Figure 2.12: The Topology-shape-metrics Approach for Orthogonal Grid Drawings

Dummy vertices are added to maintain the planarity of the graph, represented by the square-shaped vertices. In the orthogonalisation step, *G'* is given a good orthogonal shape which has only few edge bends, while maintaining its topology. Up to this stage, the vertices are not yet associated with coordinates, but each is assigned with their own list of orthogonal bends which will be retained in the final drawing (Di Battista et al., 1998). In the compaction step, G' is given a good orthogonal grid drawing where the metrics of the vertices are determined, at the same time following the topology and shape predetermined in the two preceding steps.

Orthogonal drawings have the advantage of giving aesthetically pretty satisfying output graphs to the viewers since the edges are visually distinct from one another because of planarisation. Reading of the graph also improves as vertices can have only few edges

(Duncan & Goodrich, 2013). However, it is also one of its disadvantages when vertices can only have four degree maximum. If a graph contains vertices with degree higher than that, the orthogonal approach is not possible to be applied (Di Giacomo et al., 2018). Therefore, its application is preferred for selective data only.

iii. Hierarchical Approach

The hierarchical approach is also known as layering approach. It was initiated by Sugiyama, Tagawa, and Toda (Sugiyama et al., 1981), who proposed a method for drawing digraphs in horizontal rows and the edges directed downwards. In a hierarchical digraph, vertices are divided into *layers*, where adjacent vertices are assigned to different layers (Di Giacomo et al., 2018). If vertices are assigned to the same layer, they are placed on the same line horizontally. The hierarchical digraph shown in Figure 2.13 has three layers.

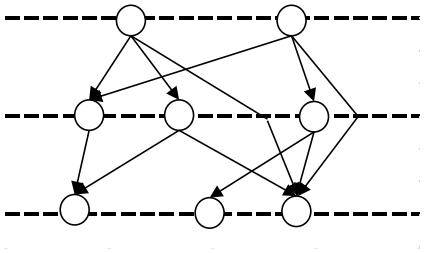


Figure 2.13: A Hierarchical Digraph with Three Layers

A hierarchical approach is comprised of three main steps (1) layer assignment, (2) crossing reduction and (3) x-coordinate assignment (Di Battista et al., 1998). In the layer assignment step, given an acyclic input digraph *G*, a layered digraph *G'* is produced, by

placing the more dominant vertices at a higher layer and vertices with the same dominance in the same layer, similar to a flow chart (refer Figure 2.14).

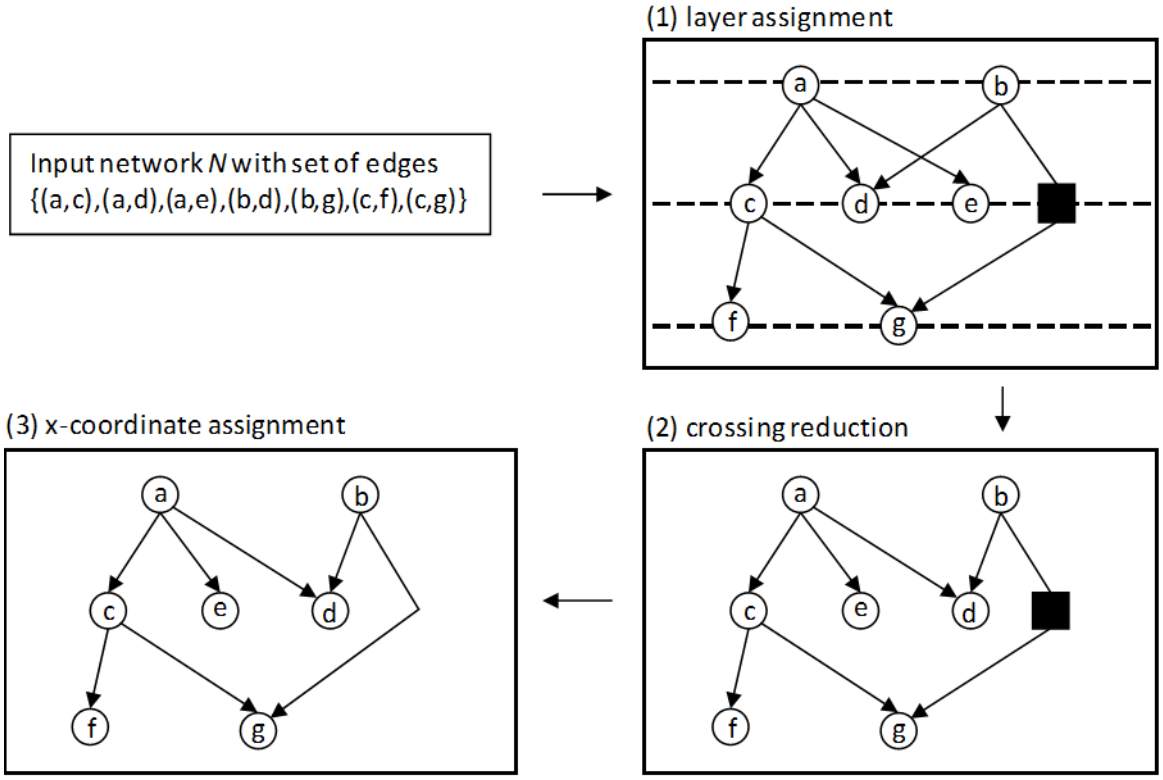


Figure 2.14: The Hierarchical Approach

For edges that cross a layer or more to connect to their endpoints, dummy vertices are inserted at each of the layer so that *G'* becomes a proper layered digraph. In the crossing reduction step, the topology of the final drawing is decided by rearranging the sequence of vertices on *G'* so that edge crossing is minimum. In the x-coordinate assignment step, the final x-coordinates of the vertices in *G'* are determined, at the same time conserving the topology and sequence of vertices produced in the previous steps. The final drawing G' is completed by aligning the dummy vertices in one straight line and then removing the vertices.

For Healy and Nikolov, the hierarchical approach proposed by Sugiyama has a few characteristics which make a graph more readable: vertices pointing to the same direction and short edges which make a graph more intelligible, vertices placed evenly thus are easily distinguished from one another, less edge crossings and straight edges which make a graph more understandable (Healy & Nikolov, 2013). However, the first step of this approach must receive an acyclic digraph input (Healy & Nikolov, 2013). If the input graph is a cyclic digraph, the direction of the cyclic edges will be reversed first so that the output remains an acyclic digraph. This means that relations in the graph may be altered to maintain the characteristics of the approach. However, the final drawing *G'* can have the relations restored to maintain the information.

iv. Force-directed Approach

Force-directed approaches, also referred to as the spring layout method (the spring embedder model – a model proposed by Eades in 1984) is considered as adjustable techniques to determine the drawings of simple undirected graphs (Kobourov, 2012). In the approach, graph is seen as a physical system (Tarawneh et al., 2011) consisting of charged atoms (represented by the vertices) linked to one another by a collection of springs (represented by the edges). In general, each vertex possesses attraction force and repulsive force (Tarawneh et al., 2011) and the total energy that a vertex holds is the aggregate of its attraction and repulsive force. The total energy possessed by the graph is the aggregate of the total energy possessed by all vertices in the graph.

Tutte's 1963 barycentric method is historically the earliest force-directed graph drawing algorithm developed (Kobourov, 2012) as it first applies the spring attraction forces

on graph edges. Eades further developed force-directed concept by applying attraction force to adjacent vertices and repulsion force on every vertex, which then became the foundation to the succeeding force-directed algorithms till date.

In Eades's algorithm concept, the vertices of an input graph *G* are first located at random places (Kobourov, 2012). The vertices are then moved gradually to form a final output graph *G'* which has minimum total energy (example of output *G'* is shown in Figure 2.15). The moving of the vertices depend on the total energy of each vertex. In the method, the attraction force f_a and the repulsive force f_r between two vertices are calculated according to the formula in Equation 2.1.

$$f_a(d) = k_1 \log\left(\frac{d}{k_2}\right) \qquad f_r(d) = \frac{k_3}{d^2} \qquad \text{Equation 2.1}$$

where k_1, k_2 and k_3 are constants, and d is the length of edge connecting the two vertices. The constants can be changed with different graphs, but the optimised value is $k_1 = 2$, $k_2 = 1$ and $k_3 = 1$. Once forces are calculated for each vertex, the vertex is moved to achieve a new total energy of "k_4 * (calculated total force on the vertex)", where k_4 is a constant that is often assigned the value 0.1 for most networks. The recalculation of the vertices' energy and the moving of the vertices are iterated for *N* number of times, where *N* is usually assigned the value of 100 to enable most networks achieve minimum total energy.

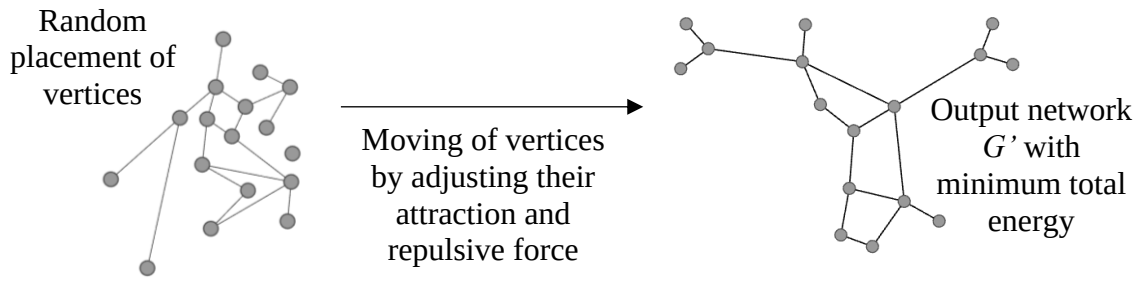


Figure 2.15: Network Generated using Force Directed Method

Since Eades (1984), force-directed concept has been applied in many visualisation algorithms (Tarawneh et al., 2011). The approaches are quite successful in producing well-balanced layouts for networks with minimum edge-crossing without any supplementary efforts (Frick et al., 1994). They also work well on small and sparse networks. However, there is a drawback in its running time when applied to big networks. This is due to the difficulty in minimising the total energy of large networks. While running, all pairs of vertices in the network needed to be visited iteratively during computation and the higher the number of iterations, the higher the quality of the output network, but unfortunately the more time it will take. In order to overcome this drawback, various improvements have been made on force-directed method which then leads to the birth of various variants of force-directed graph algorithms. Luckily, one of the utmost benefits of force-directed approaches is it is flexible to enhancement. The approaches have been reworked on and optimised many times for further improvement (Tarawneh et al., 2011) causing the approaches among the most popular methods used to generate layouts automatically.

Table 2.1 summarises all of the reviewed graph drawing methods along with their concept, advantages and disadvantages.

Table 2.1: Categories of Graph Drawing Methods

Method	Concept	Advantages	Disadvantages	Example of Algorithms
Planarity (Di Giacomo et al., 2018)	Draw graph on a grid pattern with no edge crossing.	Planar presentation of the vertices and edges are simple and clear.	Real-world graphs are not always possible drawn without edge-crossings.	Planar algorithm, Planar polyline algorithm, Planar grid algorithm.
Hierarchical (Sugiyama et al., 1981)	Draw digraph with vertices placed in layers and edges to point only downwards/upwards.	Minimum size of output graph when vertices are arranged in layer. Uniform direction of edges creates intelligible output graph.	Preferred only for unidirected input graph for uniform final output. Graph with too many ununiform edges direction confuse viewers.	Hierarchical graph algorithm.
Orthogonal (Di Battista et al., 1998)	Draw graph with each edge drawn as either horizontally or vertically 90 degrees from each other to form polygonal-shaped segments.	Can be used to solve planar graph drawing problems where edges must not bend.	It is NP-hard to minimise bends in all possible embeddings of a planar graph. Vertices can have maximum four degree only which is not always the case for real-world data.	Planar orthogonal algorithm.
Force-directed (Eades, 1984)	Draw graph where vertices and edges have forces (vertices repel and edge attract).	Many claimed that it gives pleasant drawing results. Still popular in use till date.	Finding minimum total energy takes time for large graphs.	Fruchterman-Reingold, OpenORD, Hu's multilevel layout, ForceAtlas algorithm.

In this study, the force-directed methods will be evaluated as many researchers agreed that they can produce good graph outputs (Hua et al., 2018; Kobourov, 2013; Gibson et al., 2014) and since different forms of force directed algorithms have been the most frequently used and modified graph layout algorithms (Gibson et al., 2014). The methods are still widely in use till date and there are many versions of it available, from the basic

force-directed algorithm to the modified versions, ranging from having multilevel layout to improve performance and simulated annealing phase. Apart from that, the focus of the study is on graph algorithms that can be used to display relations between NEs in historical documents. Hierarchical algorithms preferred data that is unidirected and have layers to retain their principles. Data from historical relations do not always have layers and are not always unidirected; they can be mutually-related. Planar algorithm is not evaluated as planar situation cannot be promised in historical graphs especially when one NE from historical relations can be a vertex with a degree of more than 4 at a time. Orthogonal methods are also not evaluated because historical data do not often have exactly four relations among keywords in it. Therefore, force-directed methods are chosen to be evaluated in this study. There will be three popular force-directed algorithms evaluated in this study, namely ForceAtlas2 algorithm, Hu's multilevel algorithm and OpenORD algorithm. Further details on each of the stated algorithm will be discussed in Chapter 3.

2.4.2 Graph Clustering Algorithms

Graph clustering is a type of unsupervised machine learning method. Clustering on a graph is performed to partition a collection of objects into groups called *clusters*. The objects represented by the vertices in the same cluster have similar property. Graph and the cluster structures in it are visualised simultaneously when there are relationships in the graph, whether the actual relationships contained in the graph have been acquired earlier on to be compared with the result of the clustering, or to be discovered later during the analysis of the generated clusters (Vehlow et al., 2016). Clustering algorithms intend to seize the conception that vertices are linked to many vertices of the same community but linked to just few vertices of different communities (Emmons et al., 2016). Graph clustering can be

categorised into four different approaches: superimposed methods, juxtaposed methods, embedded visualisations and vertex attributes methods (Vehlow et al., 2016).

i. Superimposed methods

In superimposed methods, the cluster structures of a graph are overlaid on the drawing of the graph. Various styles are used to indicate the different cluster regions, including using lines with the same colour to connect vertices in the same cluster, or contouring the region of the vertices of the same cluster with the same colour. Using these methods, the cluster structures may overlap with one another or separated as shown in Figure 2.16.

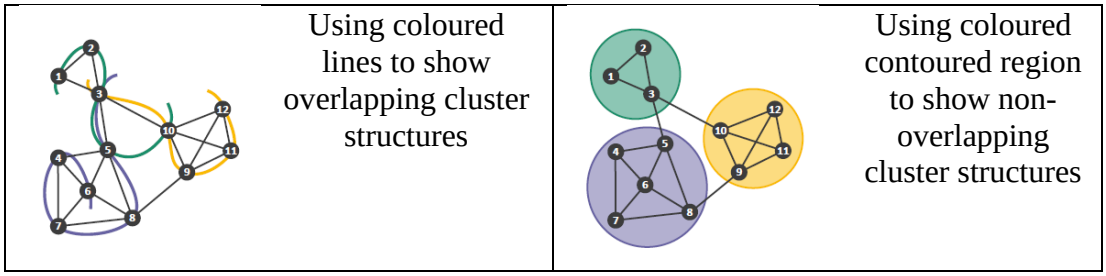


Figure 2.16: Superimposed Methods using Lines and Contouring (Vehlow et al., 2016)

ii. Juxtaposed methods

In these methods, the cluster structures in graphs are represented next to the drawing of the input graphs. The cluster structures can be either isolated from or associated with the input graph. For isolated cluster structures, they are drawn separately from the input graph but actions like clicking on a vertex can link the clicked vertex to the cluster structures either through visual connection or vertices colours. Associated clusters are drawn directly linked

to the drawing of the graph. Figure 2.17 visualised the different clustering mentioned. However, for large real-world graphs, juxtaposed methods take a lot of space. Presenting the large graph itself in a single display while ensuring the vertices remain distinguishable to the viewers is already hard, but to provide an additional space for separated cluster structures information can be even more difficult. Even if the cluster structures are associated with the graph, extra space is still required. The plenty number of elements to be included in the final drawing can reduce the size of the original graph drawing making distinction of vertices more difficult.

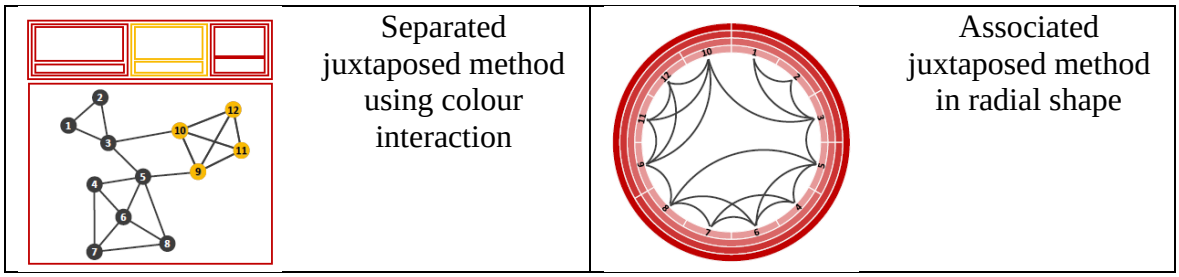


Figure 2.17: Juxtaposed Method Examples (Vehlow et al., 2016)

iii. Embedded visualisation methods

In these methods, the vertices in the input graph drawing themselves are remodelled to form clusters. The final output of these methods can appear quite similar to that of superimposed method using contouring, however these methods allow vertices of similar cluster to be combined together. Figure 2.18 shows two examples of the final outputs of embedded visualisation methods where vertices falling under the same groups are combined to represent each cluster that exists in the graphs. These methods allow a simpler version of the input graph by displaying only the main clusters found in the graph. It benefits large graphs

as the size of the graph is reduced through vertices combination. However, the final output contains less meaning as some relationships and vertices in the graph are altered to emphasise on the cluster structure information alone.

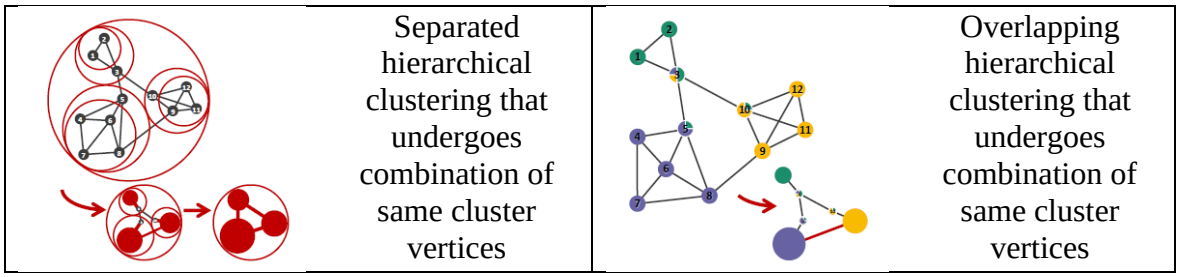


Figure 2.18: Embedded Visualisation Methods (Vehlow et al., 2016)

iv. Vertex attributes method

Vertex attributes method changes the appearance of the vertices to assign them to clusters. For example, different colours or different symbols are assigned to vertices of different groups, as shown in Figure 2.19.

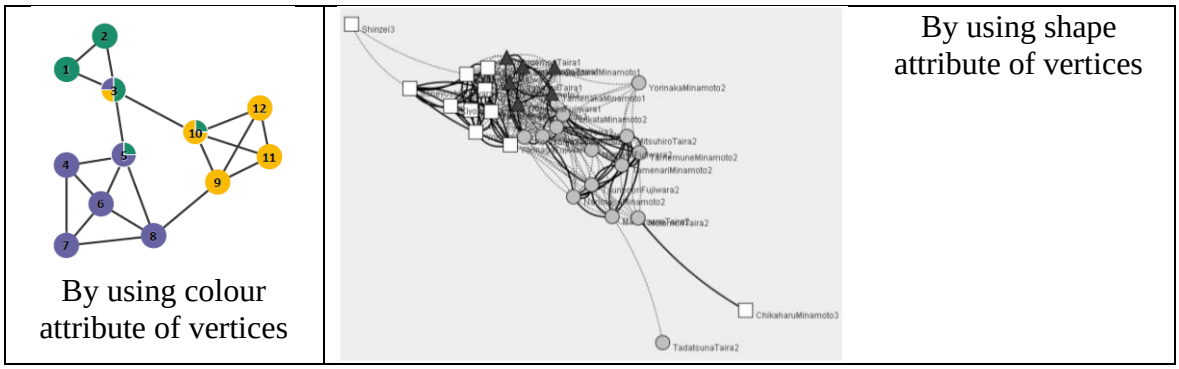


Figure 2.19: Vertex Attribute Methods by Changing Colour and Shapes of Vertices (Vehlow et al., 2016; Osaki et al., 2011)

The sole appearance of the vertices intuitively makes clusters more perceivable and easier to be understood. Apart from that, no additional space is required for clustering as only the vertices in the original drawing of the graph are reused. The three methods discussed previously are basically different from one another, however some superimposed and embedded visualisation are seen to adopt vertex attributes as an extra measure to display clusters in graphs, especially through colour differentiation. This is because this method adds to the convenience in immediately distinguishing a cluster from another. Since using colour differentiation is much simpler to be implemented yet carry an essential impact even when used in the other methods, graph clustering algorithms using vertex attribute methods, specifically through colour differentiation will be evaluated for the task of partitioning our graphs. The algorithms to be used in this study are the three commonly used graph clustering algorithms which falls under the category of vertex attribute methods using colours, specifically Markov Clustering, Girvan-Newmann and Chinese Whispers algorithm. Further details on their mechanisms will be explained in Chapter 3.

2.5 Graph Visualisation Tools

Graph visualisation tools are any software or tools used to automatically generate graphs. There are many graph clustering algorithms that exist nowadays; however, not all have been implemented into graph visualisation tools, that provide quick and easy application. To do visualisation with multiple numbers of algorithms, a suitable choice of graphing tool is required. There are many excellent options to choose from, depending on the features needed.

In order to identify the most appropriate graph visualisation tool to be used in this research work, four existing graph visualisation tools have been compared. These are

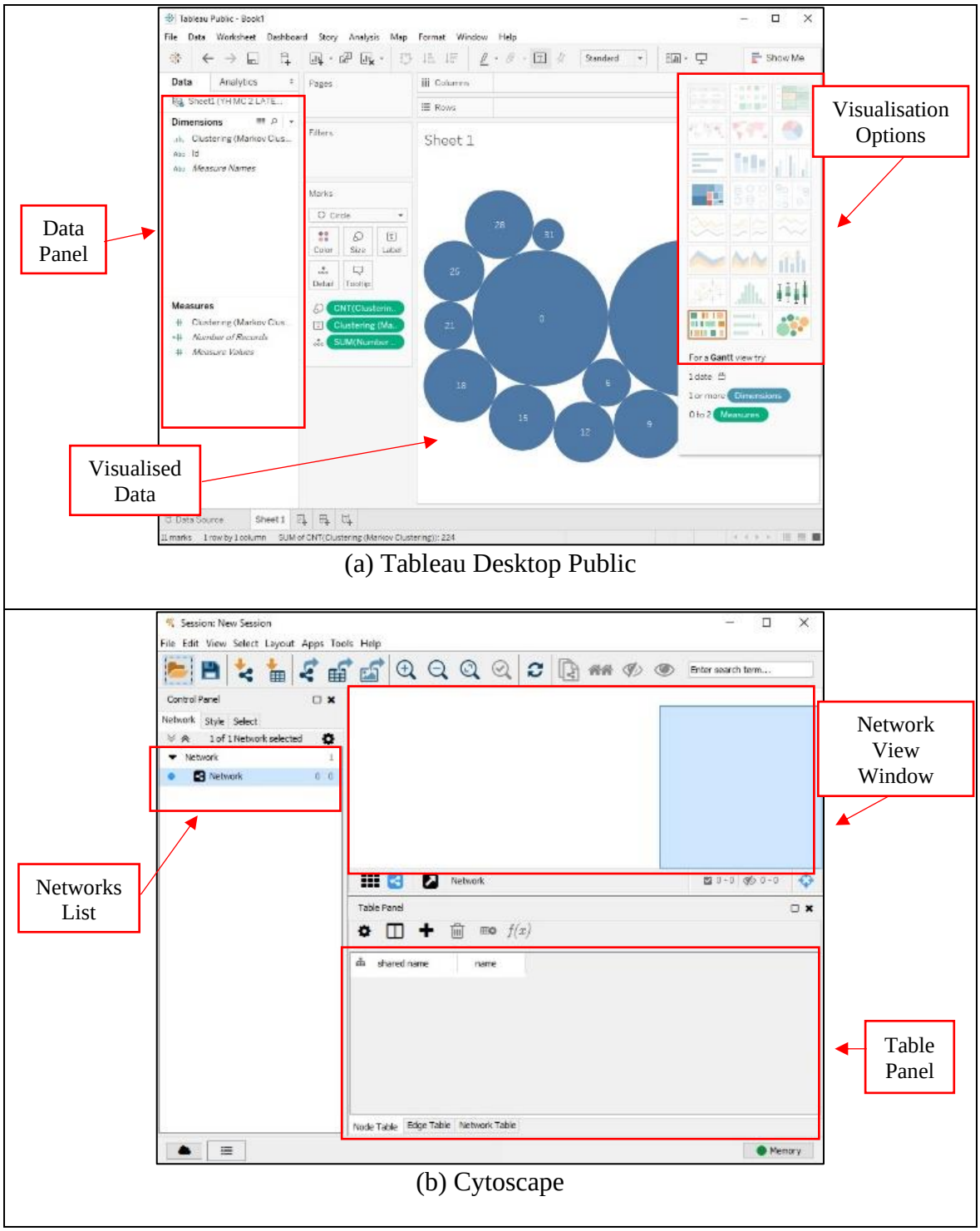
Tableau Desktop Public[6], Cytoscape[7], Gephi[8], and NodeXL[9] (refer Figure 2.20 and Figure 2.21). The selection of these graph visualisation tools is based on their popularity in graph visualisation and key features in generating graphs.

Tableau Desktop Public is a free software used to generate visualisations of data interactively and explore it. The users can arrange the presentation of their data in the form of analytical dashboard for better interpretation. The tool prohibits the saving of the generated visualisations to local machine but allows their uploads to Tableau Website where they will be available to all. No coding skills are required to use the tool. Unfortunately, there is no built-in option to generate network in Tableau Desktop Public. In order to create a network, users would have to draw their network, providing the x and y coordinates of the vertices and instruct Tableau on how to connect them.

Cytoscape is an open source Java application platform for creating complex network which is initially used to integrate biomolecular interaction network with high-throughput expression data and other molecular states into a unified conceptual framework (Shannon, et al., 2003). Its basic functionalities are to layout and query network, visually integrate network with expression profiles, and link network with databases of functional annotation.

Gephi is an open source software invented specifically for graph and network analysis. Gephi allows the process of spatializing, filtering, navigating, and clustering of the graphs in a real-time visualisation (Bastian et al., 2009).

[6] https://public.tableau.com/en-us/s/
[7] https://cytoscape.org/
[8] https://gephi.org/
[9] https://archive.codeplex.com/?p=nodexl



(a) Tableau Desktop Public

(b) Cytoscape

Figure 2.20: Graph Visualisation Tools Interface (Part 1)

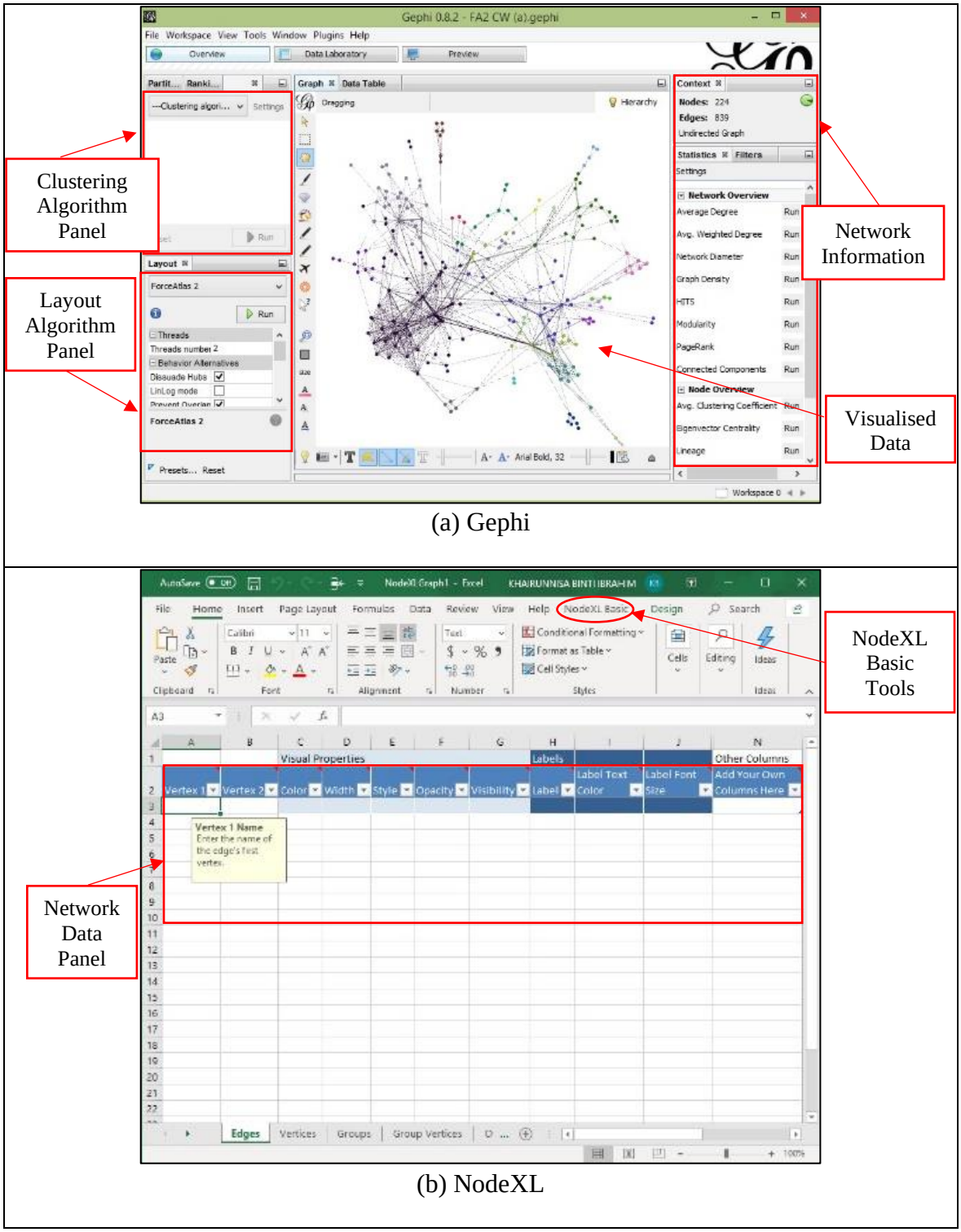


Figure 2.21: Graph Visualisation Tools Interface (Part 2)

NodeXL is an open source network visualisation and analysis software package for Microsoft Excel 2007/2010/2013/2016 (Smith et al., 2009). It adds "NodeXL Basic" tab to the original Excel spreadsheet. However, for NodeXL Basic which is a free version of NodeXL Pro, its usable features are quite restricted.

The comparisons of these tools are shown in Table 2.2.

Table 2.2: Comparison Study of Selected Graph Visualisation Tools

Graphing tool	Open Source	Dynamic real-time visualisation	Layout algorithm	Clustering algorithm	Used in research
Tableau Desktop Public	✘ (Free. Limited features compared to paid version.)	✘	✓	✘	✓
Cytoscape	✓	✘	✓	✘	✓
Gephi	✓	✓	✓	✓	✓
NodeXL	✓ (Limited features compared to paid version.)	✘	✓	✘	✓

Based on the review of the graph visualisation tools, it can be seen that although all of the reviewed tools are free software, and some are open source but Tableau Public and NodeXL have some restrictions in their provided features compared to their paid version. This is important because the essential network analysis tools to do further evaluation on the algorithms and gain insights of graphs are absent in the free version, for example, calculation of betweenness and closeness centralities of the vertices, and calculation of clustering coefficient of the vertices. Clustering algorithms which are meant to be evaluated in this research are absent in Tableau Public, Cytoscape and NodeXL Basic. It is a major drawback

for this research since clustering is intended to be applied to Sarawak Gazette's network to gain more information on the graph generated.

Therefore, the tool that fits to be used in this research is GEPHI. This graphing tool allows dynamic real-time visualisation of graphs, which allows us to distort the vertices in the graph while the tool maintains the graph itself. More importantly, GEPHI provides clustering algorithms which are not provided in most of the reviewed tools.

2.6 Evaluation Method of Visual Network Algorithms

There is a myriad of graph algorithms making the choice difficult. Hence, it is ineluctable to evaluate these algorithms through their properties and the networks they can produce. This section focuses on the evaluation metrics of two algorithm categories which are graph layout algorithms and graph clustering algorithms.

The effectiveness of a graph layout can be evaluated through several criteria; the algorithms running time (computational complexity), the size of the network that they are able to generate layout for (scalability), the aesthetics or drawing conventions that they are able to satisfy, the judgement given by the viewers by looking at the network, and other preferable attributes like clustering (Gibson et al., 2014).

2.6.1 Common Evaluation Metrics of Graph Layout and Clustering Algorithms

Two common evaluation metrics can be used to evaluate both graph layout and clustering algorithms, namely running time and scalability.

i. Running time

The running time of an algorithm signifies the total time needed by the algorithm to run until the layout or the cluster of the network is generated. Obviously, a fast execution time is more appreciated than a slow execution. Few works that have been using running time to compare graph layout algorithms can be found in (Teshome, 2013) where Teshome compared force-directed algorithm with hierarchical and MetaVis algorithm. Other works that considered the same aspect in comparing graph layout or clustering algorithms would be in Hu (2006), Hachul and Jünger (2006), Jacomy et al. (2014) and Brandes and Erlebach (2005).

ii. Scalability

Scalability is a measure whether the graph layout or clustering algorithm is capable in dealing with large graphs. Creating large graphs, especially with millions of vertices, is a critical task (Mi et al., 2016). According to (Tikhonova & Ma, 2008), a graph algorithm may produce nice layouts for graphs of several hundred vertices but may not work well as the vertices increase to several hundred thousand. The same goes to graph clustering algorithms. Scalability are evaluated by looking at the change in the time performance of the algorithm when applied to varied sizes of graphs. The graph algorithm is said to be scalable if it can handle large graphs within finite time, as some may not be successful in finishing the running at all, as in (Tikhonova & Ma, 2008), for the largest graph which they evaluated.

2.6.2 Evaluation Metrics of Graph Layout Algorithms

i. Edge Length (Aesthetics)

This metric evaluates the quality of the drawing generated by a graph layout algorithm based on the uniformity of edges in the network. Based on (Hachul & Jünger, 2005), a graph drawing with more uniform edge lengths are more readable. The uniformity of edge length is obtained through observations of the printed drawings generated by the algorithm. Comments are made on how well the drawing display the structure of each graph by keeping the modelled aesthetic criteria in mind. In their work, it is found that compared to Grid-Variant Algorithm and force-directed algorithm GRIP, Fast Multipole Multilevel Method generate graphs that are more uniform in edge length.

ii. Edge Crossing (Aesthetics)

This metric evaluates the visibility of the edges in the graph generated by the algorithm. A good layout algorithm should position the edges in a less tangled way so that the output graph is easier to be perceived by the reader. This metric has been applied in (Hachul & Jünger, 2005) when they evaluated whether there are many unnecessary edge crossings in layout algorithms for drawing large graphs where all of the algorithms involved are force-directed. In their work, the number of edge crossings are determined through observations of the graph drawing, and graphs with fewer edge crossings are preferred over graphs with many. However, since only observations were made, and no real number of edge crossings were obtained, it is hard to determine which algorithm ranked above other. Hence, in this study the number of edge crossing will be measured from its number, not only from observations.

iii. Cluttered Vertices (Aesthetics)

This metric evaluates the visibility of the vertices of the generated graph. This metric is applied by (Hu, 2006) where he considers graph with less cluttered vertices gives a graph that is more readable by the human's eye, especially for large graph with big number of vertices. In his work of evaluating layout algorithms for drawing large graph output, he compared six algorithms in total, including Grid-Variant Algorithm, Method GRIP, Fast Multi-Scale Method, Fast Multipole Multilevel Method, Algebraic Multigrid Method ACE and High-Dimensional Embedding algorithm. In their work, the layout of the network is printed out and compared via observations on the cluttering of the vertices, where less cluttered network is considered the preferable output, and no real number of cluttered vertices were obtained. However, in our study the intensity of cluttered vertices will be measured from its quantity, not only through eye observations.

Table 2.3 shows the list of evaluation metrics used on various kinds of graph algorithms previously done by researchers.

Table 2.3: List of Evaluation Method Used in Previous Works

Criteria	Description	Sources	Algorithms Used
Running Time	Time taken to create the graph	(Hu, 2006)	Yifan Hu algorithm and Walshaw's agorithm
		(Jacomy et al., 2014)	ForceAtlas2 algorithm and Yifan Hu algorithm
	Time taken for small graph and big graph are recorded for each algorithm.	(Hachul & Jünger, 2006)	Grid-Variant algorithm, Method GRIP, Fast multi-scale method (FMS), Fast multipole multilevel method (FM3), Algebraic multigrid method ACE, High-dimensional embedding (HDE)

Table 2.3 continued

Scalability	Evaluate on whether the algorithm is applicable from small to big graph	(Mueller et al., 2006)	Distributed force-directed graph layout
Aesthetic Criteria			
Edge crossing	Evaluate on whether there are many unnecessary edge crossings	(Hachul & Jünger, 2006)	Grid-Variant algorithm, Method GRIP and Fast multi-scale method (FMS), Fast multipole multilevel method (FM3), Algebraic multigrid method ACE and High-dimensional embedding (HDE)
Cluttered vertices	Evaluate on whether the vertices are cluttered or not	(Hu, 2006)	Yifan Hu algorithm and Walshaw's algorithm
Edge length	Uniformity of edge length	(Hachul & Jünger, 2006)	Grid-Variant algorithm, Method GRIP, Fast multi-scale method (FMS), Fast multipole multilevel method (FM3), Algebraic multigrid method ACE, High-dimensional embedding (HDE)

Most of the reviewed papers proposed relative evaluation criteria such as the running time of the algorithms when comparing graph layout algorithms (Hu, 2013; Jacomy et al., 2014; Hachul & Jünger, 2006). All researchers agreed that good algorithms should produce graphs in a shorter amount of time. When using this relative evaluation metric, the same graph and system specifications are required to avoid bias. The resulting running time for the algorithms may vary though according to the size of the graph used. Visual graph algorithms may give out entangled network output although they are fast. Hence, running

time alone is not adequate to verify the quality of the algorithms used. This suggests the need of additional metrics in order to evaluate the performance of a visual graph algorithm.

Another metric worth to be considered when comparing the algorithms is the number of edge crossings in the output graph (Hachul & Jünger, 2006). Edge crossing was once found as the aesthetic criteria that has the most significant impact on graph readability (Purchase, 1997). Relations detection is hard to be done if there are too many unnecessary crossings between edges .Similarly, cluttering of vertices were observed and used as one of the evaluation metrics in Yifan Hu's work (Hu, 2006) as it can affect the ability of viewers to read the graph. This metric is important to be considered as according to Dunne et al. (2015), the region in the network where vertices occlusion is high will also trouble the viewers of the network to obtain the number of vertices in a cluster.

On the other hand, uniformity of edge length metric is not applicable to force-directed algorithms as some algorithms positioned the vertices closer or further away from each other in order to show modularity in the graph. In fact, the graph data used is connected graphs with no meaning behind the position of their vertices. It does not have to be in a grid position especially when using force-directed methods. Therefore, as a conclusion, in this research, only running time, the presence of cluttered vertices and presence of edge crossings will be evaluated.

The aspect of scalability unfortunately can only be examined if the data of the graph is as big as thousand numbers of vertices. Small graphs with only hundreds of vertices are not big enough to determine the scalability potential of an algorithm. In this study, this metric will have to be omitted from being evaluated and the reason will be explained once our data has been introduced. The reason is presented in Section 3.5.1.

2.6.3 Evaluation Metrics of Graph Clustering Algorithms

It is possible to evaluate manually the results of clustering in a small graph with a small number of vertices. However, manual evaluation will become less applicable as the size of the graph increases (Almeida et al., 2011). Manual human evaluation will be hopeless too if the actual groups in the data are not known beforehand as comparison cannot be done. According to Zaidi et al. (2010), the evaluation of cluster quality can be categorised as *external*, *relative*, and *internal*. The literature mentions that in general, there are three types of metrics for evaluating the quality of clustering results: external metrics (Zaidi et al., 2010; Zhao & Karypis, 2011), relative metrics (Zaidi et al., 2010), and internal metrics (Zaidi et al., 2010; Zhao & Karypis, 2011).

External validity metrics are used when there are some predefined groups or ground truth to be compared with the output of the algorithm (Zaidi et al., 2010). The metrics are required to use a priori information of the natural clusters in the dataset. Unfortunately, in this research, external validity criteria are not an option as there are no a priori clustering structures available for the selected datasets. This fact reinforces the statement that often the external validity metric is not applicable to real world datasets (Zaidi et al., 2010).

Relative validity metrics are used when comparison is done on the clustering output of the algorithm with the output of other clustering algorithms (Zaidi et al., 2010). However, no single graph clustering algorithm can generate a benchmark for the clusters of any datasets which usually have varied cluster structures (Zaidi et al., 2010). Therefore, relative validity metrics are not applicable in this evaluation.

Internal validity metrics are used when the evaluation considers the similarity between the vertices in the same cluster and the dissimilarity between vertices in separated clusters (Zhao & Karypis, 2011). The concept of internal metrics is based on the definition

of clusters themselves – a good cluster should assemble objects that have similar or common properties. These metrics are applicable as they use only the information given to the graph clustering algorithms (Zhao & Karypis, 2011). Examples of internal validity metrics are coverage, conductance, performance, modularity and cluster path length.

i. Relative Density

Density is a metric that calculates the ratio of number of edges in a group of vertices to the total number of edges possible in that group. Density of a cluster, *d* is indicated by the following equation, $d = (e_{\text{actual}} / e_{\text{total}})$, where e_{actual} is the number of edges in the cluster and e_{total} is the number of edges possible in the cluster, calculable as shown in Equation 2.2.

$$e_{total} = \frac{number\ of\ nodes\ in\ the\ cluster \times (number\ of\ nodes\ in\ the\ cluster - 1)}{2} \qquad \text{Equation 2.2}$$

The Relative Density (Mihail et al., 2002) of a cluster calculates the ratio of the edge density inside a cluster to the sum of the edge densities inside and outside of that cluster. The Relative Density of cluster *a*, is indicated by the equation $rd_a = (d_a / d_a + d_{b\ldots z})$ where *b…z* are other clusters in the graph.

The *Final Relative Density* of a graph is the averaged sum of these individual relative densities for all clusters. Final Relative Density of graph *A* is given by the equation *rd* of graph A = $\sum rd_{a\ldots z}$ where *a…z* are every cluster in graph A. The closer the value of *rd* of graph *A* to 1, the better the clustering done on the graph.

ii. Modularity (*Q* Measures)

Modularity measures the fraction of the edges in the network that connect within-community edges minus the expected value of the same quantity in a network with the same community divisions but random connections between the vertices. If the number of within-community edges is no better than random, we will get $Q = 0$. If the value of Q approaches 1, which is the maximum value, it indicates that there is a strong community structure in the graph.

To understand this metric, we let *k* be the number of communities in a graph. Then, we define a *k x k* symmetric matrix *e* whose element e_{ij} is the fraction of all edges in the graph that link vertices in community *i* to vertices in community *j*. The trace of this matrix, $TR\ e = \Sigma_i e_{ii}$ gives the fraction of edges in the network that connect vertices in the same community. If the value of the trace is high, the division between the communities is said to be good. However, the trace itself is not sufficient though to indicate good quality of the division. For example, placing all vertices in one single community would give the maximal value of $Tr\ e = 1$ while giving no information about community structure at all.

Therefore, we let row (or column) sums $a_i = \sum_j e_{ij}$, which represent the fraction of external edges that connect to vertices in community *i*. The modularity measure, $Q = \sum (e_{ii} - a_i^2) = Tr\ e - ||\ e^2\ ||$, where $||x||$ is the sum of the elements of the matrix x (Newman & Girvan, 2004).

iii. Cluster Path Length

Cluster path length (CPL) is a metric to evaluate cluster (Zaidi et al., 2010). It is used to assign a positive score to evaluate the quality of clustering subtracted by a negative score

which is based on the inter-cluster density. The values of CPL lie in the range of [0,1] where low values indicate poor clustering and high values indicate better clustering.

The metric is composed of two components, the positive component $(M^{+}(G))$ which assigns a positive score to a cluster and a negative component $(M^{-}(G))$ which attributes a negative score to edges between clusters. The positive component is assigned on the basis of the density, compactness and mutuality of the cluster whereas the negative component is assigned on the basis of the separation of the cluster from other clusters. The final quality of a cluster is simply the sum of the two components given by Equation 2.3.

$$M(G)=M^{+}(G)-M^{-}(G) \qquad \text{Equation 2.3}$$

Let the average path length of each cluster be called Cluster Path Length. The best possible average path length for any cluster can be 1 in the case when every vertex is connected to every other vertex forming a clique. The normalised cluster path length can be given by Equation 2.4.

$$\text{Normalised cluster path length, } CPL_i = 1 / AvgPathLen_i \qquad \text{Equation 2.4}$$

where $AvgPAthLen_i$ is the average path length of the vertices in cluster i. The values lie in the range [0,1] where higher value of the normalized cluster path length indicates better quality of the cluster. The overall cluster path length is then averaged for all clusters where k is the total number of clusters. The equation for $M^{+}(G)$ is given by Equation 2.5.

$$M^{+}(G) = CPL_1 \dots k = \frac{1}{k}\sum_{i=1}^{k} CPL_i \qquad \text{Equation 2.5}$$

On the other hand, $M^{-}(G)$ is used to assign a negative score to penalize the inter-cluster edges. The value of M^{-} evaluates the separation of the two clusters. This score is calculated for each pair of clusters and is based on the number of edges that link two clusters *i* and *j* compared to the total number of edges possible between these two clusters. To calculate M^{-}, we let n_i and n_j be the number of vertices contained in clusters *i* and *j* respectively. The edge penalty for the edges present between these two cluster is given by Equation 2.6.

$$EdgePenalty_{(i,j)} = e_{ij} \ / \ (n_i * n_j) \qquad \text{Equation 2.6}$$

where e_{ij} is the number of edges present between clusters *i* and *j*. The overall Edge Penalty $(M^{-}(G))$ is the average calculated for all pair of clusters given by Equation 2.7.

$$M^{-}(G) = \frac{2}{k*(k-1)} \sum_{i=1 j=1}^{k} EdgePenalty_{(i,j)} \ where \ (i \neq j) \qquad \text{Equation 2.7}$$

The negative score sums all edge penalties over all pairs of clusters and then normalizes the value by $k(k-1)/2$ to produce an overall penalty in the range [0,1].

iv. Conductance (intercluster)

Conductance can be defined either using the intercluster edges (edges between different clusters) or the intracluster edges (edges within a cluster or with both endpoints in the cluster). In this section, we focus on the intercluster edges because relative density,

modularity and average cluster path length metrics have included the intra-cluster edges in their calculation.

The intercluster conductance of a cluster is the ratio of the number of intercluster edges to whichever is smaller between the number of intracluster edges or the number of edges that are not connected to the cluster at all (Emmons et al., 2016). Figure 2.22 illustrates the definition of the conductance metric.

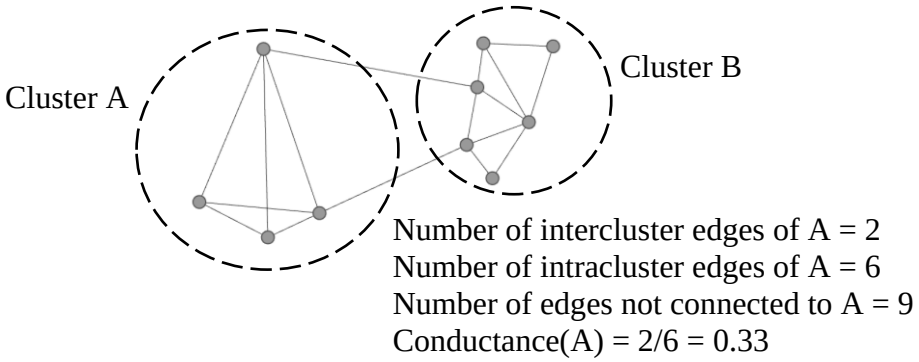


Figure 2.22: Illustration of Conductance Metric

The conductance of a cluster A in a given network can be calculated as in Equation 2.8.

$$\phi(A) = \frac{\Sigma_{i \in A, j \notin A} edge_{ij}}{min\{edge(A), edge(\bar{A})\}} \qquad \text{Equation 2.8}$$

where A is a cluster in the given network, $\Sigma_{i \in A, j \notin A} edge_{ij}$ = intercluster edges of A, $edge(A)$ = edges with both endpoints in cluster A and $edge(\bar{A})$ = edges with no endpoints in A. To find the value of the overall conductance in a network G, 1 is subtracted with the average of the conductance of all clusters in the network, as shown in Equation 2.9.

$$\phi(G) = 1 - \frac{1}{N}\Sigma_N \phi(A) \qquad \text{Equation 2.9}$$

where $\phi(G)$ = the conductance of network G, N = number of clusters in G, A = a cluster in the graph. The subtraction from 1 is used to ensure that the value 1 is the most preferred value to indicate graph with a good conductance. The conductance of a network ranges from 0 until 1. In the analysis of clustering algorithms done by (Emmons et al., 2016), they concluded that among the three metrics that they used, which include modularity, conductance and coverage, conductance is regarded as "the stand-alone quality metric that best indicates performance on the information recovery metrics".

v. Coverage

The *coverage* of a given clustering output is the result of dividing the weight or the number of all intracluster edges and the total weight or number of all edges in the graph G (Brandes et al., 2003; Almeida et al., 2011) as depicted in Equation 2.10.

$$Coverage(G) = \frac{weight(edges\ in\ all\ clusters\ in\ G)}{weight(edges\ in\ G)} \qquad \text{Equation 2.10}$$

The values can vary within the range [0,1] in which a value of 1 or near to 1 means that more edges are linking vertices inside the cluster, indicating a good clustering. It goes without saying that this metric actually captures the compactness of the intra-cluster edges (Emmons et al., 2016). However, when all vertices are assigned to the same cluster, the metric gives a high value, but the clustering done on the network can be meaningless. Figure 2.23 shows example of coverage calculation.

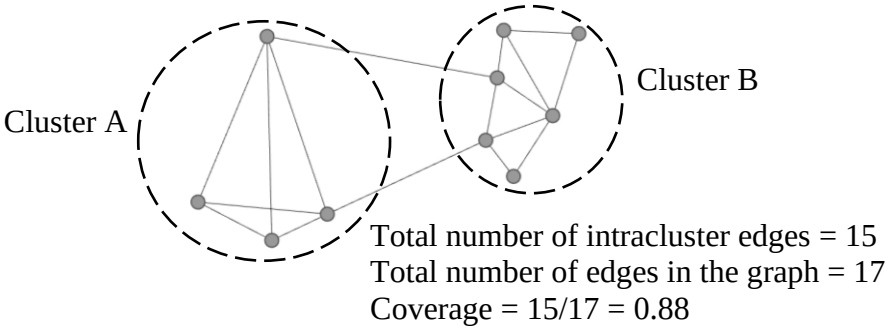


Figure 2.23: Illustration of Coverage Metric

Various computational metrics have been introduced to evaluate clustering algorithms. Table 2.4 shows the evaluations conducted on clustering algorithms by previous researchers for the past years along with the metrics they used.

Table 2.4: Evaluation Metrics Used on Clustering Algorithms for the Past Years

Source	**Clustering Evaluation Metrics**						**Evaluated Clustering Algorithms**
	Cluster Path Length	**Coverage**	**Conductance**	**Running Time**	**Relative Density**	**Modularity**	
(Brandes et al., 2003)		✓	✓				Markov Clustering, Iterative Conductance Cutting, Geometric MST Clustering.
(Kannan et al., 2004)			✓	✓	✓		Spectral clustering.
(Zaidi et al., 2010)	✓				✓	✓	Bi-secting K-Means algorithm, Divisive Clustering Algorithm Based on Edge Centrality.
(Almeida et al., 2011)		✓	✓			✓	Markov Clustering, Bisecting K-means, Normalised Cut, Spectral Clustering.
(Almeida et al., 2012)			✓		✓	✓	None.

Table 2.4 continued

(Emmons et al., 2016)		✓	✓			✓	Louvain, Infomap, Label Propagation, Smart Local Moving.
(Hamasuna et al., 2017)						✓	Louvain, k-medoids Clustering
(Lizinski et al., 2015)				✓			Modularity-based Algorithm, Label Propagation and Information Flow Algorithm

Modularity and conductance are the most commonly used metrics when evaluating graph clustering algorithms. In fact, modularity, or also known as Q metric, is among the mostly used and initially used metric to evaluate clustering (Girvan & Newman, 2002). Relative Density is another popular metric used which is related to the edge density in a graph.

Relative density and modularity are among the most commonly used metrics by previous researchers. However, interestingly, relative density metric alone will not be sufficient to indicate good clustering algorithms as it is possible for a cluster to have the same value of density, but one could have vertices that are more distant from each other compared to another. This is the reason why multiple metrics must be used at once.

Modularity metric may also give negative values in the case where there are singleton clusters, which are clusters with only one vertex. Since there will be no internal edge in singleton cluster, the value of its trace can be zero. Too many singleton clusters in a graph may cause its trace value to be so low that it may affect the value of the modularity by lowering it further, even though other non-singleton clusters are very well formed (Almeida et al., 2011).

Interestingly, conductance has been claimed to be as the best stand-alone metric by Emmons et al. (Emmons et al., 2016) compared to modularity and coverage. CPL metric on the other hand has only been invented in that same year of 2016. Its inventors claimed that the metric covers broader aspects than most metrics when evaluating clustering algorithms. It is instead an interesting metric where there is an extra aspect considered in its calculation, when compared with the other metrics. Table 2.5 shows the features required by each metric in its computation.

Table 2.5: Features Involved in Metrics Computation

Cluster Metric	Intra-cluster edges	Extra-cluster edges	Path Lengths
Relative Density	✓	✗	✗
Modularity	✓	✗	✗
CPL metric	✓	✓	✓
Conductance	✓	✓	✗

This research also considers running time, the common evaluation metric for graph clustering and layout algorithm to evaluate the efficiency of the algorithms in performing their clustering. Another common evaluation metric, which is the scalability of algorithms are omitted and the reason is explained in Section 3.3.4. Coverage metric on the other hand has a basic concept based on the intracluster edges of the graph, which are also considered in the other metrics including modularity, conductance, relative density and average cluster path length. Therefore, this metric is omitted as it possessed similarity.

Since researchers nowadays are still arguing on which cluster metric is best to evaluate clustering algorithms, and no agreement has been achieved so far on the matter, the running time, relative density, modularity, conductance and CPL metric will be used and discussed about in the evaluation part of this research. It can be seen that most of the metrics

adopt edges as significant criteria to be evaluated. Further details on the calculation of each metric will be discussed in Chapter 3.

2.7 Textual Data Transformation for Visualisation

Documents, speeches, news articles, and websites produce a large number of texts (Thomas & Cook, 2005) and these data keep growing in volume. Textual data can be unstructured (e.g., a collection of plain text), semi-structured (e.g., XML texts), structured (e.g., relational tables) and imperfect (texts with noise, errors, missing texts or ambiguous values (Kasik et al., 2009)).

Today, many historical documents have been digitally archived worldwide and the trend now is to visualise these documents in many different forms such as visual networks, map (Hinrichs et al., 2015), word cloud (e.g., IBM Watson News Explorer), vertical tag cloud (Hinrichs et al., 2015), timeline (e.g., Wiki SAGA (Tan et al., 2015), treemap (e.g., the Yale project Photogrammar), dashboard (e.g., the Yale project Photogrammar), etc. The common objective of document visualisation is to access visually and interactively into the document content to enable exploration, analysis, search, or browsing. Various methods have been used by previous researchers to produce a network visualisation.

Grobelnik and Mladenic (2004) created Contexter, a software to visualise the summaries of news articles. The text of the articles is first cleaned from noises and bag-of-words representations are created from it. NEs consisting of names of people, companies, place, and product names are also extracted from the cleaned texts and are used to generate a visual network of the articles. In the network, two NEs are connected if they occur in at least one shared document. The NEs are identified based on word capitalisation (to identify proper nouns) added with extra method to unify the different surface forms of the same NE.

The user interface of Contexter allows a user to visualise the network of NEs and select an NE to view its contexts, depicted by the bag-of-words generated. User is also able to read the original texts containing the selected NE after the NE is clicked. Grobelnik and Mladenic (2004) illustrated their approach by processing 11,000 article summaries of the length 200-400 words from the Association for Computing Machinery (ACM) Technology News service. However, they did not apply any visual network algorithms nor provide any evaluation of Contexter.

The interests of researchers in representing historical documents into visual networks can be seen in the visualisation of Hyohanki, a diary written by aristocrat during the late Heian period of Japanese classical era (Osaki et al., 2011). Their work aimed to propose a method to reveal and visualise relationships between historical persons using locational information expressed in historical documents. Specifically, they are using digitised text of Hyohanki, which is written personally by Nobunori Taira for 52 years from 1132 to 1184. He was a trusted vassal of the emperors and ministers at the time and the diary he wrote was a valuable record of official and political events that occurred during the period. Only NEs of category PEOPLE and LOCATION are used in this study. If the name of a person appears in the same paragraph with the name of a location, they are assumed to be a co-occurrence and are connected in the visual network. Location is used as features of vertices in the network. Cosine similarity and a modified K-means algorithm are used to generate and cluster the vertices according to their respective locations. In the end, 78 selected known aristocrats and samurai names are clustered using their location similarities and are compared with the factions that they are known to belong to. Although it is possible that although a person's name and a location co-occur in the same paragraph, their relationship

is such as ‘A has never visited B’. The results of their work showed that their locational clusters correspond to the person’s historically known factions.

An interactive visualisation system which is able to generate a network from a historical data has been proposed by Itoh and Akaishi to represent the changing relationship of historical figures in the data with time (Itoh & Akaishi, 2012). The authors used Dai-Nihon Shiryo historical database. It contains Japanese historical documents (from ninth to seventieth centuries) arranged chronologically. Each record in the database represents an event consisting of the names of historical figures, their titles, a list of location names, a list of keywords, and the text corresponding to the event. Force-directed layout algorithm is used to generate the drawing of the graph. The relation between two persons p1 and p2 in a particular year is computed by dividing the number of records that contain p1 and p2 by the number of records containing p1 only. In the generated graph, the size and colour of a vertex represent the importance of a person. The size of a vertex increases with the increase of other person’s dependency on the person represented by the vertex and the colour of the vertex changed according to the ratio of the in-degree to the out-degree of the vertex (Itoh & Akaishi, 2012). The strength of a relationship between two persons is emphasised by the length of an edge. A short edge indicates a strong relationship.

Due to the availability of many open-source software like Gephi for visual network generation and analysis, a great number of projects has emerged in recent years. These projects are usually exhibited on websites. For instance, the website *Visualising Historical Networks* provides visual networks that featured “the way people in the past interacted with each other and their surroundings”. Another website using Gephi is the visualisation of the history of philosophy created by Simon Raper (Raper, 2012), where data about philosophers in Wikipedia are connected in a graph with an “influenced by” relation. The author used the

information stored in DBpedia from an infobox on a Wikipedia page. A website created by Chris Harrison (Harrison, 2007) exhibits the visualisation of various data using his own visual tools. One of them is a visual network of NEs from the King James Bible. The focus is on Person and Location. The vertices in the graph are people and places and the edges were defined based on the co-occurrence of pairs of NEs mentioned in the same verse. A clustering algorithm was used to layout the vertices so that related NEs are placed close to each other. Labels are scaled according to the number of connections they have.

2.8 Summary

This chapter started with an introduction to the historical document used in this study which is SAGA, the background of graph theory, and the categories of visual network algorithms. The review done focused on two categories, which are graph layout algorithms and graph clustering algorithms. The review also contains comparisons between network visualisation tools and their features. The commonly used evaluation metrics and the specifically used evaluation metrics for each category of the focused visual network algorithms are also reviewed. The chapter ends by reviewing on the previously published works on the visualisation of textual data into visual networks, covering how the relation between keywords in the datasets are established and the algorithms they used (if exist).

CHAPTER 3

METHODOLOGY

3.1 Overview

In this chapter, we introduce and propose a generic framework to perform the evaluation of visual network algorithms to obtain the best network representation of a document. In order to evaluate the effect of the algorithms on SAGA, we need to turn the documents into a network first. Therefore, a discussion on the aspects that need to be considered to turn SAGA into a network is also included. A simpler cluster evaluation metric which is reliable to evaluate both the internal and external characteristic of the clusters in graphs is also introduced.

3.2 The Proposed Generic Evaluation Framework

The proposed framework to do evaluation on visual network algorithms to find the best network representation of a single documents is shown in Figure 3.1. It is modular in characteristic where each part is able to be modified or removed according to situation or preference. The proposed framework is divided into six parts namely document filtering process, annotation process, data transformation process, graph tool input formatting process, graph generation process and evaluation process. The following subsections include the explanation for each process in the framework in sequence.

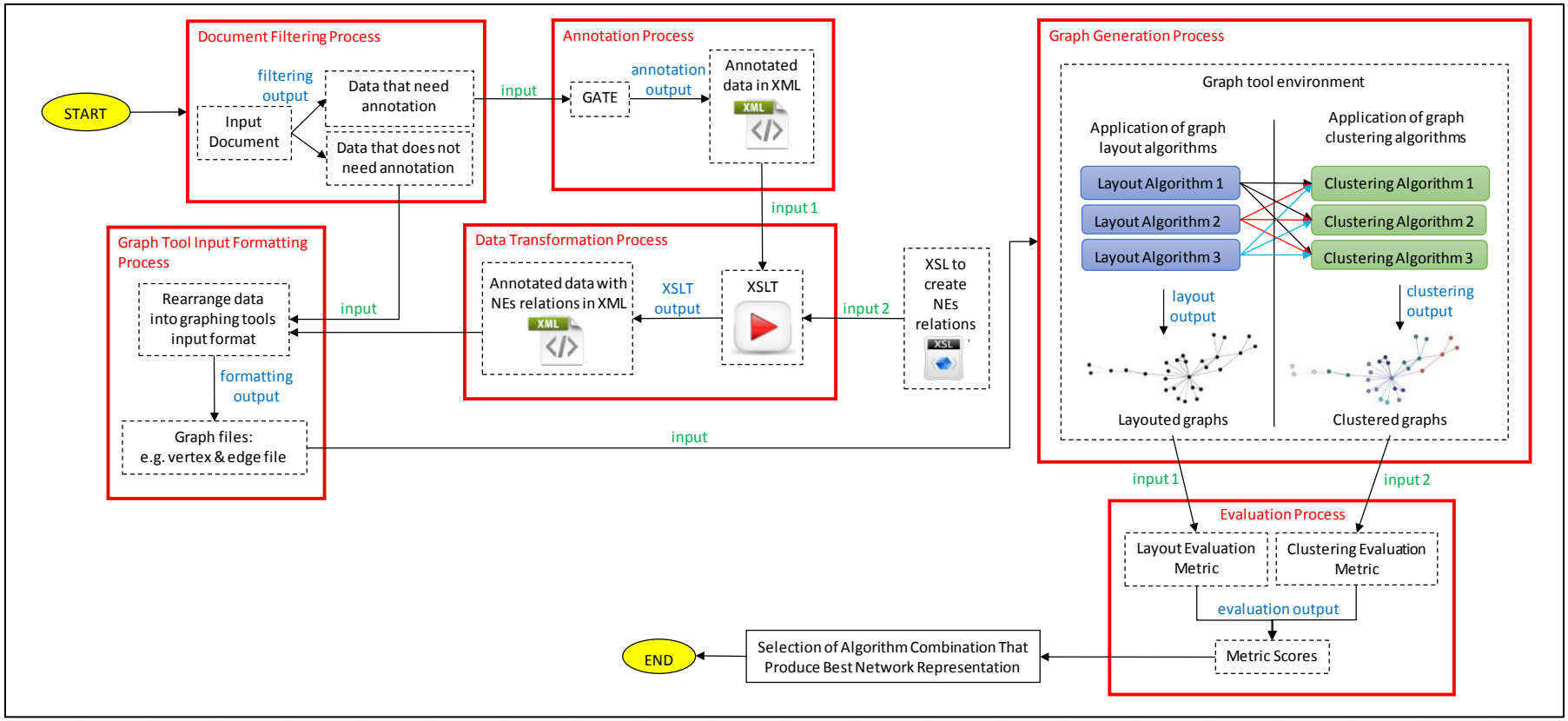


Figure 3.1: The Proposed Generic Framework

3.2.1 Document Filtering Process

The first process in the proposed framework is the document filtering process. In this process, the input documents are first filtered to decide whether they fall under the category of data that needs annotation or data that does not need annotation. If they are data that need annotation, the data will undergo annotation process first. If they are data that does not need annotation, the data will be forwarded directly to the graph tool input formatting process which is explained in Section 3.2.4.

3.2.2 Annotation Process

In this process, the annotation tool that is readily suggested to be used is the GATE (General Architecture for Text Engineering) annotation tool. At this part, GATE can be replaced with any other annotation tool of preference, but if GATE is used, then the annotation output will be annotated data, preferably in the form of XML. This is because the GATE XML output file will be required to undergo another process after this annotation process. At this part, the XML output file only contains details of NEs in the documents. There is yet no information on the relations between NEs in the documents. Therefore, this XML output file needs to be forwarded to the data transformation process to generate information on the relations between the NEs.

3.2.3 Data Transformation Process

In this process, the annotated XML file obtained from the annotation process is combined with an XSL file to undergo XSL transformation process to produce a newly-structured XML file which contains the details of the annotated NEs along with their

relations information. Information on the NEs relations to be extracted by the XSL file can be the sentence id of the sentence, paragraph id of the sentence, or etc. depending on our preferred choice.

3.2.4 Graph Tool Input Formatting Process

In this process, the newly-structured XML file which is obtained from the data transformation process is restructured to be in the format that is acceptable by the graphing tool. Normally, graph tools required both the vertices and edges files to generate a network. These input files depend on the graph tools chosen to be used in the graph generation process.

3.2.5 Graph Generation Process

In this process, the output of the graph tool input formatting process is supplied to the graph tool to generate the network. Since our framework aims to find the best network representation of a document, both graph layout algorithms and graph clustering algorithms are applied to the network to produce a number of layouted and clustered networks to be evaluated. Multiple graph layout algorithms are applied first to the network. The layouted networks outputs are then applied with multiple graph clustering algorithms to produce a number of clustered networks.

3.2.6 Evaluation Process

In this process, each of the layouted network is evaluated using layout evaluation metrics of our choice. Each of the clustered network is also evaluated using clustering evaluation metrics of our choice. From the evaluation, the scores of each metric used will be

obtained. And these scores will help to determine the best combination of graph layout and clustering algorithms which can produce the best network representation of a document – has both good layout and high-quality clustering.

3.3 Applying the Generic Framework on SAGA

Before applying the framework on SAGA, we need to determine first the information that are to be extracted to constitute its visual network. One characteristic of SAGA is that it consists of long sentences. Figure 3.2 shows an example of SAGA long sentence. There are several ways to generate ad-hoc relations for SAGA NEs, either to relate NEs in the same edition, NEs in the same articles, NEs in the same paragraphs or NEs in the same sentence. Visualising NEs in the same sentence is the most appropriate for SAGA since we want to generate a network that is intelligible and readable. This is because connecting NEs in the same sentence already result in a network with many edges.

...Those present included, with few exceptions, her Royal Highness Princess Christians, with Mrs. D. and Major Evan Martin in waiting; the Ranee of Sarawak, Lord and Lady Rosmead, Lord and Lady Arthur Hill and Miss Hill, Lady and Miss Brooke, Lord Glenesk,the Mayor and Mayoress of Salisbury, the Dowager Countess De La Warr, Princess Alexis Dolgorouki, Mr. and Mrs. G. W. Palmer, Dr. and Mrs. Waller and family, Professor and Mrs. Poulton and Miss Poulton, Mr. and Mrs. C. H. Palmer, Miss F. Palmer, Miss Craig, Mr. and Mrs. Donald Craig, Miss N. Palmer, Mr. and Mrs. S. E. Craig, Captain and Mrs. Johnson, Mr. and Mrs. Alfred Palmer, Mr. and Mrs. Lionel Gosling, the Rev. T. and Mrs. Brackenbury, Colonel and Mrs. Lewis Hope, Mr. Bampfylde, Mrs. Job, Mr. and Mrs. H. de Windt, Mr. and Mrs. Craig, Mr. and Mrs. F. Harris, Mr. E. P. Poulton, Miss A. Brooke, Mrs. J Craig, Miss D. Craig, Mr. Archie Craig, Admiral Sir Cyprian and Lady Bridge, Lady Swansea and the Hon. Misses Vivian, Lady and Miss Davey, Lady Amherst and the Hon. Misses Amherst, Sir, P. Burne-Jones, the Countess of Romney, Sir Douglas Straight, Sir R. and Lady Penrose-Fitzgerald, the Hon. Alexander Yorke, Sir C. and Lady Furness, Colonel the Hon. Heneage Legge, M. P., Sir Lewis and Lady Melver, Lady Dorothy Nevill and Miss Nevill, Lady Palmer, the Hon. Eustace and Mrs. Fiennes, Lady Seymour, the Dowager Marchioness of Londonderry, the Marchioness de Sain, Sir C. and Lady Cayzer and Miss Cayzer, the Dowager Viscountess Helmsley, Lady and the Misses Lister, Count and Countess Lutzow, Miss Emily Loch, Sir H. and Lady Alderson, Sir John and Lady Mazwell, Sir William and Lady Miller, Sir A. and Lady Rucker, Lord and Lady Churston and the Hon. Lois Yarde-Buller, Lady Nottage, Lord Edmond Fitzmaurice, Lord Saye and Sele, Lady and Miss Blomfield, Lady and Miss Cowell Stepney, the Countess of Limerick, Dora Countess of Chesterfield, Sir James and the Misses Blyth, Lady Hardman, Sir R. and Lady Catheart, the Hon. Mrs. Newdigate and the Hon. Emily Ward, Mr. Piuckney, and many others. ...

Figure 3.2: An Example of SAGA Long Sentence

Table 3.1 shows an example of how the number of NEs and relations in the SAGA change after the addition of each subsequent editions starting from 4th June 1904 until 1st December 1904. It is noticed that with the addition of editions, the number of NEs (size of the graph) does not increase much compared to the increase of relations in the graph. Until at some point, addition of editions does not change the size of the graph anymore. This is due to the repetitions of the same NEs in the accumulated editions. Even the number of relations remains the same as only the weight (frequency) of the relations increase. Clearer visualisation of the change is depicted in Figure 3.3.

Table 3.1: Change in Number of NEs and Relations in SAGA

Edition	Number of NEs	Relations
4th June 1904	306	1116
1st July 1904	441	1832
2nd August 1904	735	4622
1st September 1904	855	5280
3rd October 1904	933	5760
3rd November 1904	1012	6151
1st December 1904	1012	6151

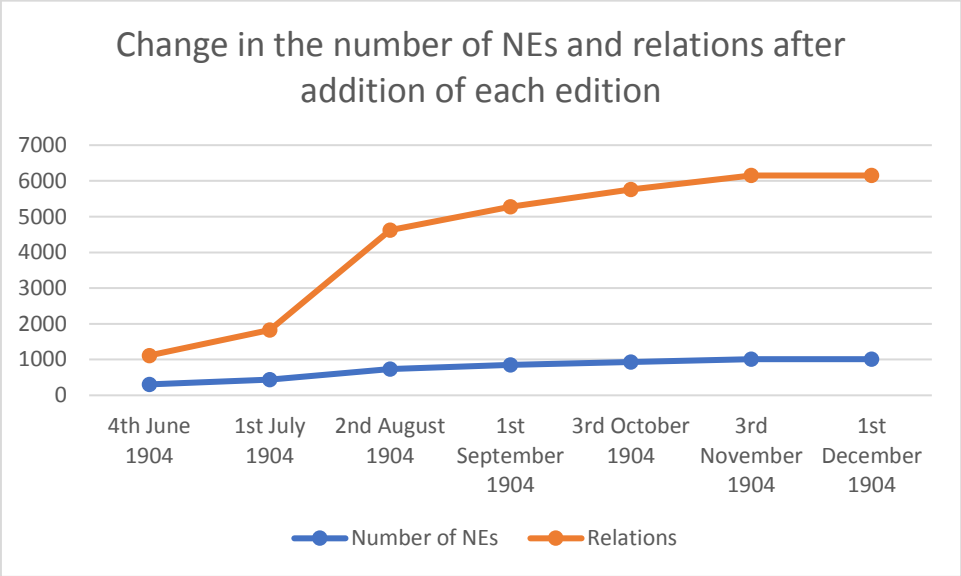


Figure 3.3: Line Graph for Changes in NEs and Relations in SAGA

We also identify SAGA as a complex dataset because one edition of SAGA has a small number of vertices but a large number of relations. This is due to the long sentences present in SAGA and because the same NEs tend to be repeated in SAGA. Since using many editions to create SAGA network has no point since it does not allow us to obtain a large enough network to perform scalability testing, it is decided to use only one edition of SAGA to avoid a network that is too complex (with too many relations). This is also to have a clearer observation on the change in the evaluation metrics especially edge crossings. Apart from

that, it is also difficult for any graph layout algorithms to draw a good layout for a network with too many edges but only a few vertices.

Therefore, in this study, only a subset of SAGA is used, specifically the edition published on 12th January 1904. When the relations are established, the data formed eleven separately linked NEs. The biggest group of the linked NEs is taken to be evaluated, since the other ten groups which are rather small consisting of only one to three linked NEs are not helpful in understanding the effect of the algorithm on the network built later. Having unconnected components in the network will also cause bias in the cluster metric scores. The biggest group of data consists of 839 NE relations between the 224 NEs.

Figure 3.4 shows the proposed generic framework when applied on SAGA using Gephi graph tool. In the document filtering process, SAGA is first filtered. Since SAGA is a document that need to be annotated first, it needs to undergo the annotation process. In this part, annotation is done using GATE and the output of the annotation process are XML files containing annotated SAGA's NEs. The XML files are then combined with an XSL file to undergo XSL transformation so that information on the NEs relations are extracted. The output of the XSL transformation process is another XML file with a new structure which contains the annotation details of NEs in SAGA along with the information on their relations. Since the graphing tool used in this study is Gephi, the XML file are then restructured to be in the format required by Gephi to generate network, which are into two CSV files representing the vertices files and the edges files of the network. When the two files are supplied to Gephi, Gephi generate the network and three layout algorithms are applied separately to the network and each of the output is evaluated using the layout evaluation metrics namely running time, cluttered vertices and edge crossing. The output of each graph

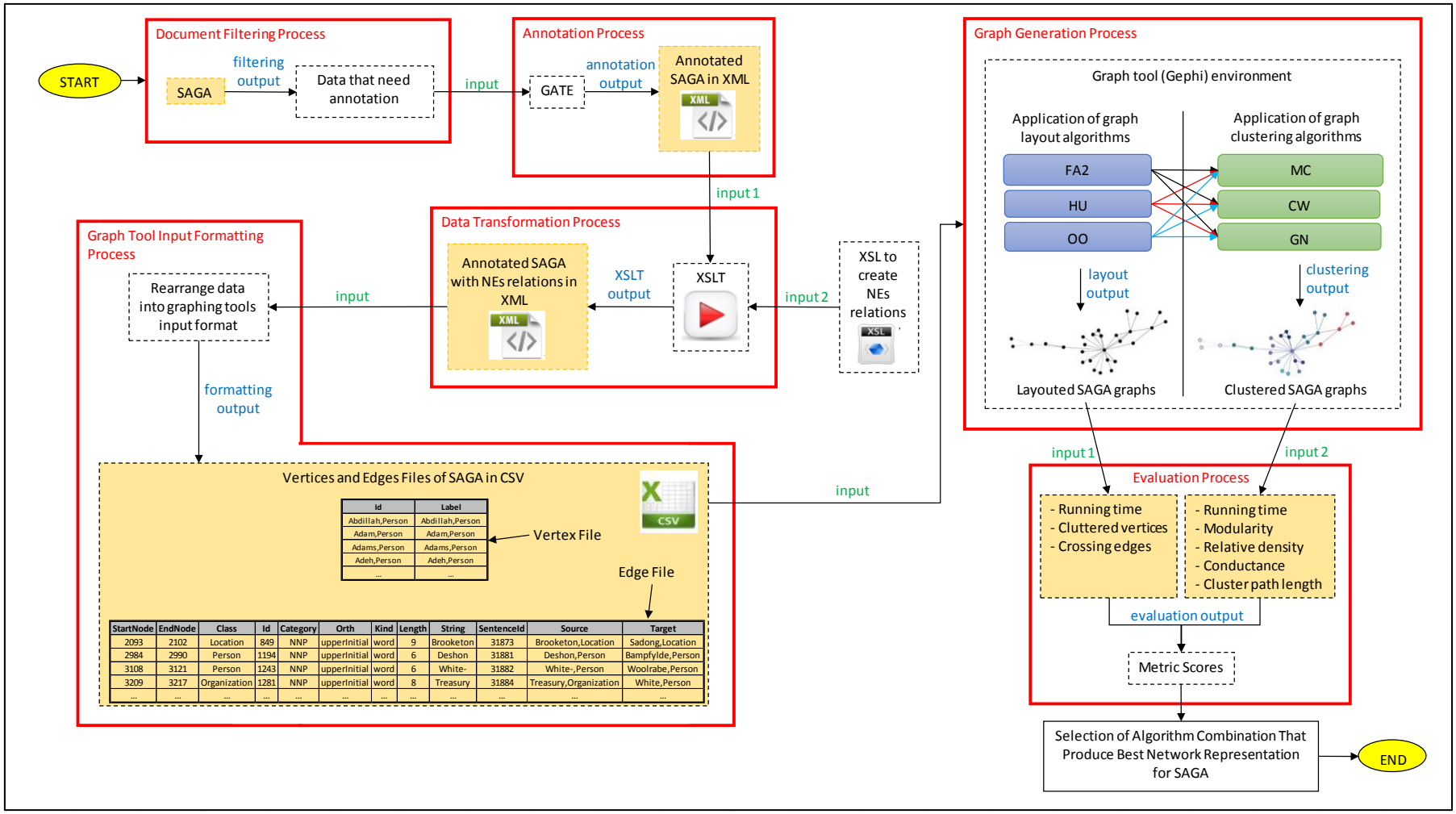


Figure 3.4: Applying the Proposed Framework on SAGA using Gephi Graph Tool

layout algorithm is also applied separately with three clustering algorithms. The output networks of the clustering algorithms are then evaluated using clustering evaluation metrics, namely running time, modularity, relative density, conductance, and cluster path length. The evaluation metric scores are then analysed and the combination of algorithms that produce the best network representation for SAGA is selected.

3.3.1 SAGA's Data Transformation Process

The data transformation process in Figure 3.4 can be represented by Figure 3.5. GATE XML files obtained from the GATE annotation tool needs to be restructured in order to first extract all annotated NEs along with their features from the files. Since the GATE XML file cannot be applied directly to the visualisation tool to generate a graph, an XSL transformation (XSLT) process is required where an XSL file is created to be parsed through the GATE XML file to produce a new XML file with extracted and restructured NEs details in it. From the parsing process, an XML file with the desired structure that can be used to generate a graph is obtained.

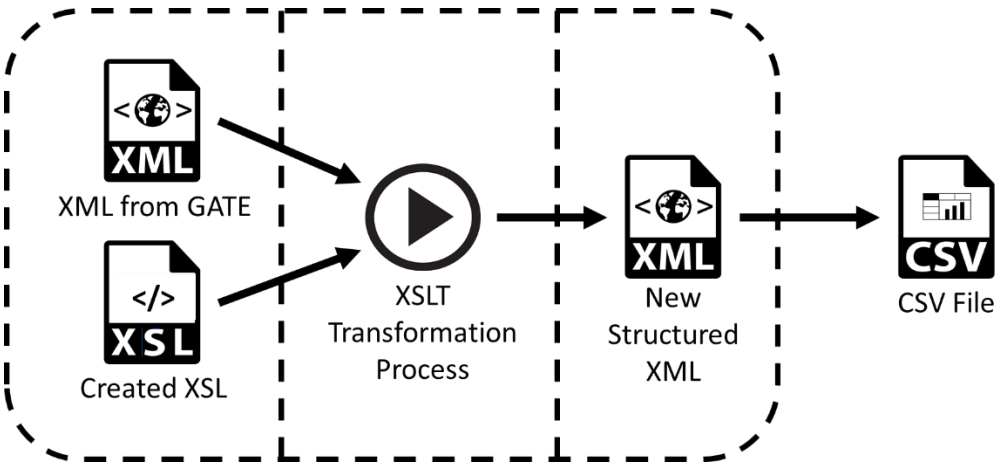


Figure 3.5: Data Transformation Process

The GATE XML document is presented in a hierarchical or tree structure as shown in Figure 3.6 (a). The root of the tree is <GateDocument> and it is highlighted in blue in the figure. The root has four sub-elements which is highlighted in red: <GateDocumentFeatures>, <TextWithVertices>, <AnnotationSet>, and <AnnotationSet.Name>. Figure 3.6 (b) shows the simple tree structure of the document.

(a) Actual XML file

(b) XML's tree structure illustrated

Figure 3.6: GATE XML Structure

For this study, the only part that is focused on is the <AnnotationSet> element as it contains all information related to the annotation of each word-token as illustrated in Figure 3.7 and Figure 3.8. The first part of the annotation, from line 135865 until line 135886 which is depicted in Figure 3.7, corresponds to the linguistic and non-linguistic features of the word *Edward*.

The linguistic features in the <Annotation> element are part of speech class ('category'), type of the string ('kind'), and the word-token itself ('string'). The non-

linguistic features are the ‘Id’, ‘Type’, position (‘StartNode’ and ‘EndNode’), ‘length’, and orthography (‘orth’).

```
135865  <Annotation Id="8775" Type="Token" StartNode="21665" EndNode="21671">
135866  <Feature>
135867    <Name className="java.lang.String">length</Name>
135868    <Value className="java.lang.String">6</Value>
135869  </Feature>
135870  <Feature>
135871    <Name className="java.lang.String">category</Name>
135872    <Value className="java.lang.String">NNP</Value>
135873  </Feature>
135874  <Feature>
135875    <Name className="java.lang.String">orth</Name>
135876    <Value className="java.lang.String">upperInitial</Value>
135877  </Feature>
135878  <Feature>
135879    <Name className="java.lang.String">kind</Name>
135880    <Value className="java.lang.String">word</Value>
135881  </Feature>
135882  <Feature>
135883    <Name className="java.lang.String">string</Name>
135884    <Value className="java.lang.String">Edward</Value>
135885  </Feature>
135886  </Annotation>
```

Figure 3.7: Annotation of the Word-token Edward

The second part of the annotation, from line 493605 until line 493630, which is shown in Figure 3.8 corresponds to the annotation of the word-token *Edward* as Person.

```
493605  <Annotation Id="39812" Type="Person" StartNode="21665" EndNode="21677">
493606  <Feature>
493607    <Name className="java.lang.String">ruleFinal</Name>
493608    <Value className="java.lang.String">PersonFinal</Value>
493609  </Feature>
493610  <Feature>
493611    <Name className="java.lang.String">surname</Name>
493612    <Value className="java.lang.String">Jenke</Value>
493613  </Feature>
493614  <Feature>
493615    <Name className="java.lang.String">gender</Name>
493616    <Value className="java.lang.String">male</Value>
493617  </Feature>
493618  <Feature>
493619    <Name className="java.lang.String">firstName</Name>
493620    <Value className="java.lang.String">Edward</Value>
493621  </Feature>
493622  <Feature>
493623    <Name className="java.lang.String">kind</Name>
493624    <Value className="java.lang.String">fullName</Value>
493625  </Feature>
493626  <Feature>
493627    <Name className="java.lang.String">rule</Name>
493628    <Value className="java.lang.String">PersonFull</Value>
493629  </Feature>
493630  </Annotation>
```

Figure 3.8: Annotation of the Word-token Edward as Person

The output of the parsing process is as depicted in Figure 3.9. It contains all the required NE features and values provided by the GATE XML. The whole version of the XSL stylesheet used in the transformation process is included in Appendix A.

StartNode	EndNode	Class	Id	Category	Orth	Kind	Length	String	SentenceId
178	182	Organization	144	NNP	upperInitial	word	4	Dyak	31854
424	438	Organization	235	NNP	upperInitial	word	7	Customs	31855
623	642	Organization	312	NNP	upperInitial	word	11	Settlements	31856
1078	1083	Location	475	NNP	upperInitial	word	5	State	31860
...	...	...	...	...	...	...	...	...	...

Figure 3.9: NE Features

XSLT is the transformation component of the XSL stylesheet technology. An XSL stylesheet can be applied to translate data from one XML grammar to another. In this study, a new XML file that has only the selected data from a GATE XML is created, in our preferred

arrangement using our own XSL, as shown in Figure 3.10. As stated in line 3 in the figure, the output file is specified to be in XML format.

```
1  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
2
3      <xsl:output method="xml"/>
4
5      <xsl:variable name="newline">
9
10     <xsl:template match="/">
13
14     <xsl:template match="GateDocument">
19
20     <xsl:template match="AnnotationSet">
70
71 </xsl:stylesheet>
72
```

Figure 3.10: Overall XSL Stylesheet to Transform GATE XML

The code from line 5 to 8, which is depicted in Figure 3.11, indicates that a global variable named as “newline” is declared so that newlines can be inserted into the output to improve readability.

```
1  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
2
3      <xsl:output method="xml"/>
4
5      <xsl:variable name="newline">
6          <xsl:text>
7          </xsl:text>
8      </xsl:variable>
9
10     <xsl:template match="/">
13
14     <xsl:template match="GateDocument">
19
20     <xsl:template match="AnnotationSet">
70
71 </xsl:stylesheet>
72
```

Figure 3.11: Inserting New Line

The symbol '/' (line 10) in the code <xsl:template match:"/"> in Figure 3.12 means the root of the XML file. In the figure, <GateDocument> is selected as the root of the file (line 10-12).

```
1  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
2
3      <xsl:output method="xml"/>
4
5      <xsl:variable name="newline">
9
10     <xsl:template match="/">
11        <xsl:apply-templates select="GateDocument"/>
12     </xsl:template>
13
14     <xsl:template match="GateDocument">
19
20     <xsl:template match="AnnotationSet">
70
71 </xsl:stylesheet>
```

Figure 3.12: Determining the XML Root

The code from line 14 to 18 in Figure 3.13 conveys that under the <GateDocument> element, an element named as <AnnotationStartsHere> is created. In this element, the default <AnnotationSet> element is selected.

```
1  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
2
3      <xsl:output method="xml"/>
4
5      <xsl:variable name="newline">
9
10     <xsl:template match="/">
13
14     <xsl:template match="GateDocument">
15        <xsl:element name="AnnotationStartsHere">
16           <xsl:apply-templates select="AnnotationSet"/>
17        </xsl:element>
18     </xsl:template>
19
20     <xsl:template match="AnnotationSet">
70
71 </xsl:stylesheet>
```

Figure 3.13: Selecting <AnnotationSet> Element

The XSL stylesheet extracts and combines the features and values of all annotated strings in the GATE XML document with the constraint that these strings must have an NE class label (Title, Organisation, River, Person, Money, Date, Location, JobTitle, FirstPerson, Ethny) as indicated in the line code 22 to 24 of Figure 3.14.

```
20  <xsl:template match="AnnotationSet">
21      <xsl:value-of select="$newline"/>
22      <xsl:for-each select=".//Annotation[@Type='Title'] | .//Annotation[@Type='Organization'] | .//Annotation[@Type='River'] |
23      .//Annotation[@Type='Person'] | .//Annotation[@Type='Date'] | .//Annotation[@Type='Location'] |
24      .//Annotation[@Type='JobTitle'] | .//Annotation[@Type='FirstPerson'] | .//Annotation[@Type='Ethny']">
25          <xsl:element name="Annotation">
26              <xsl:attribute name="StartNode">
27                  <xsl:value-of select="@StartNode"/>
28              </xsl:attribute>
29              <xsl:attribute name="EndNode">
30                  <xsl:value-of select="@EndNode"/>
31              </xsl:attribute>
32              <xsl:attribute name="Class">
33                  <xsl:value-of select="@Type"/>
34              </xsl:attribute>
35
36              <xsl:variable name="start" select="@StartNode"/>
37              <xsl:variable name="end" select="@EndNode"/>
38
39              <xsl:for-each select="//Annotation[@Type='Token' and @StartNode>=$start and @EndNode&lt;=$end]">
40                  <xsl:attribute name="Id">
41                      <xsl:value-of select="@Id"/>
42                  </xsl:attribute>
43                  <xsl:attribute name="Category">
44                      <xsl:value-of select="Feature[Name='category']/Value"/>
45                  </xsl:attribute>
46                  <xsl:attribute name="Orth">
47                      <xsl:value-of select="Feature[Name='orth']/Value"/>
48                  </xsl:attribute>
49                  <xsl:attribute name="Kind">
50                      <xsl:value-of select="Feature[Name='kind']/Value"/>
51                  </xsl:attribute>
52                  <xsl:attribute name="Length">
53                      <xsl:value-of select="Feature[Name='length']/Value"/>
54                  </xsl:attribute>
55                  <xsl:attribute name="String">
56                          <xsl:value-of select="Feature[Name='string']/Value"/>
57                  </xsl:attribute>
58              </xsl:for-each>
59
60              <xsl:for-each select="//Annotation[@Type='Sentence' and @EndNode>=$start and @StartNode&lt;=$end]">
61                  <xsl:attribute name="SentenceNumber">
62                      <xsl:value-of select="@Id"/>
63                  </xsl:attribute>
64              </xsl:for-each>
65
66          </xsl:element>
67          <xsl:value-of select="$newline"/>
68      </xsl:for-each>
69  </xsl:template>
```

Figure 3.14: Extracting and Combining NE's Features

An element named <Annotation>, having "StartNode", "EndNode" and "Class" as its attributes is created for each of the selected <Annotation> in the original document. The value of "StartNode" in the output XML file would be the value of "StartNode" in the original XML document. Similarly, the value of "EndNode" in the output XML is the value of "EndNode" in the original XML document. The value for the attribute "Class" in the

output XML will be the value of the attribute “Type” in the original document. After extracting the three attributes (“StartNode”, “EndNode”, and “Class”), other information on the annotated string also need to be extracted for the data to be meaningful. In order to do that, two local variables called “start” and “end” are declared to represent the value of “StartNode” and “EndNode” of the current <Annotation> element (line 36 to 37). Starting from line 39 to 58 in Figure 3.14, an <xsl:for-each> loop is created to check for an <Annotation> element with “Type” equal to Token that has the same “StartNode” and “EndNode” as the current <Annotation> being extracted. If there is so, then the value of attribute “Id”, “Category”, “Orth”, “Kind”, “Length” and “String” of that <Annotation> element will be extracted too to be combined as attributes of the current <Annotation>.

Line 60 to 64 create another <xsl:for-each> to check for another <Annotation> element with “Type” equal to Sentence that has the same “StartNode” and “EndNode” as the current <Annotation>. If there is so, then the value of attribute “Id” of that <Annotation> element will be extracted too to be combined as another attribute of the current <Annotation>, which is “SentenceId”.

The next step is to parse the GATE XML file with the created XSL file through a processor to create the new structured XML file. There are many processors to choose from, and in this case, Saxon is used as it is simple and easy to handle. Saxon is a command line tool and it is built as a Java application. A Java runtime must be installed in the computer in order to run Saxon. The latest version of Saxon can be downloaded from saxon.sourceforge.net. Once the downloading and installation are done, the Java CLASSPATH environment variable has to be set. This will allow Java to know where all of the executable class files are located on the computer. A good documentation of Saxon is

available at this webpage www.saxonica/documentation/index.html. The steps for parsing a GATE XML file under SXLT process using Saxon are explained in Figure 3.15.

1. Place the original XML file and the XSL file created in Saxon folder.

2. Open Command Prompt.

3. Type **cd saxon** to change directory to Saxon directory. (Be aware of the folder where Saxon is installed in your computer, command is case sensitive).

4. Type **java -jar saxon9he.jar -xsl:xslfile.xsl -s:originalXML.xml -o:outputXML.xml**
   Where: **xslfile.xsl** = name of the xsl file created
   originalXML.xml = name of the original XML annotation file from GATE
   outputXML.xml = name of the output XML file

Once the command finishes running, the output XML file is generated in the Saxon folder.

Figure 3.15: Running Saxon to Parse GATE XML

After the XSLT process is done, the output XML file is as shown in Figure 3.16 for the first 10 lines and in Figure 3.17 for the last 10 lines. The new XML output contains complete information of each annotated string starting from its 'StartNode', 'EndNode', 'Class', 'Id', 'Category', 'Orth', 'Kind', 'Length', 'String', and 'SentenceId'.

```
1  <?xml version="1.0" encoding="UTF-8"?><AnnotationStartsHere>
2      <Annotation StartNode="53630" EndNode="53638" Class="Person" Id="21889" Category="NNP" Orth="upperInitial" Kind="word" Length
3      <Annotation StartNode="54454" EndNode="54463" Class="Date" Id="22231" Category="NN" Orth="lowercase" Kind="word" Length="4" S
4      <Annotation StartNode="53210" EndNode="53218" Class="Person" Id="21723" Category="NNP" Orth="upperInitial" Kind="word" Length
5      <Annotation StartNode="53269" EndNode="53272" Class="Person" Id="21747" Category="NNP" Orth="upperInitial" Kind="word" Length
6      <Annotation StartNode="54622" EndNode="54630" Class="Date" Id="22304" Category="NNP" Orth="upperInitial" Kind="word" Length="
7      <Annotation StartNode="54633" EndNode="54640" Class="Person" Id="22311" Category="NNP" Orth="upperInitial" Kind="word" Length
8      <Annotation StartNode="54565" EndNode="54578" Class="Person" Id="22276" Category="NNP" Orth="allCaps" Kind="word" Length="10"
9      <Annotation StartNode="54599" EndNode="54602" Class="Person" Id="22286" Category="NNP" Orth="upperInitial" Kind="word" Length
10     <Annotation StartNode="52457" EndNode="52466" Class="Organization" Id="21421" Category="NNP" Orth="upperInitial" Kind="word"
```

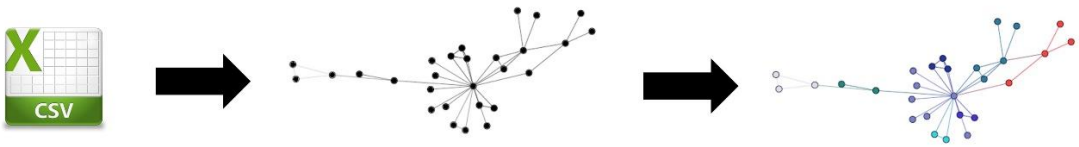
Figure 3.16: Output XML (First 10 Lines)

```
1189        <Annotation StartNode="39071" EndNode="39079" Class="Location" Id="16207" Category="NNP" Orth="upperInitial" Kind="word" Leng
1190        <Annotation StartNode="18221" EndNode="18229" Class="Location" Id="7321" Category="NNP" Orth="upperInitial" Kind="word" Lengt
1191        <Annotation StartNode="18151" EndNode="18159" Class="Location" Id="7290" Category="NNP" Orth="upperInitial" Kind="word" Lengt
1192        <Annotation StartNode="16414" EndNode="16422" Class="Location" Id="6549" Category="NNP" Orth="upperInitial" Kind="word" Lengt
1193        <Annotation StartNode="18235" EndNode="18243" Class="Person" Id="7327" Category="NNP" Orth="upperInitial" Kind="word" Length=
1194        <Annotation StartNode="18092" EndNode="18101" Class="Person" Id="7262" Category="NNP" Orth="upperInitial" Kind="word" Length=
1195        <Annotation StartNode="17064" EndNode="17077" Class="Location" Id="6825" Category="NNP" Orth="upperInitial" Kind="word" Lengt
1196        <Annotation StartNode="17064" EndNode="17077" Class="Location" Id="6825" Category="NNP" Orth="upperInitial" Kind="word" Lengt
1197
1198    </AnnotationStartsHere>
```

Figure 3.17: Output XML (Last 10 Lines)

3.3.2 Gephi's Input Formatting Process for SAGA

From the literature review, Gephi is chosen to visualise the network. This visualisation tool allows dynamic visualisation of network, which allows us to distort the vertices in the network while the tool maintains the network itself. More importantly, Gephi provides clustering algorithms which are not provided in most of the reviewed tools. Figure 3.18 shows the network generation process.



CSV file

Generating the layout using graph layout algorithms

Clustering the network using graph clustering algorithms

Figure 3.18: Network Generation

Gephi needs two CSV files to generate the network of NEs, which are, a file of vertices and a file of edges. The vertex file contains the list of vertices to be created in the network along with vertices' labels. It must include the column 'Id' and column 'Label'. In our network, vertices represent NEs. For this work, NEs of type 'Title', 'Organization', 'River', 'Person', 'Money', 'Date', 'Location', 'JobTitle', 'FirstPerson' and 'Ethny' are

visualised in the network. In our case, each NE corresponds to one single vertex, even though in the original document the NE is formed by a sequence of two or more words. For example, the full NE is *Charles Brooke.* This name of a person will be displayed in two vertices; one vertex for *Charles:Person,* and another vertex for *Brooke:Person.* Each vertex will have its NE type at the end of its label to ease identification of the NE. For example, *Sarawak* vertex, which is a NE of type Location, will be labelled as *Sarawak:Location.*

Figure 3.19 shows the table in the vertex file which is saved in CSV that is needed by Gephi. 'Id' is the name that will be used by Gephi to identify a given vertex in the network and its 'Label' will be the texts to be displayed by Gephi as the label of the vertex in the network.

Id	Label
Charles, Person	Charles, Person
Brooke, Person	Brooke, Person
Sarawak, Location	Sarawak, Location
…	…

Figure 3.19: Table in Vertex File

The edge file must include column 'Source' and 'Target'. The source vertex and the target vertex are determined by a predefined relationship. Other columns are optional. In this work, vertices that co-occur in a sentence, or in other word, have the same Sentence ID when annotated by GATE in the GATE XML file will be connected to each other with an edge. The 'Id' of each NE will be used as reference in the 'Source' and 'Target' column.

Figure 3.20 shows the table in the edge file which is saved in CSV that is needed by Gephi. In the table, 'StartNode' indicates the position number where the token starts in the document, and 'EndNode' indicates the position number where the token ends in the

documents. The token itself can be found in ‘String’ column. ‘Class’ column also contains the type of the token. ‘Id’ column will be the name used by Gephi to identify a given edge in the network. ‘Category’ column contains the part of speech (POS) tag of the token, and ‘Orth’ column contains the orthography of the token. ‘Kind’ column explains either the token is a word or a punctuation in the text. ‘Length’ columns contains the number of characters in the token and ‘SentenceId’ contains the id of the sentence in XML Gate document. ‘Source’ column contains the start vertex of the edge and ‘Target’ column contains the end vertex of the edge.

StartNode	EndNode	Class	Id	Category	Orth	Kind	Length	String	SentenceId	Source	Target
2093	2102	Location	849	NNP	upperInitial	word	9	Brooketon	31873	Brooketon,Location	Sadong,Location
2984	2990	Person	1194	NNP	upperInitial	word	6	Deshon	31881	Deshon,Person	Bampfylde,Person
3108	3121	Person	1243	NNP	upperInitial	word	6	White-	31882	White-,Person	Woolrabe,Person
3209	3217	Organization	1281	NNP	upperInitial	word	8	Treasury	31884	Treasury,Organization	White,Person
…	…	…	…	…	…	…	…	…	…	…	…

Figure 3.20: Table in Edge File

3.3.3 Graph Generation Process for SAGA

Once the basic network has been generated, the selected graph algorithms are applied for the evaluation stage. As mentioned in the review, three force-directed graph layout algorithms are used, namely ForceAtlas algorithm, Hu’s Multilevel layout algorithm, and OpenORD algorithm. The other three clustering algorithms used are Chinese Whispers clustering algorithm, Markov Clustering algorithm and Girvan-Newman clustering algorithm.

i. ForceAtlas2 Algorithm (FA2)

ForceAtlas2, (Jacomy et al., 2014), henceforth will be abbreviated as FA2, is a force-directed layout, that generates a network by assuming it as a physical system. The algorithm makes vertices to repel one another, acting similar to charged particles of the same type, while the edges are pulling the vertices, acting similar to springs. The network is allowed to move according to these forces until the forces are balanced in the end, where the final output is easier to be read. Using this concept, the algorithm can be said to rely on other vertices to find the right place of each vertex. The placement relies on the connection of the vertices, without taking into consideration their attributes.

The attraction force F_a between two vertices, v_1 and v_2 in FA2 is simply equals to the linear distance between them, $d(v_1,v_2)$, as shown in Equation 3.1.

$$F_a(v_1,v_2) = d(v_1,v_2) \qquad \text{Equation 3.1}$$

The characteristic that differs it from other force-directed algorithm was its repulsion force concept. A network consisting of a few vertices of high degree connected to many leave vertices can cause cluttering in the network output of force-directed algorithms. To address this issue, FA2 intentionally place the leave vertices closer to their parent vertex by weakening the repulsion force between them. This is done by including the degree of the vertices in the repulsion force F_r calculation, as shown in Equation 3.2.

$$F_r(v_1, v_2) = k_r \frac{(deg(v_1)+1)(deg(v_2)+1)}{d(v_1,v_2)} \qquad \text{Equation 3.2}$$

The constant k_r is the scaling value of the repulsion force that can be set by users in the algorithm's scaling setting. The addition of 1 is meant to provide each vertex in the network with a repulsion force, even for vertices with zero degree.

FA2 implemented in Gephi software consists of many settings which can be set by the users to alter the placement of the vertices to fit their needs. The settings include LinLog mode, dissuade hubs mode, gravity parameter, scaling parameter, and edge weight parameter.

Setting FA2 to be in LinLog mode will modify the attraction force in the network into a logarithmic force as shown in Equation 3.3.

$$F_a(n_1, n_2) = log(1+d(n_1,n_2)) \qquad \text{Equation 3.3}$$

This mode allows clusters in the network to be more compact. In the logarithmic force, 1 is added to the distance so that two overlapped vertices which have 0 distance between them will not give error when calculating log(0).

Gravity parameter is provided so that users can bring outlier vertices in the network closer to the centre of the network. It is good for sparse and networks with disconnected vertices, however, using it may distract the placement of vertices according to the attraction force principle.

Scaling parameter controls the repulsion force coefficient, k_r in Equation 3.2. The higher the value set by the user, the larger the network.

Edge weight parameter enables the user to decide whether the weight of the edges affect the attraction force in the network. The default value 1 set the force to be proportional to the weight. When it is 0, the weight does not affect the force.

Dissuade Hubs mode affects the shape of the network by placing vertices with high indegree in the centre of the network, compared to other vertices. This mode benefits social networks where vertices with high indegree are significant compared to other vertices.

The output network of FA2 is not a Cartesian coordinate system as the coordinates of the vertices does not imply anything. However, the placement of the vertices can be interpreted by comparing its position with other vertices. Its benefit is in providing a continuous layout of the network which can be analysed visually live. At the same time, the output network displays the modular aspect of the network, where communities are placed together as groups of vertices.

This algorithm has been used to generate graphs of 68 datasets collected from various network that have been used by Gephi users, mostly social networks and some biological networks (Jacomy, 2014). The results have been compared with Fruchterman Reingold and Hu's non-multilevel version and it was found that among all the three algorithms, FA2 gave better quality graphs after a fewer iterations. The other two required more iterations to come out with a good layout of the graph. The implementation of FA2 in LinLog mode also gave a graph with a more compact cluster. Apart from that, Gibson et al. (2014) stated that the algorithm compute with less iterations compared to Fruchterman Reingold algorithm to generate a network and showed no edge crossing in the output network. It is also emphasised that when referring to the algorithm, to get a more accurate layout with minimum force, the time taken for the algorithm to run can be sacrificed.

ii. Hu's Multilevel Layout Algorithm (HU)

Hu's multilevel algorithm (Hu, 2006), henceforth will be abbreviated as HU, invented by Yifan Hu is a force directed algorithm that generates the layout of a network by undergoing multiple levels, from coarser level to finer level. There are three stages in the algorithm, 1) coarsening, 2) coarsest network layout, and 3) prolongation and refinement. In the first stage, HU starts by coarsening the input network. Coarsening is a process of making the graph rougher, in other words, similar to making the graph simpler by combining adjacent edges and vertices together to result with a graph that has less vertices and edges (said to be coarser). Coarsening are done in series from the input network N^0 to the coarsest network N^n where each network N^{k+1} which has less edges and vertices, has the details required to generate the layout of N^k. The coarsening stops when stage 2 is obtained, which is N^n, the coarsest network. When the coarsest graph is obtained, it is easy to find the optimal layout for it since it has very few vertices and edges. In stage 3, the optimal layout of the coarsest network is used as the starting layout to regenerate the whole network back using force-directed concept. Vertices added to the coarser network N^{k+1} in the refinement phase must retain their position if they are found in N^{k+1} or must have one or more neighbour vertices from N^{k+1} (it means when coarsening is done, the vertices are combined with other vertices) where their position is the average of their neighbours' position.

There are two stop criteria for the coarsening process, each met when using two different coarsening methods respectively. The two types of coarsening methods are edge collapsing method (EC) and maximal independent vertex set (MIVS) coarsening method. The specialty of HU is using a hybrid coarsening approach (Gibson et al., 2014). EC is used initially in coarsening stage but if after the edge collapsing process, the number of remaining

vertices in the coarser network are as shown in Equation (3.4), then the coarsening scheme is changed to MIVS.

In the EC method, adjacent vertices are combined to form a new vertex. The new vertex obtained a weight that indicates the number of vertices combined to form it. The edge also obtained a weight which are a combination of the weight of edges it replaces.

In the MIVS method, the coarsest network is expected to consist of the maximal independent vertex set of the network. It is a set of vertices which are not adjacent in the input network (in other word, independent). It is said to be maximal when adding one more vertex eliminate independency in the set.

The stop criteria for the coarsening process is if there are only two vertices left in the network, or Equation 3.4 is met,

$$\frac{|V^{k+1}|}{|V^{k}|} > 0.75. \qquad \text{Equation 3.4}$$

where V^{k+1} is vertices in N^{k+1} and V^{k} is the vertices in N^{k+1}. When the condition is met, it indicates the network has reach the coarsest state.

Gibson et al. (2014) stated that HU can generate network with more than 100,000 vertices in less than a minute. Comparing it with Walshaw's algorithm (Walshaw, 2006) and FM^3 (Hachul & Jünger, 2005) algorithm, HU is found to be similar in terms of time taken with them but can give a more pleasing layout compared to Walshaw's. They also mentioned that HU generates an output network that has even vertices distribution, although there are some edge crossings in it.

iii. OpenOrd Algorithm (OO)

OpenOrd, henceforth will be abbreviated as OO, is a force-directed layout algorithm based on Fruchterman Reingold algorithm (Fruchterman & Reingold, 1991), a force-directed layout algorithm which implements VxOrd (Davidson et al., 2001). As stated by Kobourov (2013), the networks generated by force-directed layout algorithms are usually aesthetically appealing, display symmetricity, and are usually cross-free for planar networks. However, one of the problems that lead to the creation of OO is the issue where contrarily, the use of force-directed layout algorithms on large real-world graphs often results in visually unappealing layout. Therefore, unlike its preceding force-directed layout algorithms, OO is invented to scale well to large graphs and work well on real-world data (Martin et al., 2011).

To better understand the algorithm, assume N as an undirected network consisting of a set of vertices $\{v_1, v_2, \dots, v_n\}$. Adjacent vertices v_i and v_j are connected by an edge $E\{e_{ij}\}$ and its weight is w_{ij}. OO is meant to generate the layout of a network N in two dimensions. If $x_i = (x_{i,1}, x_{i,2})$ indicates the placement of vertex v_i in the network and Dx_i indicates the density of the points $x_1, \dots, x_n$ around x_i, Equation 3.5 shows the formula for which OO works.

$$min_{x_1,\dots,x_n} \sum_i \left(\sum_j \left(w_{ij}\, d\left(x_i, x_j\right)^2 \right) + Dx_i \right) \qquad \text{Equation 3.5}$$

Like any other force-directed concept, the force equation has an attractive and a repulsive term. The attractive term $\sum_j (w_{ij}\, d(x_i, x_j)^2)$ is to place vertices with heavy edge

weight between them together, encouraging clusters and the repulsive term Dx_i is to repulse the vertices so that they are placed apart.

Initially, the vertices in network N are placed in its original place, and one by one their positions are updated by optimising their attraction force and repulsive force according to Equation 3.5. The optimisation update is done repeatedly for all vertices in the network and is controlled by the simulated annealing schedule.

Simulated annealing schedule has five stages namely liquid, expansion, cool-down, crunch and simmer. In each stage, two potential placements of the vertex are computed, a random placement and a placement calculated as the centre for its neighbouring vertices. The placement which gives the minimum value of the force calculation (attraction force added with repulsive force) is selected for the vertex. Each stage allows different level to which the vertex can move. Expansion stage allows the vertex to move the farthest, followed by the liquid stage. Movement are then limited in the cool-down, crunch and simmer stage so that the network converges to a balanced state.

OO provides edge cutting parameter to the user to control the drawing of the network. The parameter is ranged from 0 to 1. Cutting long edges, or in other word, ignoring long edges in the optimisation step reduces the attractive force and enhance the repulsive force between vertices, thus encouraging clusters in the network. The value 0 indicates no edge cutting in the process. When set to 1, edge cutting in the network is vigorous, and clusters in the graph is separated. Edge cutting increases white spaces in the network thus creating a more appealing drawing.

One advantage of OO is its speed. The algorithm can be run in serial or in parallel. In parallel implementation, each processor is given a unique subset of vertices from the input network. Each processor performs optimisation updates on their own subset of vertices in

parallel and combine the outcomes to generate the output network. For very large networks, OO can work fast by increasing the effective usage of computer's memory.

Apart from that, the running time is also speed up by using multilevel approach, similar to that in HU. The difference is only that the multilevel approach consists of force-directed layout concept added with the simulated annealing stages and the edge cutting procedure. After obtaining the coarsest network, N^n, the layout of N^n is drawn using a default edge cutting value and the standard annealing schedule. In the prolongation and refinement step, the default edge cutting value is used along with a modified annealing schedule (no liquid, smaller expansion and no simmer stage). The new layout for the original network is then drawn using a more vigorous edge cutting along with the simmer stage, so that the network finally converges to a balanced state.

Martin et al. (2011) applied OO on real-world datasets with more than 500,000 vertices and found out that the serial and parallel implementations of the algorithm gave similar results hence approving the parallel version of the algorithm. Both results were more visually appealing compared to its predecessor, the VxOrd algorithm.

Gibson et al. (2014), in their comparison work also found that compared to HU and FMS algorithm, implementing OO gave a clearer overall view of clusters in the graph. The algorithm also is said to generate a visual network that has even vertices distribution and uniform edge, although there are edge crossings in the graph. It also put vertices into clusters readily in the network topology.

iv. Markov Graph Clustering Algorithm (MC)

The Markov graph clustering algorithm, henceforth will be abbreviated as MC, is an algorithm that simulates a random walk on a network to generate clusters (Mohammed & Meira Jr, 2014). The algorithm is based on a concept believing that doing a random walk from one vertex to another vertex in a network is likely to result in doing a walk in a cluster rather than doing a walk between clusters. This is supported by the fact that there are supposed to be more links in a cluster compared to links connecting between clusters. The random walk has the potential to reveal the pattern or direction of the flow of edges in the network, therefore indicating the position of clusters in the network.

Random walk in a network is calculated via Markov Chains. In Markov Chain, a weighted graph is turned into a probability transition matrix as shown in Figure 3.21.

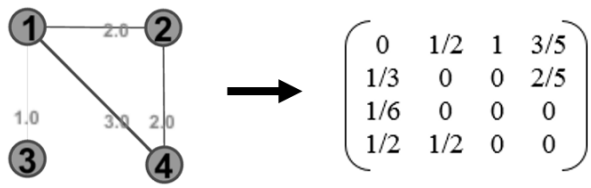


Figure 3.21: Weighted Graph Transformation into Probability Matrix

The weights of edges should be heavier in clusters, and smaller weight is expected for edges of vertices of different clusters. This indicates that there is a connection between the weights shown in the matrix and the location of the clusters in the network.

There are two concepts used in the algorithm to create clusters, which are the flow expansion and flow inflation. The inflation, with parameter *r* enhances or reduces the flow of edges in the network. If it meets a strong edge flow, it enhances the flow. If the edge flow is weak, inflation reduces it more. This is to generate a more obvious clustering. On the other hand, the expansion with parameter *e* allows the flow of the edge to be linked to other areas in the network.

Given an input network *N*, MC initially generates an adjacency matrix for the network. The matrix is then normalised to get a probability transition matrix as shown in Figure 3.21. The values in the matrix are then expanded and inflated (enhanced or weakened) repeatedly until the values in the matrix reach their balanced state. The final values in the matrix reveal clusters in the network. Vertices in the same cluster are in the same column with the same value.

This algorithm scales well with increasing graph size. However, it is said to be unfit for clusters of big size. It has been applied for detection of large-scale protein families (Enright et al., 2002) and give an outstanding result of 95% accuracy when compared with the available protein databases, which are InterPro and SCOP. This algorithm has also been used for various purposes including to detect orthologous genome groups (Li et al., 2003), DNA sequences (Shon et al., 2007), predicting protein complexes from protein interaction networks (Pereira-Leal et al., 2004; Brohee & Helden, 2006), and even to cluster scientific documents (Theodosiou et al., 2008) and IP network traffic analysis (Nataliani & Wellem, 2016).

v. Chinese Whispers Algorithm (CW)

The Chinese Whispers is originally a game for children. A child whispers some words to another child, who repeats the same words to another child, and the whispering continues until the first child listens to his or her original words. The funny part of the game is that often the original words become altered as they pass through the ears of all players. The name of this game was given to a graph clustering algorithm behaving similarly to the game (Biemann, 2006).

The Chinese Whispers algorithm, henceforth will be abbreviated as CW, is a graph clustering algorithm designed to work on weighted and undirected graphs. It is a very basic algorithm but still effective in partitioning vertices (Biemann, 2009). In the algorithm, the vertices of the graph are considered as the children. Initially, unique class labels are assigned to each vertex to indicate their cluster. The vertices of the graph act as if they are whispering their label to their neighbouring vertices like in the game, and the assigned label of a vertex can change by copying the top ranked neighbouring label linked to the vertex. The rank of each neighbouring label is calculated by summing the weights of edges connecting the label to the vertex. For example, if a vertex with Label A is connected to vertices with Label B and Label C, with weights of edges connecting them equal to 2 and 3 respectively, after whispering, Label A will be changed to Label C as it is the top ranked neighbouring label. This indicates that the vertex with Label A before belongs to the same cluster as the vertex with Label C as edges connecting them have more weights compared to the other vertex. This process is repeated until every vertex have been walked through. Considering the walking done, this algorithm is said to be a Markov Clustering algorithm but with a simplified process for updating the clusters of the vertices (Ustalov et al., 2017).

This algorithm is a parameter free method, making its implementation easy. It also has a linear computational complexity. Its processing time increases in linear with the number of vertices in the graph. This enables it to be applied to networks with million number of vertices and edges, where most algorithms considered as unmanageable (Biemann, 2007). However, it is a non-deterministic algorithm. Thus, the cluster results may differ after each running. This problem is more important when processing a small set of vertices where the clusters are more obvious.

Chinese Whispers algorithm is an algorithm that has been more used and cited in natural language processing field than in other areas. It has been used to find groups of POS (part-of-speech) tags in Slavonic language texts (Degorski, 2013) without the use of any manually annotated training sets. It is proven to be able to reduce human help by getting an F-score of about 50-60% on the texts. The result showed that Chinese Whispers scored quite good considering that it is an unsupervised method not needing any reference training set and the complicated morphology of the Polish language. It has also been used in a part of speech tagging system (Biemann, 2009), language separation/identification (Biemann & Teresniak, 2005), word sense induction (Ustalov et al., 2017), and also to group digital news (Pratama et al., 2017).

vi. Girvan-Newman Clustering Algorithm (GN)

Girvan-Newman algorithm (Girvan & Newman, 2002), henceforth will be abbreviated as GN, is a clustering algorithm that was designed to determine community structure (or cluster) in a network using the concept of edge betweenness centrality. The betweenness centrality of an edge coincides with the number of shortest paths between vertices that pass

through the edge (Mihalcea & Radev, 2011). In other words, edge betweenness value is the measure of the total flow that an edge carries. Vertices of high betweenness usually are located in between two densely connected groups of vertices where the vertices of high betweenness are the most sparsely connected to others. The vertices of high betweenness occupy critical roles in the network where it is said to be the "gatekeeper" to the others in the clusters. Figure 3.22 implies such situation.

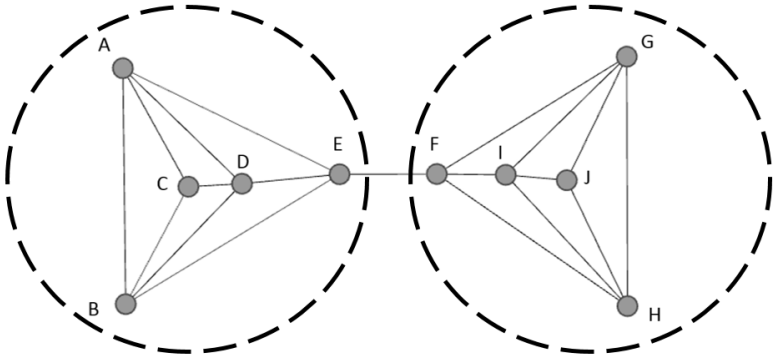


Figure 3.22: Vertices E and F being the "Gatekeeper" of the Two Clusters

According to Matijević et al. (2016), GN is an efficient, well known and one of the most widely applied algorithms to cluster social networks. It works by removing the existing edges between the gatekeeper vertices to create clusters. The edge elimination process is performed iteratively until the edge with the highest betweenness centrality in the network falls below a user-defined threshold.

GN has been applied on Zachary's karate club network data (Newman & Girvan, 2004). The results showed that GN gave high values of modularity, around 0.4 when the network is split into two communities. This showed that there is a strong natural division in the network. The classes are almost the same with the actual division in the club. Only one

vertex has been misclassified when using the shortest-path version of the algorithm, but zero error is found when using the random-walk version of the algorithm making the result a perfect score.

3.3.4 Evaluation of SAGA Network

The evaluation of visual networks deals with seven evaluation metrics, which are, running time, vertices cluttering, edge crossings, relative density, modularity, cluster path length, and conductance as reviewed in Section 2.6.2 and Section 2.6.3. The running time is assessed for layout algorithms as well as graph clustering algorithms. Cluttered vertices and edge crossings are aesthetic criteria measured in the evaluation of the layout. Relative density, modularity, cluster path length and conductance are used to assess the clustered networks.

i. Layout Algorithms

The layout algorithms are evaluated when they are applied to the basic network of SAGA. In total, there are three networks to be evaluated. The evaluation procedure is as shown in Figure 3.23.

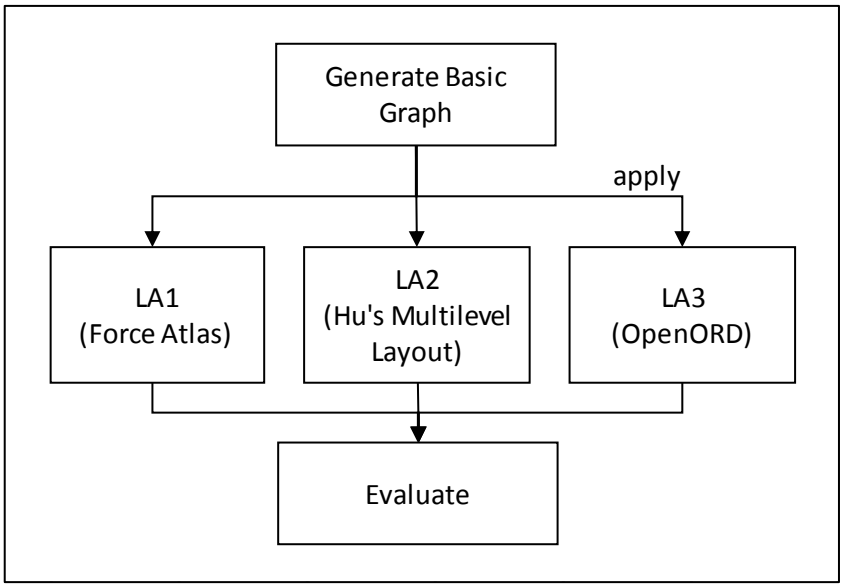


Figure 3.23: Evaluation of Layout Algorithms

The metrics that are used to evaluate the layout algorithms are running time, cluttered vertices and edge crossings. The reason why edge length metric is not used for the evaluation has been explained in Section 2.6.3. The reason why scalability test cannot be performed is further explained here. There is a restriction to the size of our historical data SAGA. Since SAGA is a newspaper reporting events confined to Sarawak, repetitions of NEs tend to happen. Although we have several numbers of editions in hand, it is not enough to collect thousands of NEs to perform scalability test.

ii. Layout+Clustering Algorithms

The clustering algorithms are evaluated on all of the networks generated by the layout algorithms (FA2, HU, and OO). This means that we also hope to observe the effects of the different clustering algorithms on different network layouts. For SAGA, there are nine

networks to be evaluated for layout+clustering algorithms. The evaluation procedure is as shown in Figure 3.24.

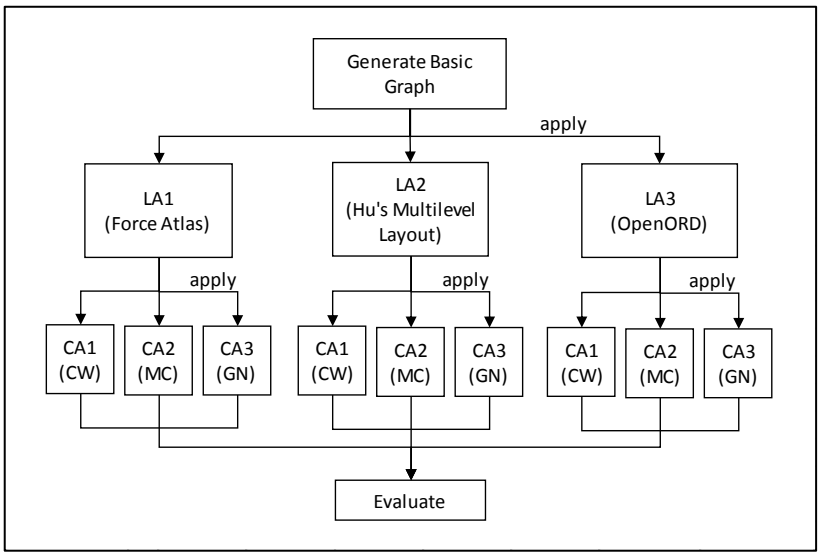


Figure 3.24: Evaluation of Layout+Clustering Algorithms

As discussed in the literature, metrics that are used to evaluate the layout+clustering algorithms are running time, relative density, modularity, cluster path lengths and conductance.

3.4 Applying the Generic Framework on Other Datasets

Once the algorithms have been evaluated on SAGA graph, the proposed framework is also applied to other datasets to evaluate the performance of the combination of algorithms that excelled on SAGA. The objective is to find out whether the algorithms that excelled on SAGA can perform the same when applied on other documents. As mentioned before in Section 3.3, SAGA is identified as a complex-structured dataset because one edition of

SAGA has a small number of vertices but a large number of relations. Two datasets used for this further evaluation are the Biotext Corpus (biological dataset) and dBPedia-Sarawak (city-related facts). The contents of these datasets are different from SAGA, but what is more significant is that the structure of the dataset is different from SAGA. Biotext Corpus has a simple structure where the number of edges and vertices is small and dBPedia-Sarawak dataset is centralised, where the vertices are arranged in a star-like position with only one vertex in the center. Figure 3.25 shows the application of the framework on the two datasets.

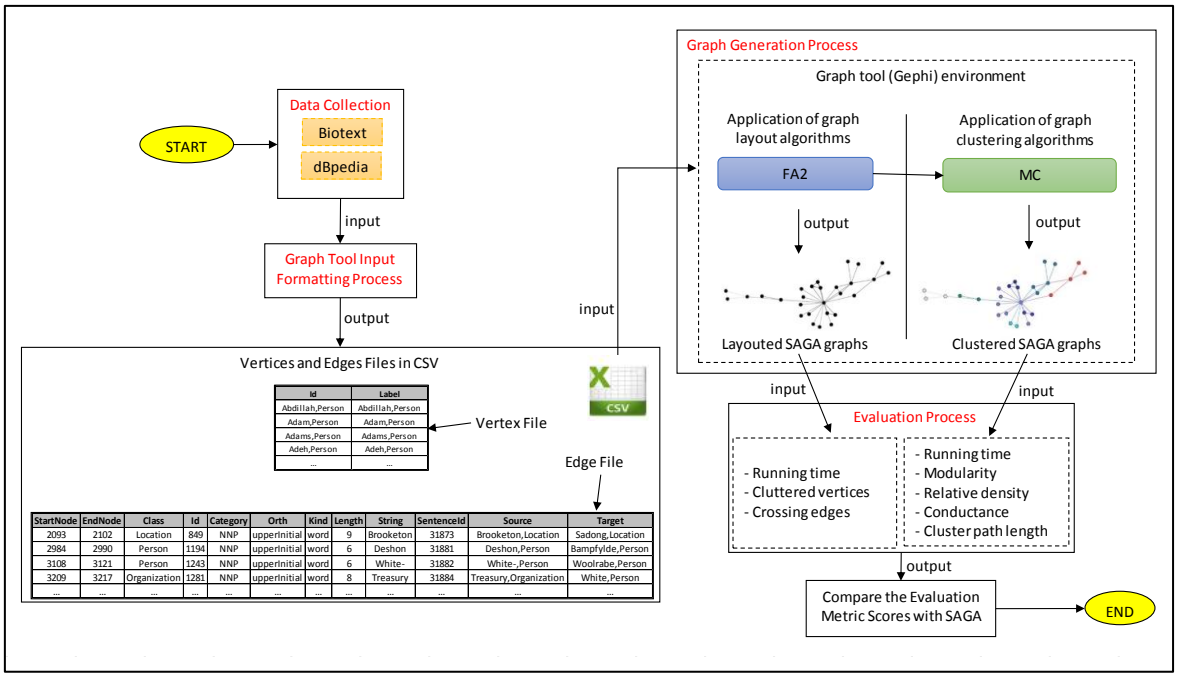


Figure 3.25: Applying Other Documents on the Generic Framework

3.4.1 BioText Corpus: Disease/Treatment Entities

The corpus containing disease/treatment entities (Rosario & Hearst, 2004) is available freely from the BioText Project[10] conducted at University of California, Berkeley. The corpus has not received any name from its creators, and thus in this thesis, it is named "BioText Corpus". The corpus corresponds to the first 100 titles and the first 40 abstracts from the 59 files medline01n*.xml in Medline 2001 (Rosario & Hearst, 2004). The entities (disease and treatment) as well as their relations were annotated manually (Rosario & Hearst, 2004).

The original corpus contains eight relations: "Cure", "Only Disease", "Only Treatment", "Prevent", "Side Effect", "Vague", "Does Not Cure", and "Complex". For this research, only five of these relations are considered: "Cure", "Prevent", "Side Effect", "Vague", and "Does Not Cure". The "Only Disease" and "Only Treatment" relations are discarded from the experiments because they do not show significant relations. The "Complex" relation labelled as <TO SEE> are also excluded since the type of relations that exist have not been identified. The NE types involved in these relations are DIS, DIS_NO, DIS_PREV, DIS_SIDE_EFF, DIS_VAG, TREAT, TREAT_NO, TREAT_PREV, TREAT_SIDE_EFF, and TREAT_VAG. In total, the Biotext corpus used in this research contains 964 sentences, 254 relations, and 185 NEs. The corpus is stored in plain text file. An excerpt of that file is shown in Figure 3.26.

[10] http://biotext.berkeley.edu/data.html

```
The efficiency of <TREAT> testicular sperm retrieval by testicular fine needle aspiration
( TEFNA ) </TREAT> was compared with <TREAT> open biopsy and testicular sperm extraction
( TESE ) </TREAT> , in 37 rigorously selected patients with <DIS> non-obstructive
azoospermia </DIS> .||TREAT_FOR_DIS
Complications included one case of <DIS_SIDE_EFF> testicular bleeding </DIS_SIDE_EFF>
following <TREAT_SIDE_EFF> fine needle aspiration </TREAT_SIDE_EFF> , treated locally ,
and two cases of <DIS_SIDE_EFF> extratunical haematomata </DIS_SIDE_EFF> following TESE
requiring no intervention .||SIDE_EFF
Neither serum FSH values nor testicular size were predictive of the chances to find
spermatozoa for ICSI .||NONE
Some complications may occur even following <TREATONLY> TEFNA </TREATONLY> .||TREATONLY
```

Figure 3.26: Excerpt of BioText Corpus

The annotated entities along with their relations can be acquired without much pre-processing. It only requires rearrangement of the data positions to be turned into a graph. Figure 3.27 shows the data when it is turned into CSV format. 'Source' column contains the vertex where the edge of the network starts from and 'Target' column contains the vertex where the edge ends. 'Type' column contains the characteristic of the edge either it is a directed edge or an undirected edge. 'Id' column contains the names that will be used by Gephi to identify the edges in the network. 'TypeOfRelations' column contains the type of relations indicated by the edges. 'SentenceID' column contains the number of sentence that the relations is found from the document. 'TypeOfSource' column contains the type of the token represented by the source vertex and 'TypeOfTarget' column contains the type of the token represented by the target vertex.

Source	Target	Type	Id	TypeOfRelations	SentenceID	TypeOfSource	TypeOfTarget
macrosomic infants in gestational	good glycemic control	Directed	7621	PREVENT	27	DIS_PREV	TREAT_PREV
corrective surgery	large meningomyelocele	Directed	7622	VAGUE	63	TREAT_VAG	DIS_VAG
corrective surgery	limb aplasia	Directed	7623	VAGUE	63	TREAT_VAG	DIS_VAG
corrective surgery	hypoplasia	Directed	7624	VAGUE	63	TREAT_VAG	DIS_VAG
large meningomyelocele	limb aplasia	Directed	7625	VAGUE	63	DIS_VAG	DIS_VAG
large meningomyelocele	hypoplasia	Directed	7626	VAGUE	63	DIS_VAG	DIS_VAG
limb aplasia	hypoplasia	Directed	7627	VAGUE	63	DIS_VAG	DIS_VAG
methylphenidate	epilepsy	Directed	7628	TREAT_FOR_DIS	188	TREAT	DIS

Figure 3.27: Biotext Data in CSV Format

3.4.2 dBPedia-Sarawak

dBPedia ("dBPedia", n.d.) is an openly available knowledge graph which contains data obtained from Wikipedia, a large-scale, Web knowledge base obtained through open collaborations of users on various topics. Data used in this research are not of all topics, but only topics related to the term "Sarawak", as found in dBPedia. The objective is to obtain a knowledge graph on Sarawak, either historical or modern, as found in dBPedia.

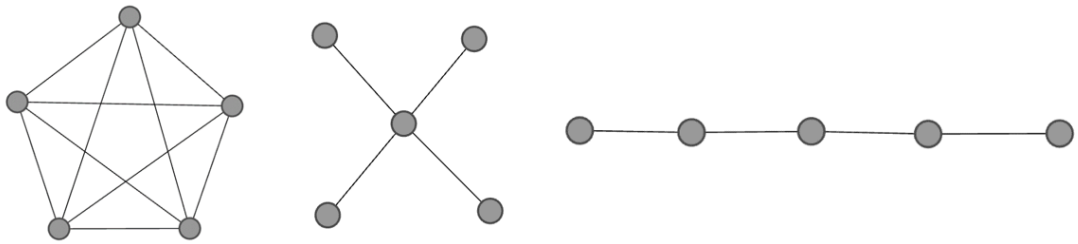
The data are obtained by querying using Simple Protocol and RDF Query Language (SPARQL) in Virtuoso SPARQL Query Editor ("Virtuoso SPARQL Query Editor", n.d.). Data related with Sarawak, either as an Object or as a Subject in their relations are queried. Overall, the data consists of 598 NEs and 741 relations. The data are queried into XML format, therefore only few rearrangements of data positions are required before the data can be turned into a graph. Figure 3.28 shows part of the data. 'Source' column contains the label of the vertex where the edge starts and 'Target' column contains the label of the vertex where the edge stops. 'Label' column contains the label to be displayed for the edge itself and 'Type' column indicates the type of the edge, either it is directed or undirected.

source	label	target	type
Azizan_Saperi	placeOfBirth	Sarawak	undirected
Stephen_Yong_Kuet_Tze	placeOfDeath	Sarawak	undirected
TV_Sarawak	broadcastArea	Sarawak	undirected
INTI_International_University	campus	Sarawak	undirected
Kuching_International_Airport	city	Sarawak	undirected
SS_Vyner_Brooke	country	Sarawak	undirected
Sunny_Hill_School	county	Sarawak	undirected
Lanang_Bridge	crosses	Sarawak	undirected
SCR_(restaurant)	foundationPlace	Sarawak	undirected

Figure 3.28: CSV File Queried from dBPedia Regarding Sarawak

3.5 Simpler Cluster Evaluation Metric (NPL-C Metric)

The fourth research question of this study is whether the many evaluation metrics used reliable? To begin discussion, we take three types of simple clusters with 5 vertices each: (a) X: a perfect cluster, which is a clique consisting of 5 vertices and 10 full edges, (b) Y: a cluster with 5 vertices and 4 edges with one vertex at its centre (star-shaped or centralised) and (c) Z: a cluster with 5 vertices and 4 edges which is linear-shaped. The networks are as shown in Figure 3.29.

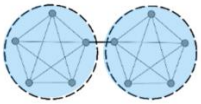
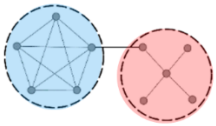
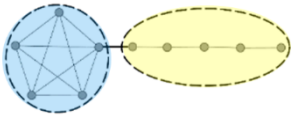


(a) X: A clique (b) Y: Star-shaped (c) Z: Linear-shaped

Figure 3.29: Simple Networks with Five Vertices each

By looking at Figure 3.29, network X is a perfect cluster, as all vertices are connected to each other. Although network Y and Z have the same number of vertices and edges, it is visually accepted that Y is a more compact cluster than Z, because vertices in Z are arranged in a linear pattern. Thus, we know that Y is a better cluster than Z. We use three different networks each made up from the combination of two of these clusters, either cluster X and X, cluster X and Y, or cluster X and Z, as shown in Table 3.2.

Table 3.2: Relative Density, Conductance, Modularity and CPL Scores on Three Networks

Metric Score	**Cluster Combinations**		
	X+X	**X+Y**	**X+Z**
			
Relative Density	1.000	0.700	0.700
Conductance	0.909	0.855	0.855
Modularity	0.452	0.353	0.353
CPL	0.920	0.733	0.670

X is a cluster that is a clique, where it has the maximum amount of edges possible for all of the vertices in it, i.e. very compact. Y and Z on the other hand both have the same number of vertices in them, which is five, and same number of edges which is four. If we are to compare cluster Y and Z, they have different compactness, where vertices in Y is closer to each other (more compact), compared to vertices in Z. Therefore, we can say that Y is a better cluster than Z, and X is the best cluster among the three. That would make X+X the best network, X+Y the second, and X+Z in the last place.

From the calculations in Table 3.2, all metrics give the highest score to network X+X where there are two clusters that are cliques. This indicates that all metrics agreed that network X+X is a network with good clusters, compared to X+Y and X+Z. It is noted that the score given by modularity metric is far lower than others since it indicates the module or communities found in the network. Here, the value of modularity is almost half to 1 (almost 0.5) since network X+X is almost divided into a perfect half, with one edge in between the clusters. Based on its calculation, modularity will only give the value of 1 if there is only one cluster in the network, which is at the same time, a clique in the network, identical to the network shown in Figure 3.29 (a). Modularity of score 0.5 can be obtained if the network consists of two cliques, each identical to the network in Figure 3.29 (a), that are not

connected to each other. Overall, there is no argument that these four metrics can detect good division of clusters in a network where the clusters consist of cliques.

However, when it comes to networks with cluster structure other than cliques, like network X+Y and X+Z, there are disagreements in the scores assigned by the metrics. It is found that modularity, conductance and relative density metrics cannot detect the difference in the structure of cluster Y and cluster Z. Although the star-shaped cluster Y is visually more compact than a linear-shaped cluster Z, they are given the same score. This is because the metrics failed to consider the structure of the cluster in their calculation.

Relative density metric only takes into consideration the internal compactness of each cluster in the network. Therefore, when dealing with cluster Y and Z which both have the same number of edges and vertices, they are given the same score although the connection between their internal vertices are different. Apart from that, this metric also abandons the number of external edges which link the cluster to one another. There could be a case where a cluster is quite compact (have many edges connecting among its internal vertices) but at the same time have many edges connecting it to external vertices. If there are more than one external edge connecting X to Y in network X+Y, the score given will also be the same. Therefore, the lack of important aspects taken into consideration in its calculation can cause relative density metric to judge the quality of clusters with indifferent score.

Conductance metric that is used in this study is external conductance or intercluster conductance. Many works use only the external conductance as the cluster evaluation metric (Almeida et al., 2012). However, external conductance metric only considers intercluster edges while omitting internal edge density in its calculation. This causes the metric to give rating to two clusters of different internal density (same number of vertices and edges but

different way connected) with the same score, as long as they have the same number of intercluster edges, and such is the case for network X+Y and X+Z. Perhaps using combination of external and internal conductance simultaneously would give a more reliable conductance score for cluster quality. Conductance also has a tendency of giving higher scores to networks with fewer number of clusters. This is because network with fewer number of clusters will have a fewer number of edges connecting between its clusters.

Modularity takes into consideration the number of intracluster and intercluster edges in the cluster. However, its calculation failed to differentiate when the structure of how Y and Z are connected although they have the same number of intracluster edges and intercluster edge. When it assigns the highest modularity score to a network, that does not indicate that the structure generated consists of good division of clusters. It may give high value of modularity if the whole network consists of one big portion. The true community structure does not necessarily correspond to the highest modularity.

On the other hand, CPL metric is able to differentiate the different structures in network Y and Z by assigning a slightly higher value for network X+Y and a slightly lower value for network X+Z. We conclude that CPL metric is a more reliable metric when evaluating cluster algorithms as they take into consideration more aspects compared to modularity, relative density and conductance, and is able to differentiate the different structures of networks although they have the same number of vertices and edges. The other metrics have bias in their calculations, making them unable to recognise certain aspect in their score.

The reason why CPL can distinguish internal compactness is because it takes path length into consideration. Path length is the minimum number of edges connecting one node to another node in the network. As explained in Chapter 2, CPL metric has M+ and M-

components. Its M+ component functions to evaluate the internal compactness of the cluster. The best value of path length of a cluster would be 1, i.e. when all vertices in the network is directly connected to one another by an edge, forming a clique. When the path length is normalised, the best value is 1 and the closer the value to 0, it means that the compactness of the cluster is decreasing. This component is what enable it to differentiate network X+X, X+Y and X+Z.

M- component (also called edge penalty) on the other hand evaluates the separateness of two clusters in the network. It is calculated for each pair of clusters in the network by dividing the number of edges connecting the two clusters with the total number of edges possible connecting the two clusters. For edge penalty value, the closer it is to 1, the denser the edges connecting each pair of clusters in the network. $(k(k-1))/2$ represents the total number of cluster pairs available, where k is the number of clusters in the network. If there are only 3 clusters in the network, then the total number of cluster pairs available are 3(3-1)/2 = 3 pairs. For network with 35 clusters, edge penalty calculation must be done for (35*(35-1))/2=595 pairs. This is the drawback of CPL metric, where edge penalty calculation is too complex.

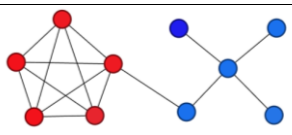
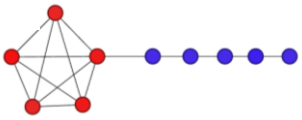
Therefore, a new cluster evaluation metric which can also differentiate the network X+X, X+Y and X+Z but uses a much simpler calculation method is introduced. This metric is given the name NPL-C metric, which stands for "normalised path length – conductance metric". This metric is enough to be used as a stand-alone cluster evaluation metric as its score gives rate to both the internal structure of clusters in the network (compactness) and the external structure of clusters in the network (either the clusters are well separated from one another). Similar to CPL metric, it can distinguish network X+Y and X+Z, as it also uses path length concept. However, the difference is that in NPL-C metric, no calculation

for each pair of clusters are required. This metric makes use of conductance and path length concept.

Based on network conductance calculation by Emmons et al. (2016), conductance of cluster A = number of intercluster edges in A / min (number of edges with endpoints in A, number of edges with no endpoints in A). The conductance of a cluster can range from 0 to 1 where low conductance value indicate that the cluster is well separated and vice versa. In Table 3.3, we use again network X+Y and X+Z. We introduce a new network $X+Y_2$, which also consists of cluster X and cluster Y. The only difference from network X+Y is that in $X+Y_2$ network, cluster X and Y are connected by two extracluster edges.

The average conductance of the three networks are calculated. From Table 3.3, we can see that network X+Y and X+Z have the same average conductance score, indicating that the clusters in the network are equally separated from one another. This is tallied with the illustration of the network where X+Y and X+Z both have only one extracluster edge. On the other hand, network $X+Y_2$ has the highest conductance among the three networks, indicating that its clusters are the least separated from one another. This is also tallied with the illustration of the network where it has two edges connecting X to Y.

Table 3.3: Conductance and Path Length Calculation for $X+Y_2$, X+Y and X+Z

Cluster Combinations		
$X+Y_2$	**X+Y**	**X+Z**
		
Conductance (X)= 2/4 = 0.500 Conductance (Y)= 2/6 = 0.333 Avg. conductance ($X+Y_2$) = 0.417	Conductance (X)= 1/4 = 0.250 Conductance (Y)= 1/5 = 0.200 Avg. conductance (X+Y) = 0.225	Conductance (X)= 1/4 = 0.250 Conductance (Z)= 1/5 = 0.200 Avg. conductance (X+Z) = 0.225
Path length (X) = 1 Path length (Y) = 1.6 Normalised path length(X) = 1 Normalised path length(Y) = 0.625 Avg. normalised path length = 0.8125	Path length (X) = 1 Path length (Y) = 1.6 Normalised path length(X) = 1 Normalised path length(Y) = 0.625 Avg. normalised path length = 0.8125	Path length (X) = 1 Path length (Z) = 2 Normalised path length(X) = 1 Normalised path length(Z) = 0.5 Avg. normalised path length = 0.75

In the table, the path length of each cluster in the network is also calculated. The path length is normalised so that the value falls between the range 0 to 1 where value 1 indicate the most compact clique structure. The normalised path length values are then averaged for each network to get the overall score of clusters compactness in each network. From the calculation, it can be seen that the network $X+Y_2$ and X+Y has the same compactness as they are made of the same clusters, i.e. cluster X and cluster Y. Network X+Z has a lower value for normalised path length as cluster Z is made up of more loosely connected vertices.

Combining the concept of conductance and normalised path length together to produce a score that give rate to both the internal structure and external structure of clusters, we came up with NPL-C metric which is explained further in Figure 3.30.

Step 1: Calculate path length value for each cluster in the network.

Step 2: Normalise path length value of each cluster in a network to set the range to be in [0,1] where 1 is the best value representing cliques.

Step 3: Average the normalised path length values.

Step 4: Calculate conductance for each cluster in the network.

Step 5: Average the conductance values.

Step 6: Averaged normalised path length value – Averaged conductance

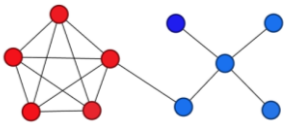
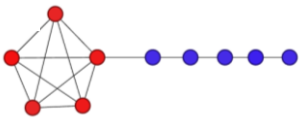
$$\text{NPL-C} = \text{Avg}^{\text{NPL}} - \text{Avg}^{\text{C}}$$

Figure 3.30: NPL-C Metric Calculation

The concept behind NPL-C metric is that a network with a good clustering should have clusters with short path lengths which indicate that their vertices are well-reachable from one another. After normalised, the best path length value will be 1, which is the case when the clusters are clique. On the other hand, the conductance value is small when the clusters are well separated, thus subtracting the averaged conductance from the averaged normalised path length sum up both the internal and external characteristics of clusters in the network. A network with good clustering which consists of compact and well-separated clusters will have NPL-C score closer to 1. The range of NPL-C metric falls within [-1,1] because there is a possibility that a network has clusters which are not very compact with its averaged NPL close to 0, and also are not very well-separated from one another which value is close to 1. Therefore, subtracting 0 from 1 can result in -1.

The benefit of using NPL-C metric is that it is able to distinguish network $X+Y_2$, X+Y and X+Z by assigning them different scores as proven in Table 3.4.

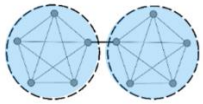
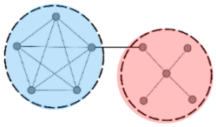
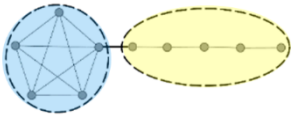
Table 3.4: NPL-C Metric Score for $X+Y_2$, X+Y and X+Z Networks

Cluster Combinations		
$X+Y_2$	**X+Y**	**X+Z**
		
AvgNPL-AvgC = 0.8125 – 0.417 = 0.3955	AvgNPL-AvgC = 0.8125 – 0.225 = 0.5875	AvgNPL-AvgC = 0.75-0.225 =0.525

Comparison of scores made by NPL-C metric and CPL metric is shown in Table 3.5. It can be seen that both CPL metric and NPL-C metric can give the same score pattern where network X+X is the highest, network X+Y is the second highest and network X+Z is the lowest. However, the benefit of using NPL-C metric rather than CPL metric is in its simpler

calculation. For CPL metric, edge penalties need to be calculated for each pair of clusters in the network, but for NPL-C metric, conductance value need to be calculated only for each cluster in the network.

Table 3.5: CPL and NPL-C Metric Scores on X+X, X+Y and X+Z Networks

Metric Score	**Cluster Combinations**		
	X+X	**X+Y**	**X+Z**
			
CPL	0.920	0.733	0.670
NPL-C	0.900	0.588	0.525

3.6 Summary

As a summary, this chapter contains detailed explanation about the proposed generic framework starting from the document filtering to the evaluation stage. Explanation on how the proposed framework is used to perform evaluation of algorithms on SAGA data is also explained. The chapter also introduced the use of the proposed framework on two other datasets, namely Biotext Corpus and dBPedia-Sarawak, which are of different backgrounds and structures to SAGA. This chapter ended with introduction to the simpler cluster evaluation metric proposed in this study.

CHAPTER 4

EXPERIMENTATION: RESULTS AND ANALYSIS

4.1 Overview

In this chapter, the evaluation metric scores are obtained in a semi-automatic way. Gephi provides some attributes which can be obtained automatically, for example the number of vertices in a network, the number of edges in a network, vertex cluster number, and the average path length of clusters in a network. Other attributes and the final score of the metrics are obtained as derivatives from the automatically-obtained attributes. This is made possible because Gephi allow exporting the edge file into excel file. Therefore, the derivative values are calculated using excel formula. For example, the number of inter- and intra-cluster edges, relative density, edge penalty, and etc. Human manual intervention is also needed to perform filtering of unwanted details in the network like uncluttered vertices and uncrossed edges.

4.2 Results and Analysis of Layout Algorithms on SAGA

Figure 4.1 shows the visual networks generated by applying the three graph layout algorithms FA2, OO and HU algorithm on SAGA dataset. The same figures are reported in Appendix B for better view. From the figure, it can be observed visually that vertices in the network produced by FA2 algorithm can be differentiated from one another. Vertices in network produced by OO and HU algorithm are overlapped at certain regions in the network.

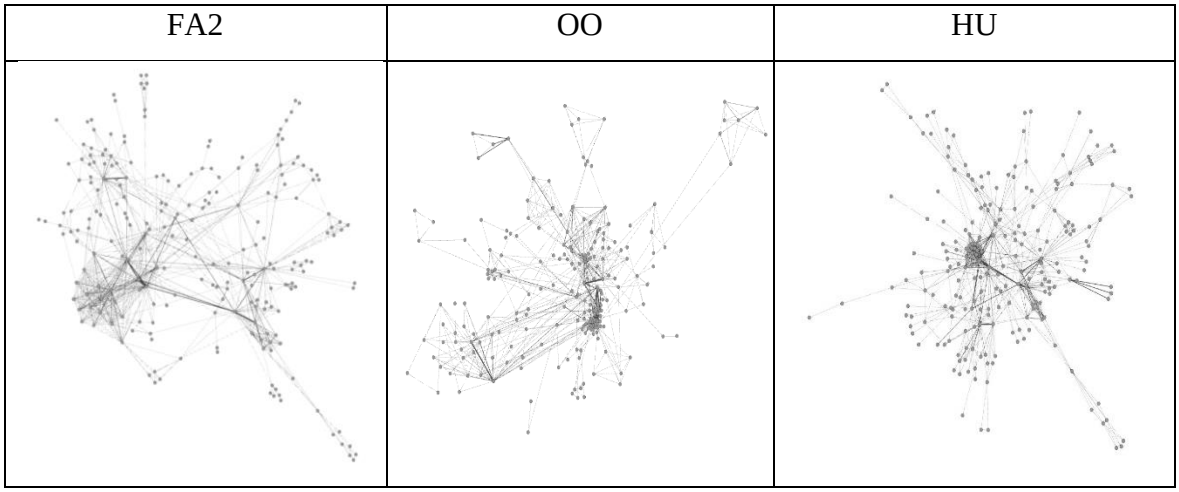


Figure 4.1: SAGA Networks Generated by FA2, OO, and HU

4.2.1 Running Time Results

As shown in Table 4.1, the fastest graph layout algorithm is OO and the slowest is FA2. The running time of FA2 was recorded when there was no more motion of vertices in the network (stable position).

Table 4.1: Running Time of FA2, OO, and HU on SAGA

Algorithm	Running Time (min)
OO	00:06.87
HU	00:13.13
FA2	Unlimited; stable position at 01:04.98

OO, which is not a continuous layout, as expected, took the shortest time to generate SAGA network. This is due to the multilevel approach and parallel implementation in OO. These features are also the reason OO is claimed to be invented to overcome the issue where most force directed algorithms cannot scale well to large networks (Martin et al., 2011). Based on the result, the algorithm is proven to be fast but unfortunately, our SAGA network

is not large enough to provide a proof in term of scalability. The coarsening of the SAGA network by HU before prolongating the network into the original network according to force-directed principle is proven to take less time than the technique used in FA2. FA2 took the longest time however. Since FA2 is a continuous layout, it will only stop on users' request. However, it is observed that actually the basic drawing of the network has been accomplished way before the stable position is achieved. But there are still a few numbers of vertices which oscillate until the SAGA network is stable (when none of the vertices are moving), which prolongs the time measured for FA2. This is explained by Gibson et al. (2014) where FA2 maximises speed until it is clear that some vertices are unstable, so it then slows down to emphasise on precision. Here, speed is meant to be sacrificed for greater precision of determining the vertices position. Therefore, we cannot just conclude that the fastest algorithm OO generates the best layout. Yet, other metrics needed to be examined further.

4.2.2 Edge Crossings Results

Based on Table 4.2, ironically, FA2 has shown that it surpasses OO and HU in term of generating network with less edge crossings. Out of 839 edges in SAGA network, FA2 is able to generate a network with minimal 655 edge crossings. HU came second, followed by OO.

Table 4.2: Edge Crossings Generated by FA2, HU and OO on SAGA

Algorithm	No of edge crossings
FA2	655
HU	732
OO	771

At best, only 184 edges are not crossed with each other, but it is emphasised here that SAGA is a complex network, where each of its vertex has an average degree of seven. An edge crossing free network is impossible to be generated unless all relations in SAGA form a circle, which is not the case for SAGA. The NEs in SAGA have many connections to each other since SAGA is a newspaper reporting repetitively different events but of the same locations or persons, as long as they were in Sarawak during that period. However, this still shows that the longer running time taken by FA2 is beneficial in term of making sure that the layout it generated is positioned precisely.

OO, although is the fastest to run, is a modified force-directed algorithm which emphasise more on being fast so that it is able to generate large networks. However, being fast often means to sacrifice precision and vice versa (Gibson et al., 2014). The aim of the invention of OO is also to emphasise on the presence of clusters in networks. In OO, vertices repulse each other, but adjacent vertices are also attracted to each other. However, here, in order to emphasise clustering, positioning adjacent vertices closer to one another tend to make the edges of those vertices to cross each other more. The less iteration of the algorithm also results in a fast decision, where the vertices are placed with less accuracy. Unfortunately, this caused the edges difficult to be distinguished by viewers. Here we observed that there is an inversely proportional benefit to the values indicated by running time and edge crossing metrics.

4.2.3 Cluttered Vertices Results

Again, FA2 scored the best as the network it generated contains zero cluttered vertices, as shown in Table 4.3. HU again ranked second, followed by OO.

Table 4.3: Cluttered Vertices Results on SAGA

Algorithm	Cluttered Vertices	No of Cluttered Vertices
FA2	No cluttered vertices. Positions of vertices are very clearly visible.	0
HU	Some vertices at the centre overlapped.	14
OO	Many cluttered vertices	45

Remarkably there is no vertex cluttering at all in FA2 although there are still some unnecessary edges overlapping taking place in the network. This is because of the algorithm's principle which make sure that all vertices pull away from each other (thus no cluttering of vertices) but are still kept together by the edges between them.

The abundant cluttered vertices in OO can be explained by the aim of the invention of the algorithm itself, which is to emphasise on the presence of clusters in network. Unfortunately, the vertices in the resulting network are hard to be distinguished by the viewers. Noverlap algorithm (an algorithm in Gephi to separate vertices so that it will not overlap) needed to be applied to the network in order for each vertex to be clearly viewed.

4.2.4 Overall Results

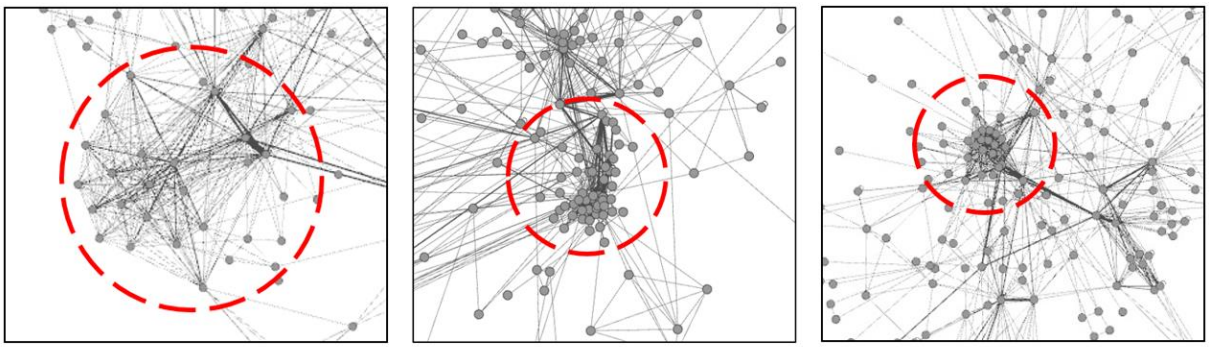
Although FA2 is a continuous algorithm and it only stops when asked by the user, the total running time it took to finalise the position of the vertices is too far slower when compared. However, in term of the layout of the network it produced, remarkably there is no vertex cluttering at all although there are still some unnecessary edges overlapping taking place in the network. This is because of the algorithm's principle which make sure that all vertices pull away from each other (thus no cluttering of vertices) but are still kept together by the edges between them. However, the overlapping is considered medium when compared to edges in networks generated by OO and HU.

OO scored the fastest in its running time compared to the others. However, in term of the layout of the network it generated, there are many vertices cluttering and edges overlapping, the worst compared to FA2 and HU. The cluttering of the vertices can be explained by the aim of the invention of the algorithm itself, which is to emphasise on the presence of clusters in network. Unfortunately, the vertices in the resulting network are hard to be distinguished by the viewers. When dealing with SAGA, a historical data that has widely interconnected relations (where there are many situations where one vertex is related to a big number of vertices at one time), the overlapping and clutteredness of edges and vertices became worse.

On the other hand, HU has its running time only slightly slower compared to OO but still faster compared to FA2. However, in term of the layout of the network it produced, it wins over OO as the layout it produced is more appealing compared to OO, with less edge crossings and less vertices overlapping.

There is one critical region in SAGA network where the NEs in the document are very inter-related with each other, causing the region representing those NEs in the network denser than other area.

Figure 4.2 shows the area where FA2 is able to represent the vertices less cluttered and easier to be viewed, compared to OO and HU, which represent the vertices in a single clutter.



(a) FA2 (b) OO (c) HU

Figure 4.2: Region of the Same NEs of SAGA in Different Network Layouts

For layout algorithms, FA2 algorithm is found to generate networks with layouts that are most pleasant to the eye, with

- No cluttered vertices
- Less edges overlapping

Although FA2 have the longest running time with 01:04.98 minutes, the time taken can still be tolerable when looking at the layout it generated. Since the aim of the research is to find out which layout algorithm can give a more understandable and easier to interpret network, a network with no cluttered vertices and least edges overlapping is chosen over as the preferred one.

4.3 Results and Analysis of Clustering Algorithms on SAGA

Figure 4.3 shows the clustered networks generated by CW, MC and GN, each on the network layouts generated by FA2, HU and OO. The nine figures in Appendix C shows the larger version of the networks for reference.

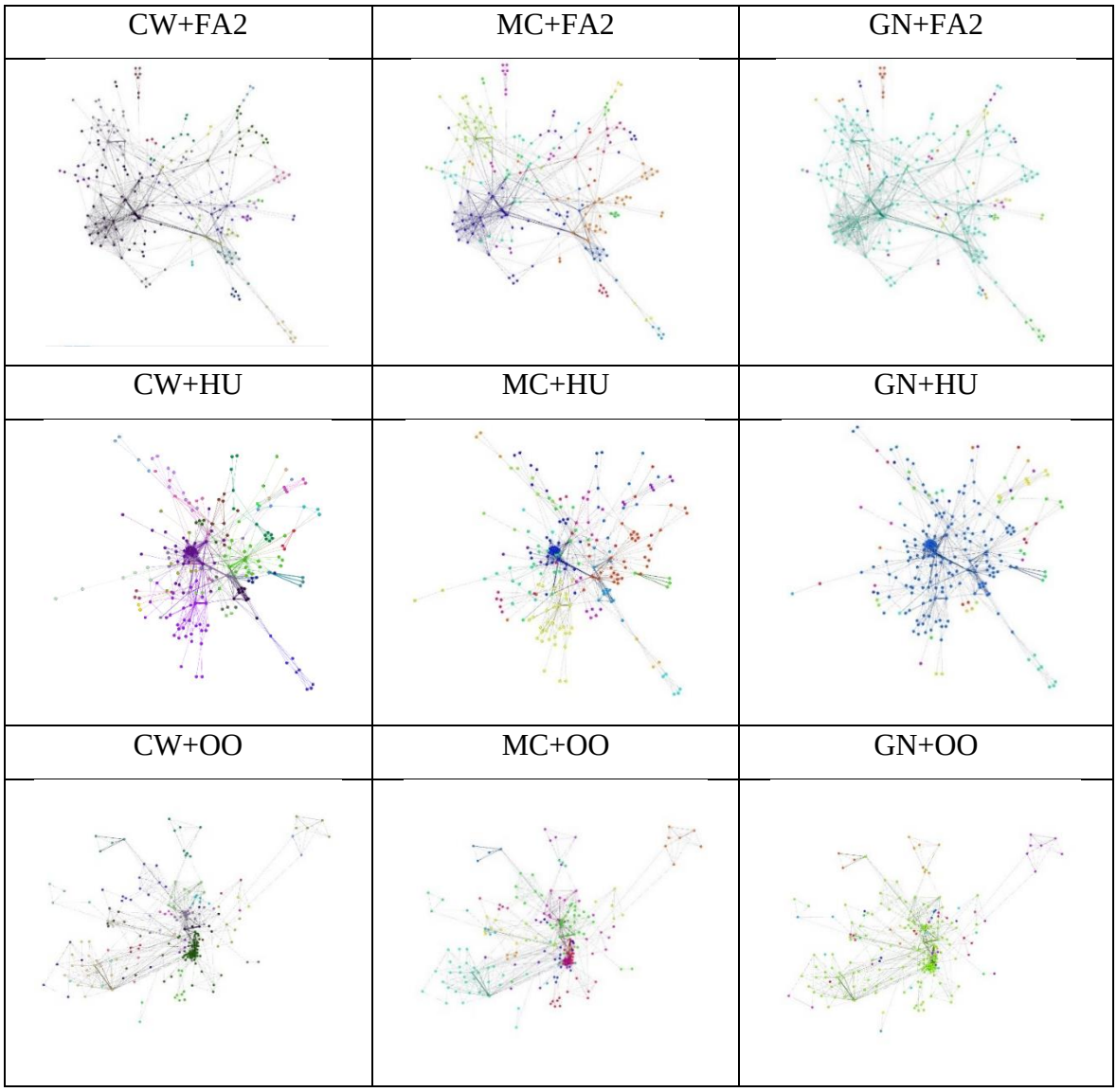


Figure 4.3: Clustered Networks Generated by CW, MC and GN

When observed visually in Figure 4.3, it is difficult to identify network with good clustering. Therefore, evaluation is required to get the evaluation metrics score. However, it is noticed that in networks generated by CW algorithm, the clustering is not consistent. The clusters may differ (different in term of number of clusters and the cluster members) each time the algorithm is applied on the network.

This inconsistency is because of CW's random decision whenever it encounters a vertex that has equal connection to two clusters. CW assigns such vertices to any of the two cluster randomly, in which this decision changes every time ran. Such situation is shown in Figure 4.4, where 'Wang,Person' is clustered together with 'Highness,Title' on the first time ran, but then clustered together with 'Hose,Person' when ran for the second time. Both networks are the same layout generated by FA2, only that CW is applied twice to it.

a) Vertex 'Wang,Person' is clustered together with vertex 'Highness,Title'

b) Vertex 'Wang,Person' is clustered together with vertex 'Hose,Person'

Figure 4.4: A Vertex with Equal Number of Relations to Two Different Clusters

In order to ensure that the evaluation metric for CW is calculated fairly and conclusion is not made based on a 1-time run only, CW is applied 10 times on each of the network, and the metric scores are calculated by their average.

On the other hand, metric scores obtained by MC and GN did not change whenever ran on the same layout of SAGA. This is because the clusters they generated is the same every time applied. Therefore a 1-time run is enough for MC and GN.

Although ran 10 times and average scores are taken, the metric values scored by CW on different layouts do not vary much neither. The scored results for each metric are explained further in the following sections.

4.3.1 Running Time Results

From Table 4.4 it is seen that the running time of each graph clustering algorithm did not differ much although applied to different layouts of SAGA network.

Table 4.4: Running Time of All Evaluated Algorithms on SAGA

Algorithms	Running Time
CW+FA2	00:00.43
CW+HU	00:00.72
CW+OO	00:00.75
MC+FA2	00:06.60
MC+OO	00:06.74
MC+HU	00:07.46
GN+FA2	00:57.31
GN+HU	00:59.62
GN+OO	01:01.12

It is observed that CW, when combined with any layout algorithm always remain the first to generate clusters on SAGA. It takes almost instantly. MC is ranked second, followed by GN with recorded time up to 1 minute and 1.12 seconds.

Compared to MC and GN, CW does not perform heavy calculations to produce clusters, making it fast and suitable for large networks. MC on the other hand, can generate clusters slightly seconds slower than CW. GN would require roughly about a minute different from the two algorithms. Therefore, to obtain immediate clusters, CW and MC would be a good choice.

4.3.2 Modularity Results

Modularity results of each clustering algorithm also show consistency in their scores regardless of the layout of the SAGA networks, as shown in Table 4.5.

Table 4.5: Modularity Results on SAGA

Algorithms	Modularity
CW+OO	0.5639
CW+FA2	0.5535
CW+HU	0.5476
MC+FA2	0.5366
MC+OO	0.5366
MC+HU	0.5366
GN+FA2	0.1466
GN+OO	0.1466
GN+HU	0.1466

Overall, CW scored the highest with modularity ranging from 0.5639 to 0.5476, followed by MC. GN, however scored far lower than the other two algorithms. Their scores in modularity showed that among the three, CW can generate clusters in SAGA with better division, in a sense that there are more edges within the clusters (or community) and only a few between them. GN is still competitive with CW as their score difference is very small. The modularity values of GN however are very low and it indicates that between the clusters generated by GN, there are still many edges between them, implying that NEs that are related to each other might have been separated into different groups by the algorithm.

4.3.3 Relative Density Results

In this metric, CW scored the highest with score ranging from 0.7921 to 0.7697. MC ranked second followed by GN, as shown in Table 4.6.

Table 4.6: Relative Density Results on SAGA

Algorithms	Relative Density
CW+HU	0.7921
CW+FA2	0.7857
CW+OO	0.7697
MC+FA2	0.7418
MC+OO	0.7418
MC+HU	0.7418
GN+FA2	0.4794
GN+OO	0.4794
GN+HU	0.4794

Their relative density score showed that CW can generate clusters which are most compact, among the three. It indicates that in each of the clusters that CW generated, the vertices are almost to forming a clique. In a much simpler word, if one vertex can only have one edge to each vertex in the cluster (a one-to-one relation), there is less possibility for us to draw an edge to cross from one vertex to another vertex in the cluster because there is already such an edge between them.

4.3.4 Conductance Results

From Table 4.7, again, we see a repetitive pattern in ranks in this metric where CW scored the highest with value ranging from 0.6873 to 0.6658, followed by MC. GN again, scored the lowest.

Their scores in conductance showed that in each of the clusters generated by CW, there is a smaller number of edges that needed to be removed in order to separate the cluster from other clusters surrounding it. This score is quite related with modularity score. If modularity score is high, conductance score should be high at the same time. This is because

of the fact that if the clusters generated are of high modularity, the number of edges between the clusters must have been very few at the same time.

Table 4.7: Conductance Results on SAGA

Algorithms	Conductance
CW+FA2	0.6873
CW+OO	0.6888
CW+HU	0.6658
MC+FA2	0.5571
MC+OO	0.5571
MC+HU	0.5571
GN+FA2	0.2296
GN+OO	0.2296
GN+HU	0.2296

4.3.5 CPL Results

Ironically, we see a change of pattern in the first and second rank in CPL results where MC scored the highest, followed by CW, as shown in Table 4.8. GN however, still scored the lowest.

Table 4.8: CPL Results on SAGA

Algorithms	CPL
MC+FA2	0.7622
MC+OO	0.7622
MC+HU	0.7622
CW+FA2	0.7506
CW+HU	0.7504
CW+OO	0.7487
GN+FA2	0.4321
GN+OO	0.4321
GN+HU	0.4321

Their scores in CPL metric on the other hand indicate that among the three graph clustering algorithms evaluated, MC can generate clusters with the shortest average path length and least number of inter cluster edges. It also indicates that CW, despite the highest score in the other cluster evaluation metrics, ranked second when it comes to its CPL. Interestingly, this metric evaluates two aspects at one time. And the other metrics which evaluate only one aspect do not agree with the value of CPL metric.

If we are to analyse based on the results obtained, we can say that among the three graph clustering algorithms evaluated, when given a network, CW can generate clusters in an immediate amount of time (almost instantly). MC on the other hand, can generate clusters slightly seconds slower than CW. GN would require roughly about a minute different from the two algorithms. Therefore, to obtain immediate clusters, CW and MC would be a good choice.

Their scores in modularity showed that among the three graph clustering algorithms evaluated, CW can generate clusters which has the better division, in a sense that there are more edges within the clusters (or community) and only a few between them. The modularity values of GN however are very low and it indicates that between the clusters generated by GN, there are still many edges between them, implying that vertices that are related to each other might have been separated into different groups by the algorithm.

Their relative density score showed that CW can generate clusters which are most compact, among the three. It indicates that in each of the clusters that CW generated, the vertices are almost to forming a clique. In a much simpler word, if one vertex can only has one edge to each vertex in the cluster (a one-to-one relation), there is less possibility for us to draw an edge to cross from one vertex to another vertex in the cluster because there is already such an edge between them.

Their scores in conductance showed that in each of the clusters generated by CW, there is less number of edges that needed to be removed in order to separate the cluster from other clusters surrounding it. This score is quite related with modularity score. If modularity score is high, conductance score should be high at the same time. This is because of the fact that if the clusters generated are of high modularity, the number of edges between the clusters must have been very few at the same time.

Their scores in CPL metric on the other hand indicate that among the three graph clustering algorithms evaluated, MC can generate clusters with the shortest average path length and least number of inter cluster edges. It also indicates that CW, despite the highest score in the other cluster evaluation metrics, ranked second when it comes to its CPL. Interestingly, this metric evaluates two aspects at one time. And the other metrics which evaluate only one aspect do not agree with the value of CPL metric.

4.3.6 NPL-C Results

Interestingly, NPL-C metric agreed with modularity, relative density and conductance in its score, causing CPL metric alone to rate MC as the best clustering algorithm. As shown in Table 4.9, it is seen that for NPL-C metric, CW scored the highest, followed by MC. GN, consistently, still scored the lowest.

Both CPL and NPL-C metric can differentiate the internal structure (compactness) of a cluster whereas the other metrics cannot. We can say that both metrics are reliable to differentiate the internal and external structure of a cluster whereas other metrics are not reliable enough. However, we can see that CPL and NPL-C metric gave different ranking to the algorithms, where CPL rank MC as first, but NPL-C ranks CW as first. The reason behind this is because they use different concept to rate the external structure of the clusters.

Table 4.9: NPL-C Results on SAGA

Algorithms	CPL
CW+OO	0.2940
CW+FA2	0.2937
CW+HU	0.2677
MC+FA2	0.2430
MC+OO	0.2430
MC+HU	0.2430
GN+FA2	-0.3528
GN+HU	-0.3528
GN+OO	-0.3528

CPL and NPL-C both subtract the value of a second aspect from the average normalised path length of clusters in the network. The second aspect is either edge penalties for CPL or conductance for NPL-C. The difference in their score is because of the difference in the value of these second aspects. CPL subtract edge penalties which is actually the density of intercluster edges connecting each pair of clusters when compared to the possible edges to connect between them. On the other hand, NPL-C subtracts the score indicating the well-separation of the clusters.

CPL ranks MC as first because the intercluster edges density in the clusters generated by MC is lower. But, NPL-C ranks CW first because clusters generated by CW is more well-separated than clusters generated by MC.

However, CW is a random algorithm. The results obtained for CW was actually averaged after running for 10 times. This is because of its random mechanism, where if it meets a vertex which have equal number of edges belonging to two different clusters, it assigns the vertex to any of the clusters randomly. MC on the other hand assign clusters based on the flow of information in the network – a more reliable concept. According to CPL and NPL-C metric, both MC and CW generate good clusterings for SAGA. The score

of MC and CW are not ranked very far from each other. However, since CW rather shows an unstable result of clustering (sometimes better than MC, sometimes not), therefore to get an always good clustering of SAGA, MC algorithm is preferred. A combination of FA2 and MC is chosen to generate a good network of SAGA.

4.4 Further Evaluation of the Chosen Algorithms for SAGA on Different Datasets

The evaluation is further carried on using datasets of different structure in order to answer the third research question of this study, i.e. can the selected algorithms for the historical documents perform the same for other datasets too? In this section, we want to find out whether the combination of the algorithms selected to perform best on SAGA (which is FA2+MC) can produce the same good layout and good cluster division again. Appendix D shows the resulting networks for the algorithms on the two datasets. Table 4.10 and Table 4.11 show the comparison of the performance of FA2 and MC on the datasets respectively.

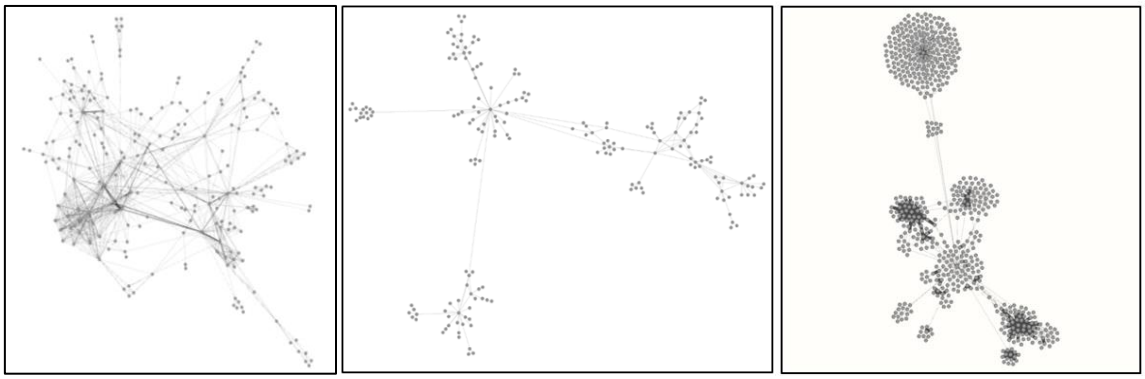
Table 4.10: Evaluation Metric Scores of FA2 on SAGA, Biotext and dBPedia

Data	Running Time (min)	No of Edge Crossings	No of Cluttered Vertices
SAGA	Unlimited; stable position at 01:04.98	655	0
Biotext	Unlimited; stable position at 00:52.71	21	0
dBPedia	Unlimited; stable position at 01:21.91	230	0

The results showed that FA2 generates the network of Biotext slightly faster than the network of SAGA and dBPedia. This is expected since the Biotext network is the simplest

network among the three with the smallest number of vertices and edges (185 vertices and 255 relations). The time taken to generate SAGA and dBpedia do not differ much. Although SAGA (224 NEs) has fewer vertices compared to dBPedia (598 NEs), it seems like the number of relations in the data is the one affecting FA2's running time. dBpedia which has 741 relations required slightly less time than SAGA, which has 839 relations to be generated. However, the important discussion here is on the other two metrics, namely edge crossings and cluttering of vertices.

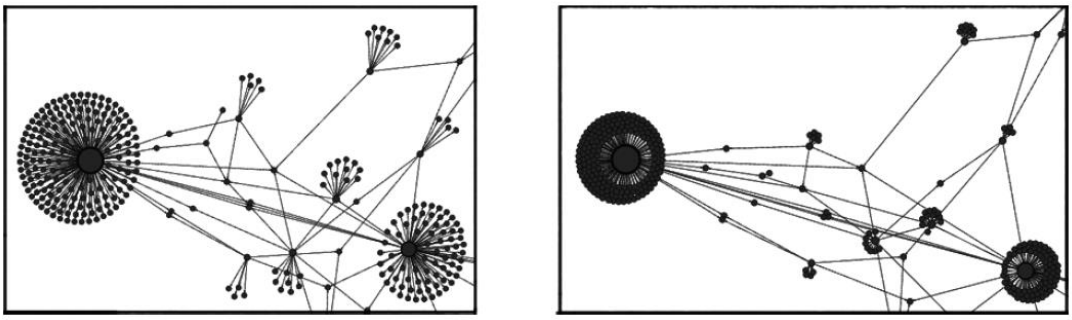
Compared to SAGA, FA2 produce network which is pleasant to the eyes on Biotext data, where the vertices and edges are easily differentiated from each other, with very few edge crossings as shown in Figure 4.5 (b). This is due to the simple structure of the data itself. Biotext data is a collection of abstracts of medical journals formed into a network. An abstract of a journal consists of concise explanation about a topic and contains less repetitive words, hence Biotext network has simple relations in it. Since cross-related vertices are less, FA2 is able to generate a clearly visible network with no vertex cluttering and very few edge crossings, unlike in SAGA (Figure 4.5 (a)). The larger version of the networks for Biotext and dBPedia datasets can be found in Appendix D.



(a) SAGA (b) Biotext (c) dBpedia

Figure 4.5: Networks Generated by FA2 on each Dataset

However, on dBPedia data, FA2 failed to prevent 230 edges from crossing each other. Although it is less than that of in SAGA network, interestingly dBPedia network is generated with a few hairball-like structures, thus decreasing the readability of the relations in the network, as can be seen in Figure 4.5 (c). This is because of the structure of the dataset where a few vertices have significantly higher degree than other vertices and the other vertices are centralised on them. We refer dBPedia as a centralised dataset. It is found that FA2 tend to generate hairballs whenever it meets dataset with centralised structure. As reported by Jacomy et al. (2011), it is an intended characteristic of FA2 when it meets vertices with many leaves. FA2 intentionally change its vertices repulsion concept to place poorly connected leaves closer to its parent vertex which has high degree in its output network. This is because the leaves are considered as a source of visual cluttering and placing them closer together is claimed to reduce the visual cluttering in the network. However, in our network, this makes it difficult to differentiate between vertices and edges. Even in the network reported by Jacomy et al. (2011) which is shown in Figure 4.6, the centralised-structured network generated by FA2 is difficult to read compared to the output of a force-directed algorithm with normal repulsion concept, Fruchterman Reingold Algorithm. This algorithm was invented in 1991 with the aim to distribute vertices evenly, making the edge lengths in the network to be uniform and to produce networks with symmetricity in it (Fruchterman, 1991).



(a) Fruchterman Reingold (b) FA2

Figure 4.6: Output of a Force-Directed Algorithm with Normal Repulsion Concept Compared to FA2 (Jacomy et al., 2011)

From the overall results, it can be concluded that FA2 perform well on simple (Biotext) and complex (SAGA) network, but it is not the case with centralised network (dBPedia). However, it must be noted that our experiment does not include any large dataset, as the largest size consist of only 598 (dBPedia) in terms of vertices or 839 in term of relations (SAGA).

The resulting networks generated by applying MC on the network of Biotext and dBPedia are included in Appendix E. Table 4.11 shows the scores of the evaluation metrics calculated on the networks.

It can be seen that there are great changes in the scores for different documents. All metric scores changes depending on the aspects that they measured. Documents with different structure have different characteristics that affect certain score. The results of each score varied based on the structure of the documents.

Table 4.11: Evaluation Metric Scores of MC on SAGA, Biotext and dBPedia

Dataset	Number of Cluster	Cluster Evaluation Metric					
		Running Time	Modularity	CPL	Relative Density	Conductance	NPL-C
SAGA	29	00:06.60	0.5366	0.7622	0.7418	0.5571	0.2430
Biotext	37	00:42.26	0.6898	0.7898	0.6871	0.7185	0.3744
dBPedia	39	00:03.39	0.7433	0.7480	0.5798	0.7590	0.3685

Modularity is affected by the number of intercluster edges in the graph. Modularity is low in SAGA because SAGA documents has a complex structure. It is said to be complex as the NEs in SAGA are very interrelated to one another, which cause the network to have many intercluster edges. The score is highest in dBPedia because it has very few intercluster edges.

Relative density score is affected by the density of intracluster edges in each cluster. Relative density is low in dBPedia because clusters in dBPedia have centralised structure, where each cluster have one vertex acting as the cluster center and all other vertices in the cluster are connected solely to it. – thus, making the density of the intracluster edges to become sparse. Relative density is highest in SAGA because there are many edges in SAGA. Relative density metric only takes into consideration the internal edge densities of the clusters. Therefore, although the relationships in SAGA are quite complex causing many edges to connect between clusters, the complexity of the edges connection actually helps increase the intracluster edges density too.

CPL is affected by both the path length of the clusters and the density of external edges between each pair of clusters. CPL score is the highest in the simple Biotext data but the lowest in dBPedia. Although clusters in dBPedia are very well-separated from one another, but because of its centralised structure, having a single node as the centre of each cluster has caused the path length between vertices in the cluster to increase, thus lowering its CPL score.

Conductance is affected by the number of intercluster edges in the graph. As expected, conductance score is low in SAGA because of its very interrelated NEs, causing many intercluster edges in the network. Conductance score is high in dBPedia because each of its centralised cluster has very few intercluster edges connecting between them.

NPL-C is affected by both the path length of the clusters and the conductance or separation between clusters in the graph. NPL-C is low in SAGA because the interrelated edges in SAGA reduce the separation between the clusters. It is also low in dBPedia because the centralised structure increase the path lengths between vertices in the structure. It is highest in Biotext network because both aspects measured are in medium score – clusters with medium average path length, mediumly separated clusters.

The important point here is that, all metric scores changes depending on the aspects that they measured. Documents with different structure have different characteristics that affect certain score. For example, complex documents with interrelated NEs like SAGA might cause the clusters to have high score for metric which consider the intercluster edges density to define a good cluster. Centralised structured document like dBPedia may cause the clusters to have large value of path lengths, thus affecting score which take into consideration path length value.

4.5 Summary

In overall, the study has conducted evaluation of three graph layout algorithms and three graph clustering algorithms on SAGA dataset using the proposed general framework. The evaluation found out that FA2 algorithm when combined with MC algorithm is able to produce a network with a good layout and a good clustering for SAGA. The evaluation also demonstrates that the scores given by evaluation metrics can disagree with one another as they each are invented based on different opinions on how to indicate a good cluster. Further evaluation of FA2 and MC algorithms conducted on Biotext and dBPedia dataset using the proposed framework implied that the performance of an algorithm, be it a layout or a clustering algorithm, actually depends on the structure of the document itself. Therefore, for a new document, to find the right algorithms to represent it, the evaluation will need to be done again as the results can be different. Therefore, it is significant to have each steps of the evaluation process concluded in a detailed framework. There is also a significance to ensure that the evaluation metric used is reliable and easy to be conducted. This is because often for real world networks, we are dealing with a large number of clusters and vertices. Therefore, a reliable but easy-to-conduct evaluation metric is required to ensure that the evaluation process is feasible. The proposed NPL-C metric, based on path length and conductance concept, is reliable yet simple to be used for any future clustering algorithm evaluation.

CHAPTER 5

CONCLUSION

5.1 Overview

Representations of historical documents into network is not just to deliver the contents faster. It is also supposed to ease the interpretation made by the viewer and to be meaningful as well. As an overall conclusion, this research proposed a generic evaluation framework to perform evaluation of both graph layout and clustering algorithms to find a network which has a good layout and a good clustering for a given document. In this study, the best representation of SAGA visual network is determined by using the proposed generic framework to evaluate three graph layout algorithms and three clustering algorithms on SAGA. From the discussion, it is found that the combination of FA2 and MC algorithm can generate a network with a good layout and good clustering on SAGA network. We have also discussed about the reliability and complexity of the evaluation metrics used and applied our proposed cluster evaluation metric which is reliable but simpler compared to the existing metrics. Based on the result of the evaluation, it is also demonstrated that the scores given by the evaluation metrics can disagree with one another as they each was invented based on different opinions on how to indicate a good cluster. Apart from that, the proposed framework is again used to evaluate FA2 and MC algorithm on Biotext and dBPedia dataset to find out whether they can perform well again when applied to documents with structure different from SAGA. From the conducted evaluation, it is found that using the same selected algorithms on SAGA does not necessarily produce the same results on other datasets. This is because the performance of a layout or a clustering algorithm actually

depends on the structure of the document itself. Therefore, evaluations of algorithms on documents are ineluctable whenever we want to represent a new document in the form of a network. Hence, there is a significance in the proposed generic framework so that it can be reused for future researches regardless of the type of documents we are working with. Since evaluations are ineluctable, it is also important to ensure that the evaluation metric used is reliable and at the same time, simple. This is because often we need to evaluate large real-world networks which contain many edges and vertices. The proposed NPL-C metric is proven to be both reliable compared to conductance, relative density and modularity metrics, yet simpler than the existing reliable CPL metric. Therefore, it allows an easy-to-conduct evaluations to be performed in the future.

5.2 Contributions

This study has three research contributions (RCs) in total:

- Able to propose a generic framework to do evaluation of both graph layout and clustering algorithms to find a good network representation for a document.

  The study proposed a generic framework which functions to perform evaluation of network algorithms to find a good network representation for a document. This framework suggests to evaluate both graph layout and clustering algorithms in order to get a good network for a document. The framework is modular in characteristic so that any part of it is able to be replaced with new module, to suit the situation. The framework has been practiced in the study to evaluate 1) document that need to be annotated (for example in the study, SAGA), and 2) documents that does not need annotation (for example in the study, dBPedia and Biotext datasets).

- Able to produce a representation of SAGA in the form of a readable and effective visual network.

  At the end of the study, SAGA is able to be turned into a network which has a good layout, which eases the viewer to distinguish the vertices and edges in the network. Apart from that, the network also consists of good clustering where the clusters generated are well-separated from one another and the vertices in each cluster are closely connected to one another.

- Able to propose an effective and simpler evaluation metric for clustering algorithms called NPL-C metric.

  In the study, a new cluster evaluation metric named NPL-C metric has been introduced and this metric is able to evaluate both the internal and external structure of clusters. It is able to be used as a stand-alone evaluation metric and at the same time is a derivative from the existing CPL metric and conductance metric. The metric uses the concept of average path length and conductance in its calculation.

Figure 5.1 shows the relations between the ROs discussed in Chapter 1 with the RCs. From the figure, it can be seen that the first and the third ROs of this study have been achieved by our second RC. On the other hand, RO2 and RO4 each has been achieved by RC1 and RC3, as depicted in the figure.

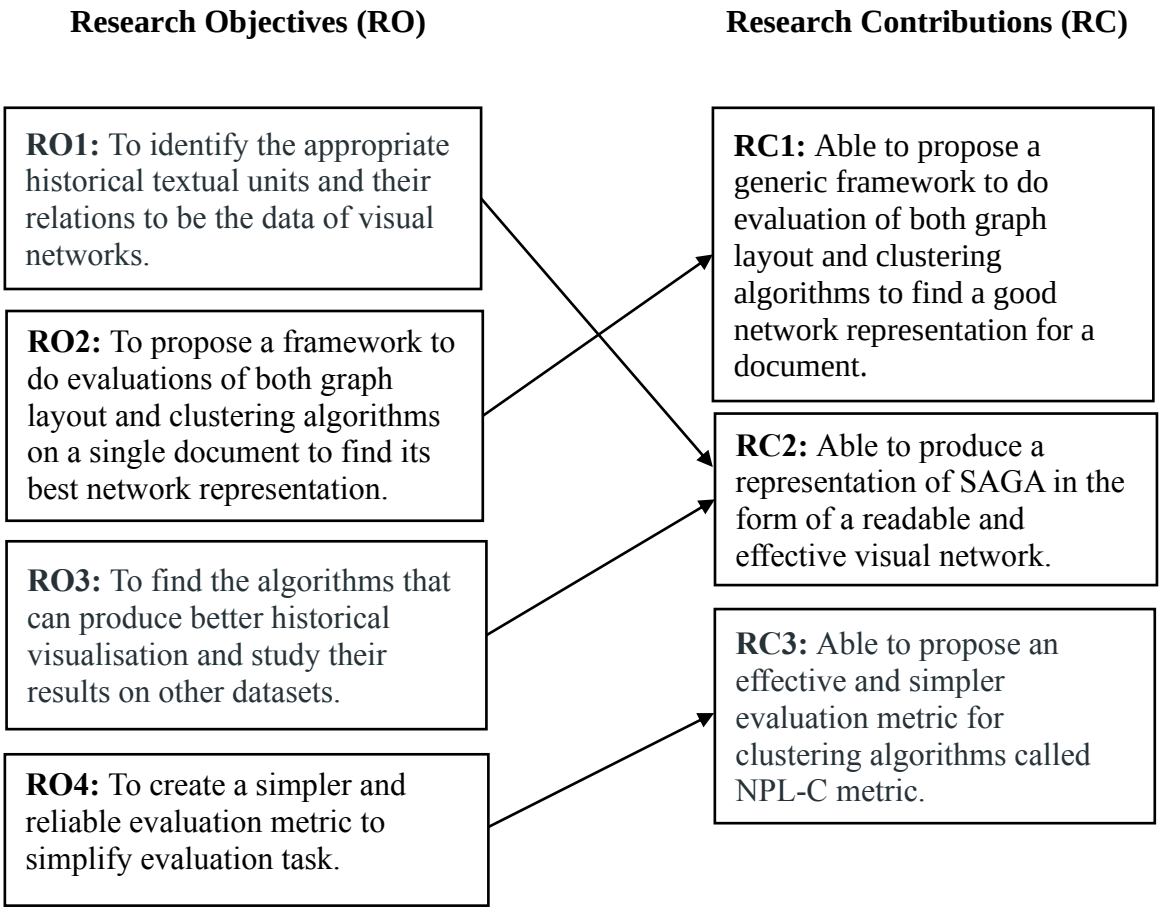


Figure 5.1: Relations between ROs and RCs of the Study

5.3 Research Limitations

In the study, the quality of the clustering generated by the graph clustering algorithms can only be evaluated empirically using metrics. Cluster content analysis by doing comparison on the actual cluster groups in SAGA data cannot be carried out as we did not acquire any a priori information on the groups in SAGA.

In evaluating the algorithms, the algorithms are evaluated based on their implementation in Gephi. When the selection on the suitable tools to be used is done, Gephi is chosen since it provides both graph layout and clustering algorithms as its features. This

is to reduce the time taken to implement the six algorithms and to proceed to the evaluation stage where the discussion really matters.

5.4 Future Works

For the future work, four possible studies are proposed. The first one is to create an automatic method to generate metric scores from the output network. This is to ease the evaluation process. Currently, there is no existing graphing tool with the feature to generate automatically evaluation metric scores. Apart from that, further development on the graph layout algorithms should be done to produce an algorithm that is not only good for a certain structure of network but also good for network with any structure. The selected algorithm for SAGA and the simpler cluster evaluation metric proposed, NPL-C are hoped to be applied and used on other datasets to further explore their effectiveness. Graphing tools also need to be improvised and extended to include clustering algorithms as most of them currently do not provide clustering options for the users.

REFERENCES

Aggarwal, C. C., & Wang, H. (2010). A Survey of Clustering Algorithms for Graph Data. In *Managing and Mining Graph Data* (Aggarwal, C. C., & Wang, H., eds), pp. 275-301. New York: Springer.

Ahn, Y. Y., Bagrow, J. P., & Lehmann, S. (2010). Link Communities Reveal Multiscale Complexity in Networks. *Nature*, *466*(7307), 761-764.

Almeida, H., Guedes, D., Meira, W., & Zaki, M. J. (2011). Is There a Best Quality Metric for Graph Clusters? In *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 44-59.

Almeida, H., Neto, D. G., Meira, W., & Zaki, M. J. (2012). Towards a Better Quality Metric for Graph Cluster Evaluation. *Journal of Information and Data Management*, *3*(3), 378-393.

Barnes, J., & Piet, H. (1986). A Hierarchical O(N log N) Force-Calculation Algorithm. *Nature, 324*(6096), 446-449.

Bastian, M., Heymann, S., & Jacomy, M. (2009). Gephi: An Open Source Software for Explorating and Manipulating Networks. In *Proceedings of International AAAI Conference on Web and Social Media*, San Jose, pp. 361-362.

Biemann, C. (2006). Chinese Whispers - an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proceedings of TextGraphs: The First Workshop on Graph Based Methods for Natural Language Processing*, pp. 73-80.

Biemann, C. (2007). *Unsupervised and Knowledge-free Natural Language Processing in the Structure Discovery Paradigm*, Germany: Leipzig University.

Biemann, C. (2009). Unsupervised Part-of-speech Tagging in the Large. *Research on Language and Computation*, *7*(2-4), 101-135.

Biemann, C., & Teresniak, S. (2005). Disentangling from Babylonian Confusion - Unsupervised Language Identification. In *Proceedings of Computational Linguistics and Intelligent Text Processing*, pp. 773-784.

Brandes, U. (2001). A Faster Algorithm for Betweenness Centrality. *Journal of Mathematical Sociology*, *25*(2), 163-177.

Brandes, U., & Erlebach, T. (2005). *Network Analysis: Methodological Foundation*s, Heidelberg: Springer.

Brandes, U., Gaertler, M., & Wagner, D. (2003). Experiments on Graph Clustering Algorithms. In *Proceedings of European Symposium on Algorithms*, pp. 568-579.

Brohée, S., & Helden, J. (2006). Evaluation of Clustering Algorithms for Protein-protein Interaction Networks. *BMC Bioinformatics*, *7*(1), 488-506.

Buchheim, C., Chimani, M., Gutwenger, C., Jünger, M., & Mutzel, P. (2013). Crossings and Planarisation. In *Handbook on Graph Drawing and Visualisation (*Tamassia, R., Ed.), pp. 43-85. Boca Raton: CRC Press.

Carpano, M. J. (1980). Automatic Display of Hierarchised Graphs for Computer Aided Decision Analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, *10*(11), 705-715.

Chen, C. (2006). *Information Visualisation: Beyond the Horizon*, London: Springer-Verlag London.

Chimani, M., Hedtke, I., & Wiedera, T. (2019). Exact Algorithms for the Maximum Planar Subgraph Problems: New Models and Experiment. *Journal of Experimental Algorithmics*, *24*(1), 2-1.

Clancy, K., Haythorpe, M., & Newcombe, A. (2019). An Effective Crossing Minimisation Heuristic Based on Star Insertion. *Journal of Graph Algorithms and Applications, 23*(2), 135-166.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms,* 3rd ed., Cambridge, Massachusetts: The MIT Press.

Davidson, G. S., Wylie, B. N., & Boyack, K. W. (2001). Cluster Stability and the Use of Noise in Interpretation of Clustering. In *Proceedings of IEEE Symposium on Information Visualisation*, pp. 23-30.

dBPedia. (n.d.). [online] Available at: https://wiki.dbpedia.org/ [Assessed on 29 January 2020]

Degórski, Ł. (2013). Fine-tuning Chinese Whispers Algorithm for a Slavonic Language POS Tagging Task and its Evaluation. In *Proceedings of the 6th Language and Technology Conference (LTC 2013)*, pp. 439-443.

Di Battista, G., Eades, P., Tamassia, R., & Tollis, I. G. (1994). Algorithms for Drawing Graphs: An Annotated Bibliography. *Computational Geometry, 4*(5), 235-282.

Di Battista, G., Eades, P., Tamassia, R., & Tollis, I. G. (1998). *Graph Drawing: Algorithms for the Visualisation of Graphs*. Prentice Hall PTR.

Di Giacomo, E., Liotta, G., & Tamassia, R. (2018). Graph drawing. In *Handbook of Discrete and Computational Geometry* (Goodman, J. E., O'Rourke, J., & Tóth, C. D., Eds.), pp. 1451-1478. Boca Raton: CRC Press.

Djidjev, H. N., & Onus, M. (2013). Scalable and Accurate Algorithm for Graph Clustering. *IEEE Transactions on Parallel and Distributed Systems, 24*(5), 1022-1029.

Dubey, A., & Sharma, S. (2013). Comparison of Graph Clustering Algorithms. *International Journal of Computer Trends and Technology, 4*(9), 3230-3235.

Duncan, C. A., & Goodrich, M. T. (2013). Planar Orthogonal and Polyline Drawing Algorithms. In *Handbook of Graph Drawing and Visualisation* (Tamassia, R., Ed.), pp. 223-246. Boca Raton: CRC Press.

Dunne, C., Ross, S. I., Shneiderman, B., & Martino, M. (2015). Readability Metric Feedback for Aiding Vertex-link Visualisation Designers. *IBM Journal of Research and Development, 59*(2/3), 1-16.

Eades, P. (1984). A Heuristic for Graph Drawing. *Congressus Numerantium*, *42*, 149–160.

Eades, P. (2015). *Graph Visualisation: Topology-Shape-Metric.* [online] Available at: http://edbt2015school.win.tue.nl/material/EadesLectures/planar_v1.pdf [Assessed on 29 January 2020]

Eades, P., & Klein, K. (2015). Graph Visualisation. In *Proceedings of Graph Data Management*, pp. 33-70.

Emmons, S., Kobourov, S., Gallant, M., & Börner, K. (2016). Analysis of Network Clustering Algorithms and Cluster Quality Metrics at Scale. *PLoS ONE*, *11*(7), 1-18.

Enright, A. J., Dongen, S. V., & Ouzounis, C. A. (2002). An Efficient Algorithm for Large-Scale Detection of Protein Families. *Nucleic Acids Research, 30*(7), 1575-1584.

Enright, A. J., Kunin, V., & Ouzounis, C. A. (2003). Protein Families and TRIBES in Genome Sequence Space. *Nucleic Acids Research, 31*(15), 4632-4638.

Foggia, P., Sansone, C., & Vento, M. (2001). A Performance Comparison of Five Algorithms for Graph Isomorphism. In *Proceedings of 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition*, pp. 188-199.

Frick, A., Ludwig, A., & Mehldau, H. (1994). A Fast Adaptive Layout Algorithm for Undirected Graphs. In *Proceedings of International Symposium on Graph Drawing*, pp. 388-403.

Fruchterman, T. M., & Reingold, E. M. (1991). Graph Drawing by Force-Directed Placement. *Journal Software-Practice & Experience, 21*(11), 1129-1164.

Garg, A., & Tamassia, R. (2001). On the Computational Complexity of Upward and Rectilinear Planarity Testing. *Society for Industrial and Applied Mathematics Journal on Computing, 31*(2), 601-625.

Gibson, H., Faith, J., & Vickers, P. (2014). A Survey of Two-Dimensional Graph Layout Techniques for Information Visualisation. *Information Visualisation, 12*(3-4), 324-357.

Girvan, M., & Newman, M. E. (2002). Community Structure in Social and Biological Networks. In *Proceedings of National Academy of Science of the United States of America*, pp. 7821-7826.

Grobelnik, M., & Mladenic, D. (2004). Visualisation of News Articles. *Informatica, 28*(4), 375-380.

Gutwenger, C., & Mutzel, P. (2003). An Experimental Study of Crossing Minimisation Heuristics. In *Proceedings of 11th International Symposium on Graph Drawing,* pp. 13-24.

Hachul, S., & Jünger, M. (2005). Drawing Large Graphs with a Potential-Field-Based Multilevel Algorithm. *Graph Drawing 2004, 3383,* 285-295.

Hachul, S., & Jünger, M. (2006). An Experimental Comparison of Fast Algorithms for Drawing General Large Graphs. In *Proceedings of International Symposium on Graph Drawing,* pp. 235-250.

Hall, K. M. (1970). An r-Dimensional Quadratic Placement Algorithm. *Management Science, 17*(3), 219-229.

Hamasuna, Y., Ozaki, R., & Endo, Y. (2017). A Study on Cluster Validity Measures for Clustering Network Data. In *Proceedings of Joint 17th World Congress of International*

Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS2017), pp. 410-415.

Harrison, C. (2007). *Bible Cross References*. [online] Available at: http://www.chrisharrison.net/index.php/Visualizations/BibleViz [Assessed on 29 January 2020]

Hartigan, J. A., & Wong, M. A. (1979). A k-Means Clustering Algorithm. *Journal of the Royal Statistical Society (Applied Statistics)*, *28*(1), 100-108.

He, H., Sýkora, O., & Vrt'o, I. (2005). Heuristic Crossing Minimisation Algorithms for 2-page Drawings. *Electronic Notes in Discrete Mathematics*, *22*, 527-534.

Healy, P., & Nikolov, N. S. (2013). Hierarchical Graph Drawing. In *Handbook of Graph Drawing and Visualisation* (Tamassia, R., & Tamassia, R., Ed.), pp. 409-453. Boca Raton: CRC Press.

Hearst, M. A. (2009). *Search User Interfaces*, New York: Cambridge University Press.

Hinrichs, U., Alex, B., Clifford, J., Watson, A., Quigley, A., Klein, E., & Coates, C. M. (2015). Trading Consequences: A Case Study of Combining Text Mining and Visualisation to Facilitate Document Exploration. *Digital Scholarship in the Humanities, 30*(1), i50-i75.

Hu, Y. (2006). Efficient, High-Quality Force-Directed Graph Drawing. *The Mathematica Journal, 10*(1), 37-71.

Hua, J., Huang, M. L., & Wang, G. (2018). Graph Layout Performance Comparisons of Force-Directed Algorithms. *International Journal of Performability Engineering, 14*(1), 67-76.

Huang, W. (2013). *An Aggregation-Based Overall Quality Measurement for Visualisation*. [online] Available at: https://arxiv.org/abs/1306.2404 [Assessed on 29 January 2020]

Itoh, M., & Akaishi, M. (2012). Visualisation for Changes in Relationships between Historical Figures in Chronicles. In *Proceedings of the International Conference on Information Visualisation*, pp. 283-290.

Jacomy, M. (2014). *benchmarkForceAtlas2*. [online] Available at: https://github.com/medialab/benchmarkForceAtlas2/blob/master/dataset.csv [Assessed on 29 January 2020]

Jacomy, M., Venturini, T., Heymann, S., & Bastian, M. (2014). ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualisation Designed for the Gephi Software. *PLoS ONE, 9*(6), e98679.

Jianbo, S., & Jitendra, M. (2000). Normalised Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 888-905.

Kannan, R., Vempala, S., & Vetta, A. (2004). On Clusterings: Good, Bad and Spectral. *Journal of the ACM, 51*(3), 497-515.

Kar, G. M., & Gilbert, R. S. (1988). Heuristic Layout Algorithms for Network Management Presentation Services. *IEEE Network, 2*(6), 29-36.

Kasik, D. J., Ebert, D., Lebanon, G., Park, H., & Pottenger, W. M. (2009). Data Transformations and Representations for Computation and Visualisation. *Information Visualisation, 8*(4), 275-285.

Khokhar, D. (2015). *Gephi Cookbook*, Birmingham: Packt Publishing.

Knuth, D. E. (1963). Computer-Drawn Flowcharts. *Magazine Communications of the ACM, 6*(9), 555-563.

Kobourov, S. G. (2012). *Spring Embedders and Force Directed Graph Drawing Algorithms*. [online] Available at: https://arxiv.org/abs/1201.3011 [Assessed on 29 January 2020]

Kobourov, S. G. (2013). Force-Directed Drawing Algorithms. In *Handbook of Graph Drawing and Visualisation* (Tamassia, R., Ed.), pp. 383-408. CRC Press.

Koren, Y. (2003). On Spectral Graph Drawing. In *Proceedings of the 9th Annual International Conference on Computing and Combinatorics*, pp. 496-508.

Krebs, V. (1996). Visualising Human Networks. *Release, 1*, 1-25.

Kruja, E., Marks, J., Blair, A., & Waters, R. (2001). A Short Note on the History of Graph Drawing. In *Proceedings of Graph Drawing*, pp. 222-286.

Li, L., Stoeckert Jr., C. J., & Roos, D. S. (2003). OrthoMCL: Identification of Ortholog Groups for Eukaryotic Genomes. *Genome Research*, *13*(9), 2178-2189.

Liang, M., Gao, C., Li, X., & Zhang, Z. (2017). An Enhanced Markov Clustering Algorithm Based on Physarum. In *Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 486-498.

Lizinski, V., Sell, G., Jansen, A. (2015). An Evaluation of Graph Clustering Methods for Unsupervised Term Discovery. In *Proceedings of International Speech Communication Association (INTERSPEECH)*, pp. 3209-3213.

Martin, S., Brown, W. M., Klavans, R., & Boyack, K. (2011). OpenOrd: Open-Source Toolbox for Large Graph Layout. In *Proceedings of SPIE 7868, Visualisation and Data Analysis*, pp. 23-27.

Matijević, T., Vujičić, T., Ljucović, J., Radunović, P., & Balota, A. (2016). Performance Analysis of Girvan-Newman Algorithm on Different Types of Random Graphs. In *Proceedings of Central European Conference on Information and Intelligent Systems*, pp. 11-16.

Mi, P., Sun, M., Masiane, M., Cao, Y., & North, C. (2016). Interactive Graph Layout of a Million Vertices. *Informatics, 3*(4), 23.

Miao, Q., Zhang, S., Zhang, B., Meng, Y., & Yu, H. (2012). Extracting and Visualising Semantic Relationships from Chinese Biomedical Text. In *Proceedings of the 26th Pacific Asia Conference on Language, Information and Computation*, pp. 99-107.

Michailidis, G. (2008). Data Visualisation Through their Graph Representations. In *Handbook of Data Visualisation* (Chen, C. -H., Härdle, W, & Unwin, A., Eds.), pp. 103-120. Berlin Heidelberg: Springer.

Mihail, M., Gkantsidis, C., Saberi, A., & Zegura, E. (2002). *On the Semantics of Internet Topologies*. USA: Georgia Institute of Technology.

Mihalcea, R., & Radev, D. (2011). *Graph-Based Natural Language Processing and Information Retrieval*. Cambridge University Press.

Mohammed, Z. J., & Meira Jr, W. (2014). *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press.

Mueller, C., Gregor, D., & Lumsdaine, A. (2006). Distributed Force-Directed Graph Layout and Visualisation. In *Proceedings of the 6th Eurographics Conference on Parallel Graphics and Visualisation*, pp. 83-90.

Nascimento, M. C., & Carvalho, A. C. (2011). A Graph Clustering Algorithm Based on a Clustering Coefficient for Weighted Graphs. *Journal Braz Comput Soc, 17*(1), 19-29.

Nataliani, Y., & Wellem, T. (2016). HTTP Traffic Graph Clustering using Markov Clustering Algorithm. *International Journal of Computer Applications, 90*(2), 37-41.

Newman, M. E., & Girvan, M. (2003). Mixing Patterns and Community Structure in Networks. In *Statistical Mechanics of Complex Networks* (Pastor-Satorras, R., Rubi, M., & Diaz-Guilera, A., Eds.), pp. 66-87. Berlin: Springer.

Newman, M. E., & Girvan, M. (2004). Finding and Evaluating Community Structure in Networks. *Physical Review E, 69*(2), 026113.

Nikolov, N. S. (2016). Sugiyama algorithm. In *Encyclopedia of Algorithms* (Kao, M. -Y., Ed.), pp. 2162-2166. Berlin, Heidelberg: Springer.

Osaki, T., Itsubo, S., Kimura, F., Tzuka, T., & Maeda, A. (2011). Visualisation of Relationships Among Historical Persons Using Locational Information. In *Proceedings of International Symposium on Web and Wireless Geographical Information Systems,* 230–239.

Patrignani, M., & Vargiu, F. (1997). 3DCube: A Tool for Three Dimensional Graph Drawing. In *Proceedings of the Symposium on Graph Drawing (GD'97),* pp. 284-290.

Penna, G. D., & Orefice, S. (2016). Supporting Information Extraction from Visual Documents. *Journal of Computer and Communications, 4,* 36-48.

Pereira-Leal, J. B., Enright, A. J., & Ouzounis, C. A. (2004). Detection of Functional Modules from Protein Interaction Networks. *PROTEINS: Structure, Function and Bioinformatics*, *54*(1), 49-57.

Poranen. T. (2006). Two New Approximation Algorithms for the Maximum Planar Subgraph Problem. *Acta Cybernetica, 18*(3), 503-527.

Pratama, M. F. E., Kemas, R. S. W., & Anisa, H. (2017). Digital News Graph Clustering using Chinese Whispers Algorithm. In *Proceedings of Journal of Physics: Conference Series*, pp. 1-7.

Purchase, H. (1997). Which Aesthetic Has the Greatest Effect on Human Understanding? In *Proceedings of the 5th International Symposium on Graph Drawing,* pp. 248–261.

Raper, S. (2012). *Graphing the History of Philosophy*. [online] Available at: http://www.coppelia.io/2012/06/graphing-the-history-of-philosophy/. [Assessed on 29 January 2020]

Rosario, B., & Hearst, M. A. (2004). Classifying Semantic Relations in Bioscience Texts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pp. 431-438.

Santanu, S. R. (2013). *Graph Theory with Algorithms and its Applications: In Applied Science and Technology*, Springer Science & Business Media.

Schaeffer, S. E. (2007). Graph Clustering. *Computer Science Review*, *1*(1), 27-64.

Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., & Ideker, T. (2003). Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks. *Genome Research*, *13*(11), 2498-2504.

Shawn, G., Milligan, I., & Weingart, S. (2013). *Networks in Historical Research*. [online] Available at: http://www.themacroscope.org/?page_id=308 [Assessed on 29 January 2020]

Shih, Y.-K., Kim, S., Shi, T., & Parthasarathy, S. (2013). Directional Component Detection via Markov Clustering in Directed Networks. In *Proceedings of the 11th Workshop on Mining and Learning with Graphs*, pp. 1-6.

Shon, H. S., Kim, S., Rhee, C. S., & Ryu, K. H. (2007). Clustering Approach using MCL Algorithm for Analysing Microarray Data. *International Journal of Bioelectromagnetism*, *9*(2), 65-66.

Smith, M. A., Shneiderman, B., Milic-Frayling, N., Rodrigues, E. M., Barash, V., Dunne, C., Capone, T., Perer, A. & Gleave, E. (2009). Analysing (Social Media) Networks with NodeXL. In *Proceedings of the Fourth International Conference on Communities and Technologies*, pp. 255-264.

Song, S., & Zhao, J. (2014). *Survey of Graph Clustering Algorithms Using Amazon Reviews*. USA: Stanford University.

Sugiyama, K., Tagawa, S., & Toda, M. (1981). Methods for Visual Understanding of Hierarchical System Structures. *IEEE Transactions on Systems, Man, and Cybernetics, 11*(2), 109-125.

Tamassia, R. (1987). On Embedding a Graph in the Grid with the Minimum Number of Bends. *Society for Industrial and Applied Mathematics Journal on Computing, 16*(3), 421–444.

Tamassia, R. (1998). Constraints in Graph Drawing Algorithms. *Constraints, 3*(1), 87-120.

Tan , D. Y., Ranaivo-Malançon, B., & Kulathuramaiyer, N. (2015). Wiki SaGa: An Interactive Timeline to Visualise Historical Documents. In *Information Science and Applications* (Kim, K. J. Ed.), pp. 705-712. Berlin Heidelberg: Springer.

Tarawneh, R. M., Keller, P., & Ebert, A. (2011). A General Introduction to Graph Visualisation Techniques. In *Proceedings of the International Research Training Group 1131 (IRTG 1131) - Visualisation of Large and Unstructured Data Sets Workshop*, pp. 151–164.

Teshome, H. L. (2013). *Evaluation of network comparison approaches*, Linnaeus University.

Theodosiou, T., Darzentas, N., Angelis, L., & Ouzounis, C. A. (2008). PuReD-MCL: A Graph-Based PubMed Document Clustering Methodology. *Bioinformatics, 24*(17), 1935–1941.

Thomas, J. J., & Cook, K. A. (2005). Data Representations and Transformations. In *Illuminating the Path: The Research and Development for Visual Analytics* (Thomas, J. J., & Cook, K. A., Eds.), pp. 105-136. Los Alamitos, CA: IEEE Computer Society Press.

Tikhonova, A., & Ma, K.-l. (2008). A Scalable Parallel Force-Directed Graph Layout Algorithm. In *Proceedings of the 8th Eurographics Conference on Parallel Graphics and Visualisation*, pp. 25-32.

Tollis, I. G., & Xia, C. (1994). Drawing telecommunication networks. In *Proceedings of International Symposium on Graph Drawing*, pp. 206-217.

Ustalov, D., Panchenko, A., & Biemann, C. (2017). Watset: Automatic Induction of Synsets from a Graph of Synonyms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 1579–1590.

Vehlow, C., Beck, F., & Weiskopf, D. (2016). Visualising Group Structures in Graphs: A Survey. *Computer Graphics Forum*, *36*(6), 201-225.

Venturini, T., Jacomy, M., & Pereira, D. (2014). *Visual Network Analysis*. Sciences Pomedialab.

Virtuoso SPARQL Query Editor. (n.d.). [online] Available at: https://dbpedia.org/sparql. [Assessed on 29 January 2020]

Walshaw, C. (2006). A Multilevel Algorithm for Force-Directed Graph Drawing. *Journal of Graph Algorithms and Applications*, *7*(3), 253–285.

Wan, T. W. F., Ranaivo-Malançon, B., & Chua, S. (2017). Minimising human labelling effort for annotating named entities in historical newspaper. *Journal of Telecommunication, Electronic and Computer Engineering*, *9*(2-10), 41-46.

Ward, M. O., Grinstein, G., & Keim, D. (2015). *Interactive Data Visualisation: Foundations, Techniques and Applications*, CRC Press.

Warfield, J. N. (1977). Crossing Theory and Hierarchy Mapping. *IEEE Transactions on Systems, Man, and Cybernetics*, *7*(7), 505-523.

Xu, K., Rooney, C., Passmore, P., Ham, D.-H., & Nguyen, P. H. (2012). A User Study on Curved Edges in Graph Visualisation. *IEEE Transactions on Visualisation and Computer Graphics*, *18*(12), 2449-2456.

Zaidi, F., Archambault, D., & Melançon, G. (2010). Evaluating the Quality of Clustering Algorithms using Cluster Path Lengths. In *Proceedings of Industrial Conference on Data Mining*, pp. 42-56.

Zhao, Y., & Karypis, G. (2011). Document clustering. In *Encyclopedia of Machine Learning* (Sammut, C., & Webb, G. I., Eds.), pp. 293-298. Springer Science & Business Media.

APPENDICES

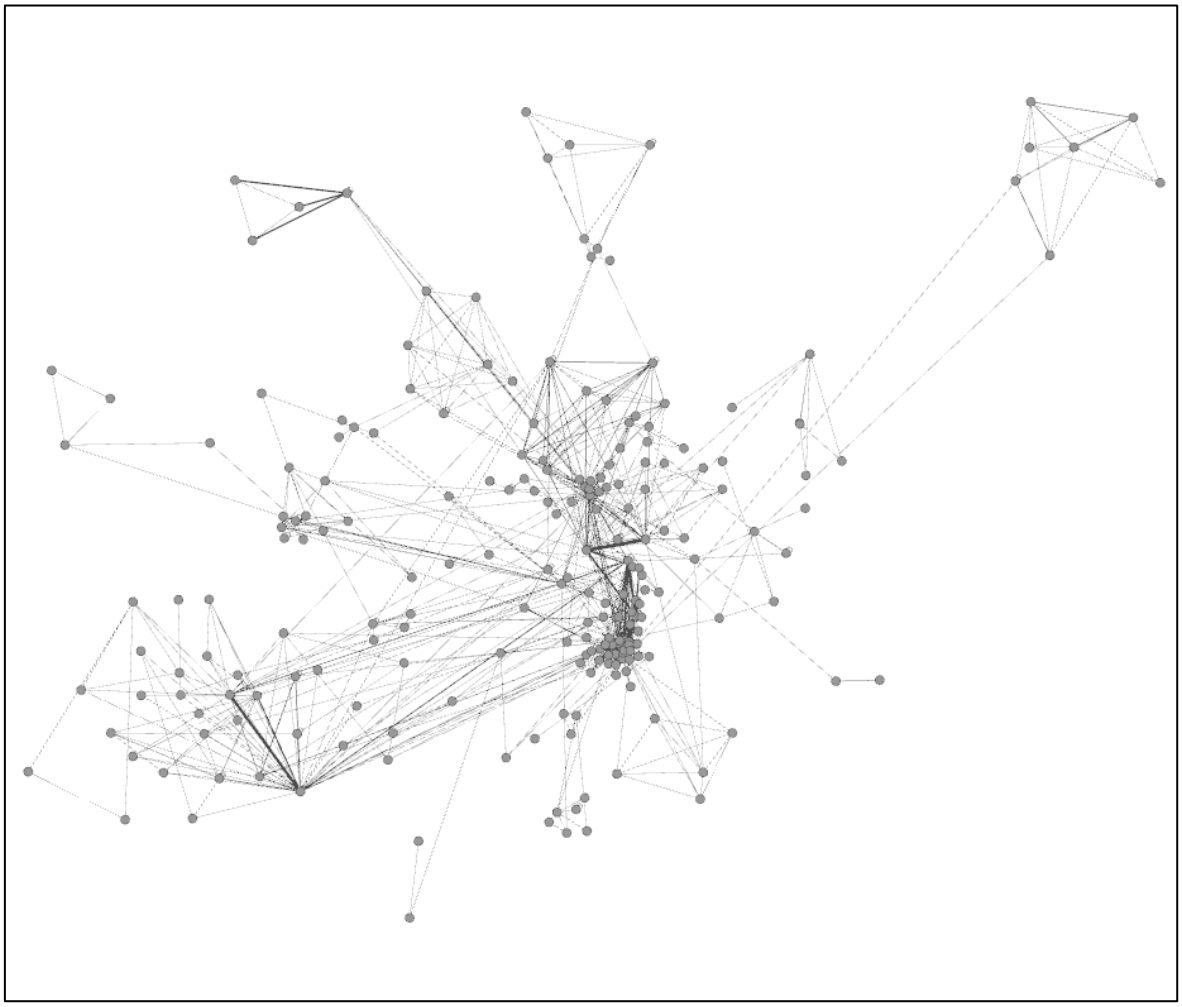
Appendix A: XSLT

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:output method="xml"/>

    <!-- Create a variable named newline which is actually to write on next line -->
    <xsl:variable name="newline">
        <xsl:text>
        </xsl:text>
    </xsl:variable>

    <xsl:template match="/"> <!-- / means the root of this file (select GateDocument as the root of this file) -->
        <xsl:apply-templates select="GateDocument"/>
    </xsl:template>

    <xsl:template match="GateDocument">
        <xsl:element name="AnnotationStartsHere">
            <xsl:apply-templates select="AnnotationSet"/>
        </xsl:element>
    </xsl:template>

    <xsl:template match="AnnotationSet">
        <xsl:value-of select="$newline"/>
        <xsl:for-each select=".//Annotation[@Type='Title'] | .//Annotation[@Type='Organization'] | .//Annotation[@Type='River'] |
        .//Annotation[@Type='Person'] | .//Annotation[@Type='Date'] | .//Annotation[@Type='Location'] |
        .//Annotation[@Type='JobTitle'] | .//Annotation[@Type='FirstPerson'] | .//Annotation[@Type='Ethny']">
            <xsl:element name="Annotation">
                <xsl:attribute name="StartNode">
                    <xsl:value-of select="@StartNode"/>
                </xsl:attribute>
                <xsl:attribute name="EndNode">
                    <xsl:value-of select="@EndNode"/>
                </xsl:attribute>
                <xsl:attribute name="Class">
                    <xsl:value-of select="@Type"/>
                </xsl:attribute>

                 <xsl:variable name="start" select="@StartNode"/>
                 <xsl:variable name="end" select="@EndNode"/>

                 <xsl:for-each select="//Annotation[@Type='Token' and @StartNode>=$start and @EndNode&lt;=$end]">
                    <xsl:attribute name="Id">
                        <xsl:value-of select="@Id"/>
                    </xsl:attribute>
                    <xsl:attribute name="Category">
                        <xsl:value-of select="Feature[Name='category']/Value"/>
                    </xsl:attribute>
                    <xsl:attribute name="Orth">
                        <xsl:value-of select="Feature[Name='orth']/Value"/>
                    </xsl:attribute>
                    <xsl:attribute name="Kind">
                        <xsl:value-of select="Feature[Name='kind']/Value"/>
                    </xsl:attribute>
                    <xsl:attribute name="Length">
                        <xsl:value-of select="Feature[Name='length']/Value"/>
                    </xsl:attribute>
                    <xsl:attribute name="String">
                            <xsl:value-of select="Feature[Name='string']/Value"/>
                    </xsl:attribute>
                 </xsl:for-each>

                 <xsl:for-each select="//Annotation[@Type='Sentence' and @EndNode>=$start and @StartNode&lt;=$end]">
                    <xsl:attribute name="SentenceNumber">
                        <xsl:value-of select="@Id"/>
                    </xsl:attribute>
                 </xsl:for-each>

            </xsl:element>
            <xsl:value-of select="$newline"/>
        </xsl:for-each>
    </xsl:template>

</xsl:stylesheet>
```

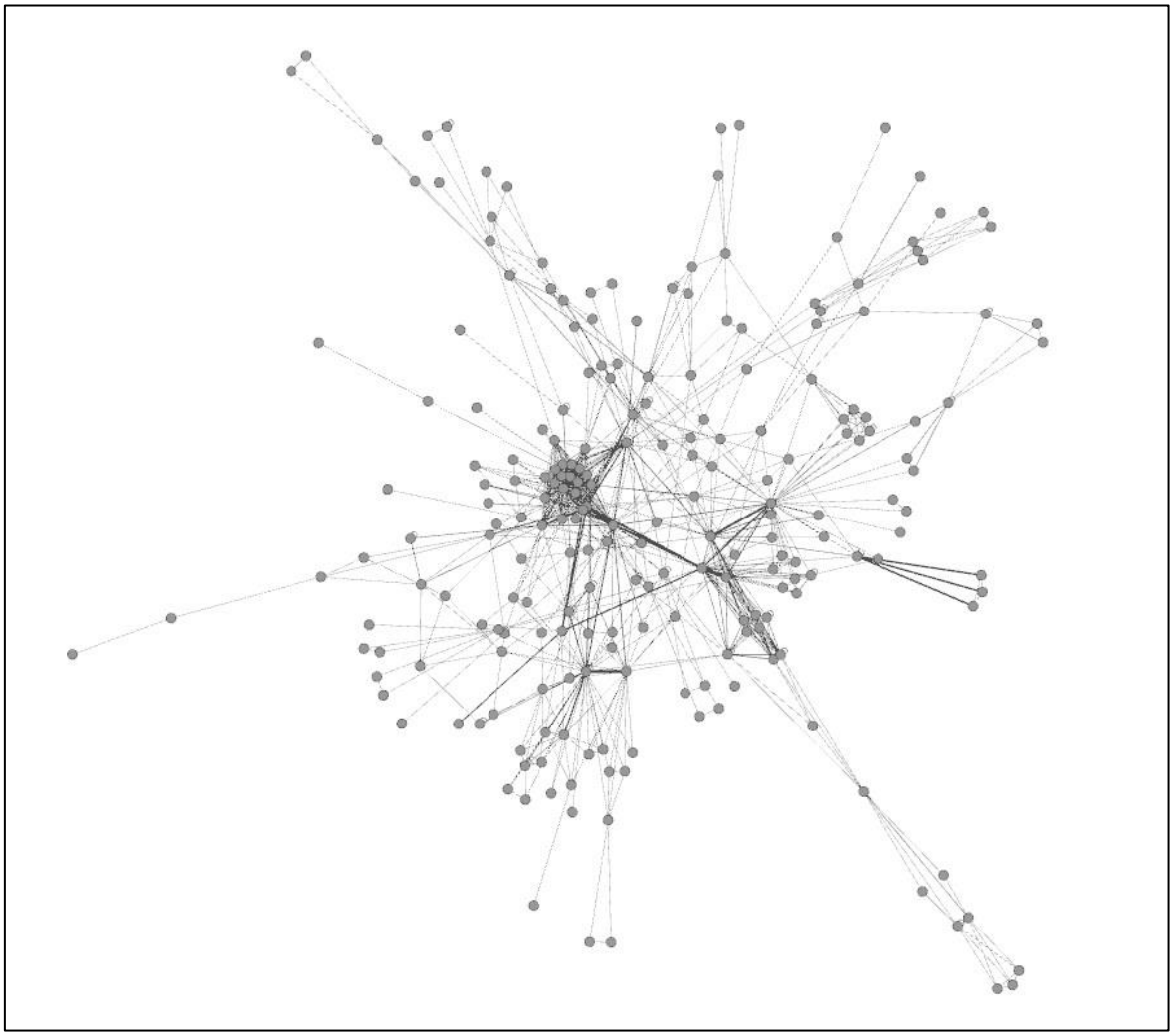
Appendix B: Output networks of graph layout algorithms on SAGA dataset

SAGA: FA2

SAGA: OO

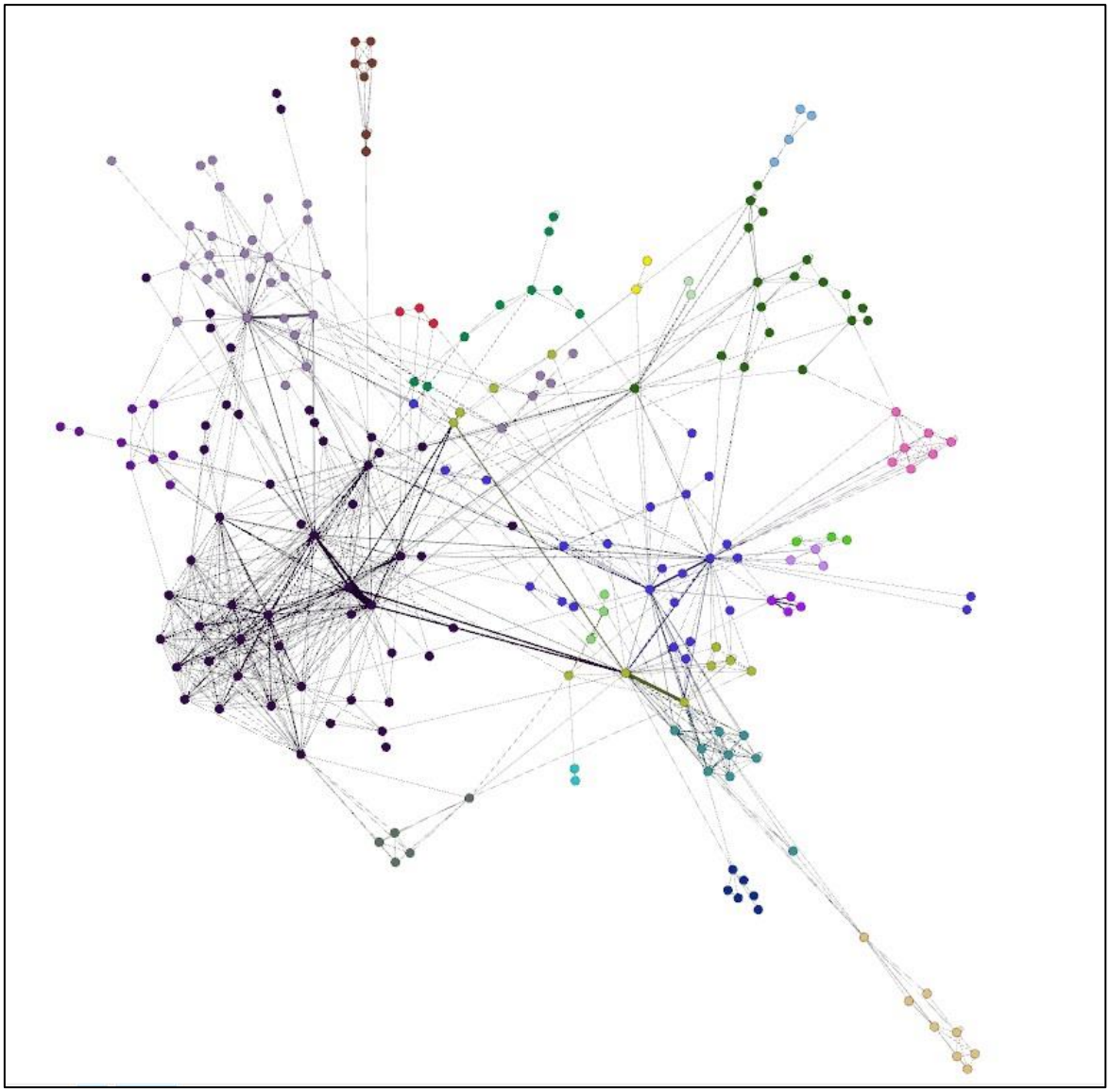


SAGA: HU

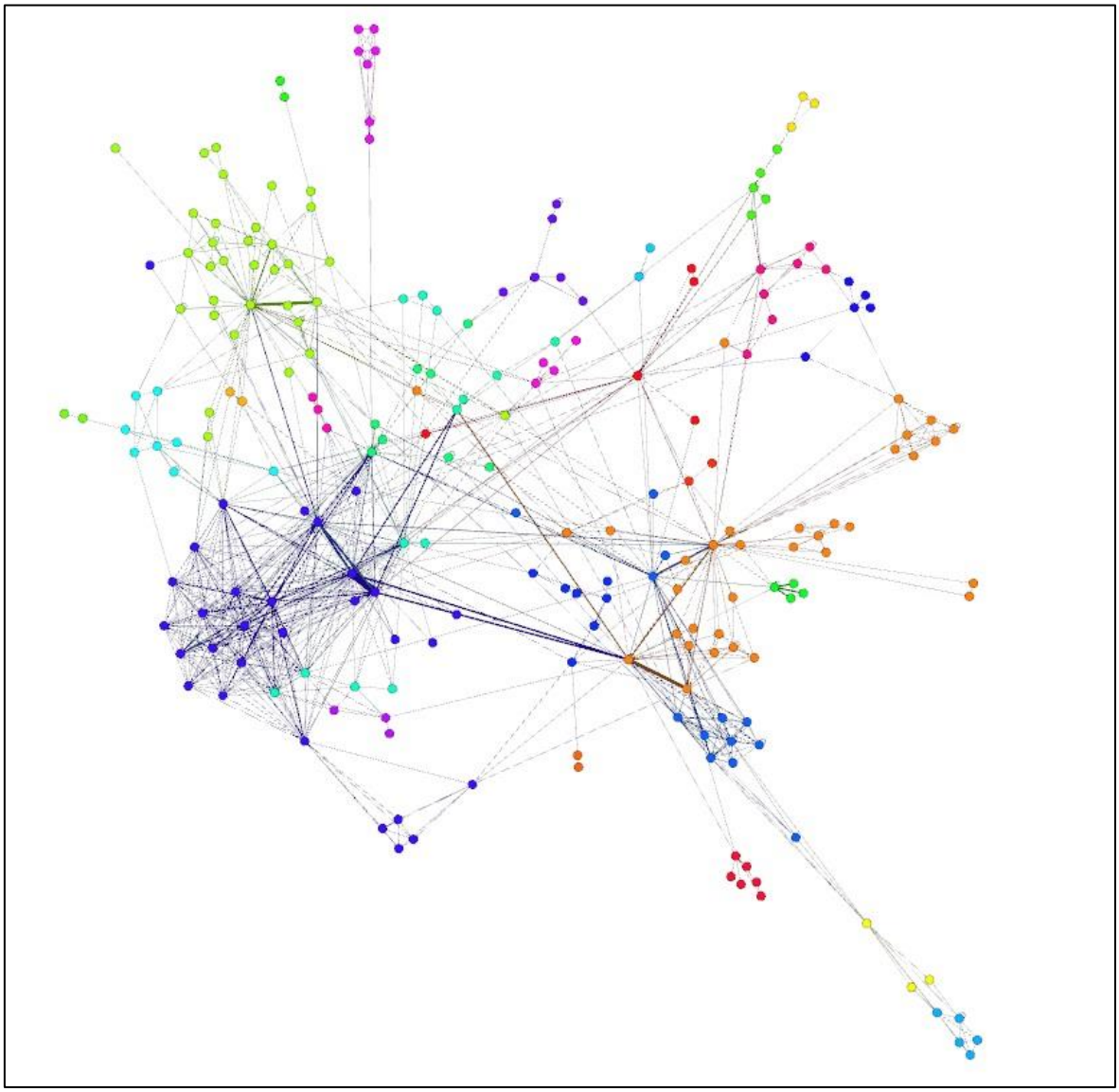


Appendix C: Output networks of graph clustering algorithm on SAGA dataset

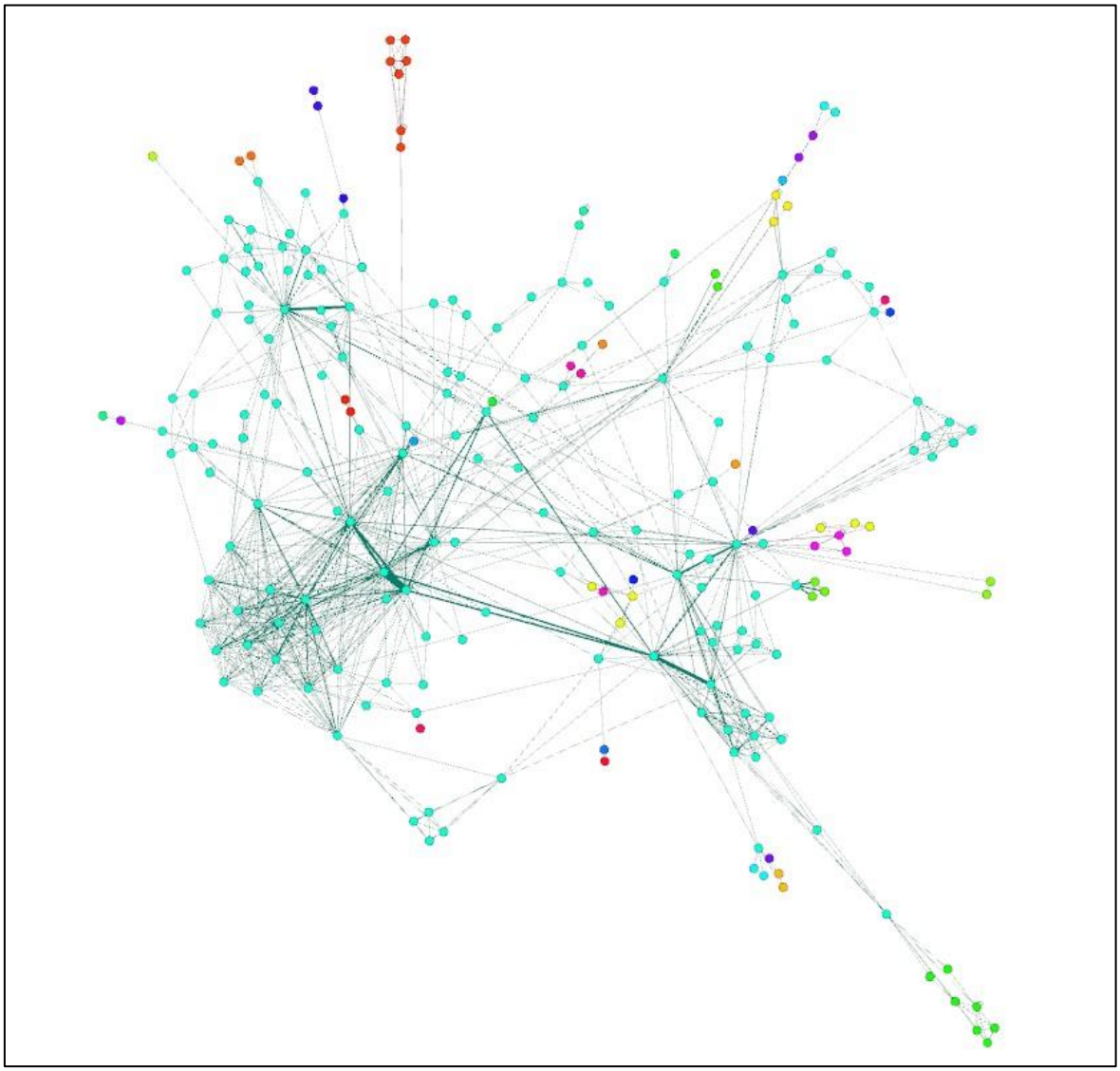
SAGA: FA2+CW



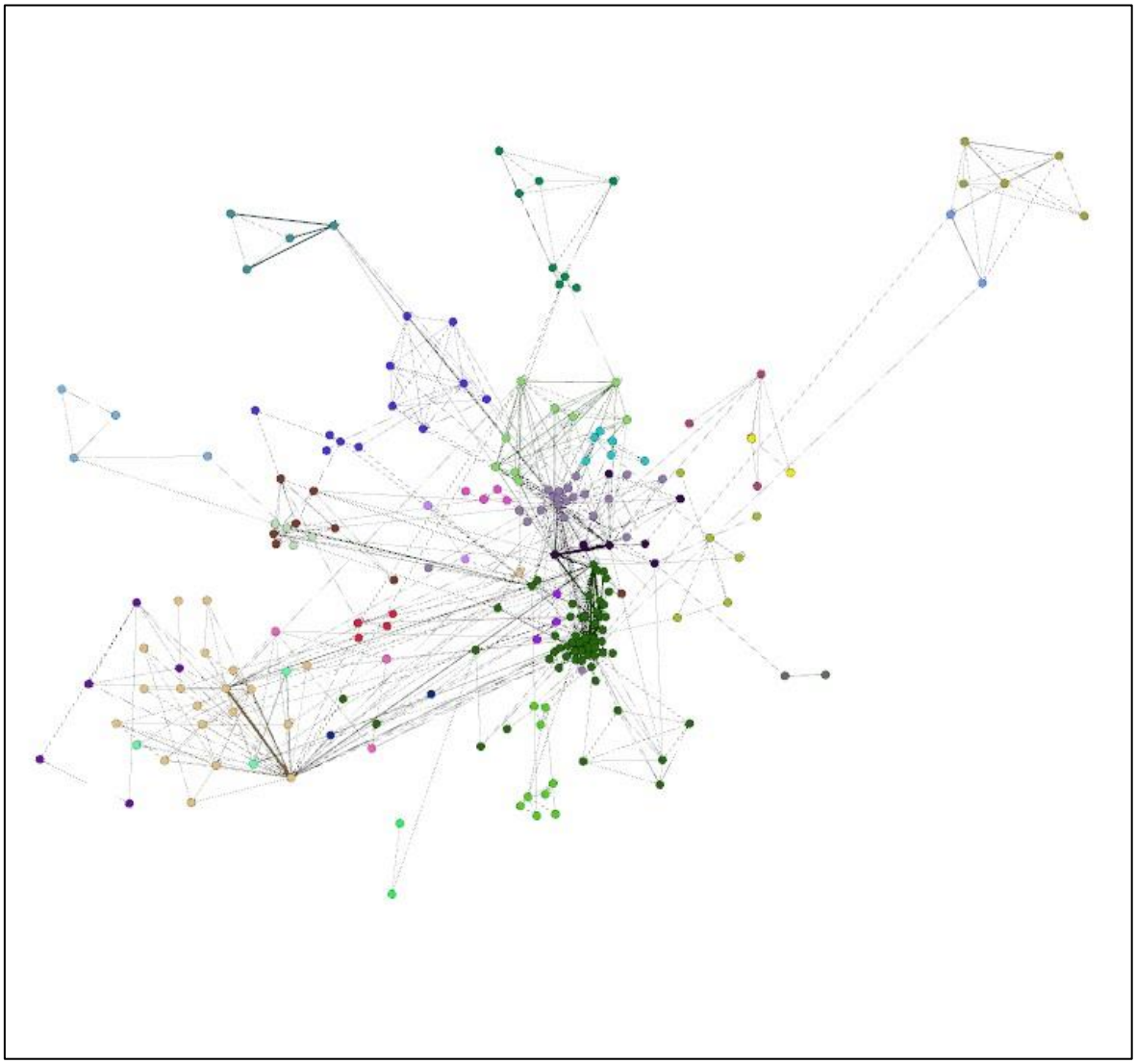
SAGA: FA2+MC



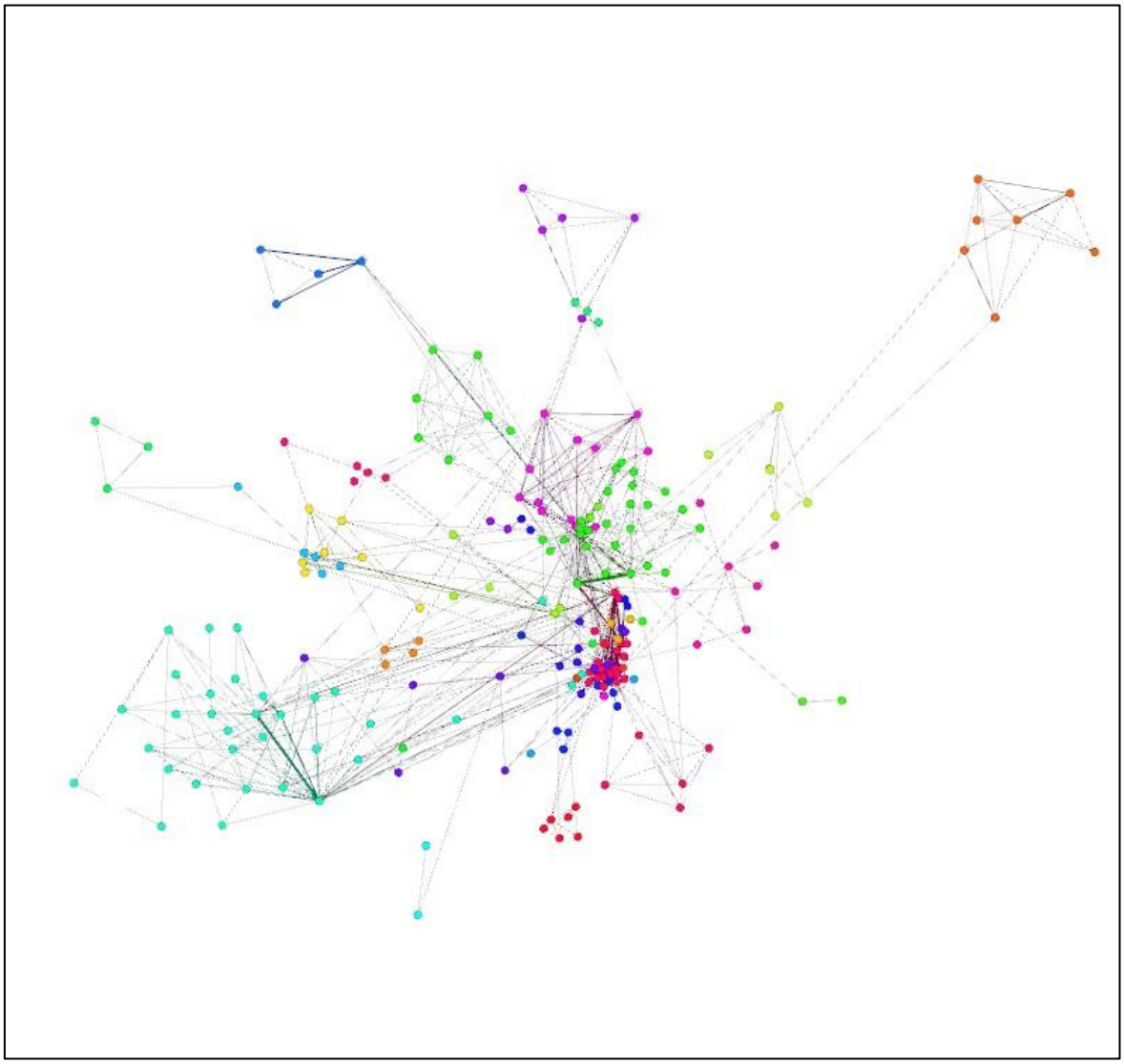
SAGA: FA2+GN



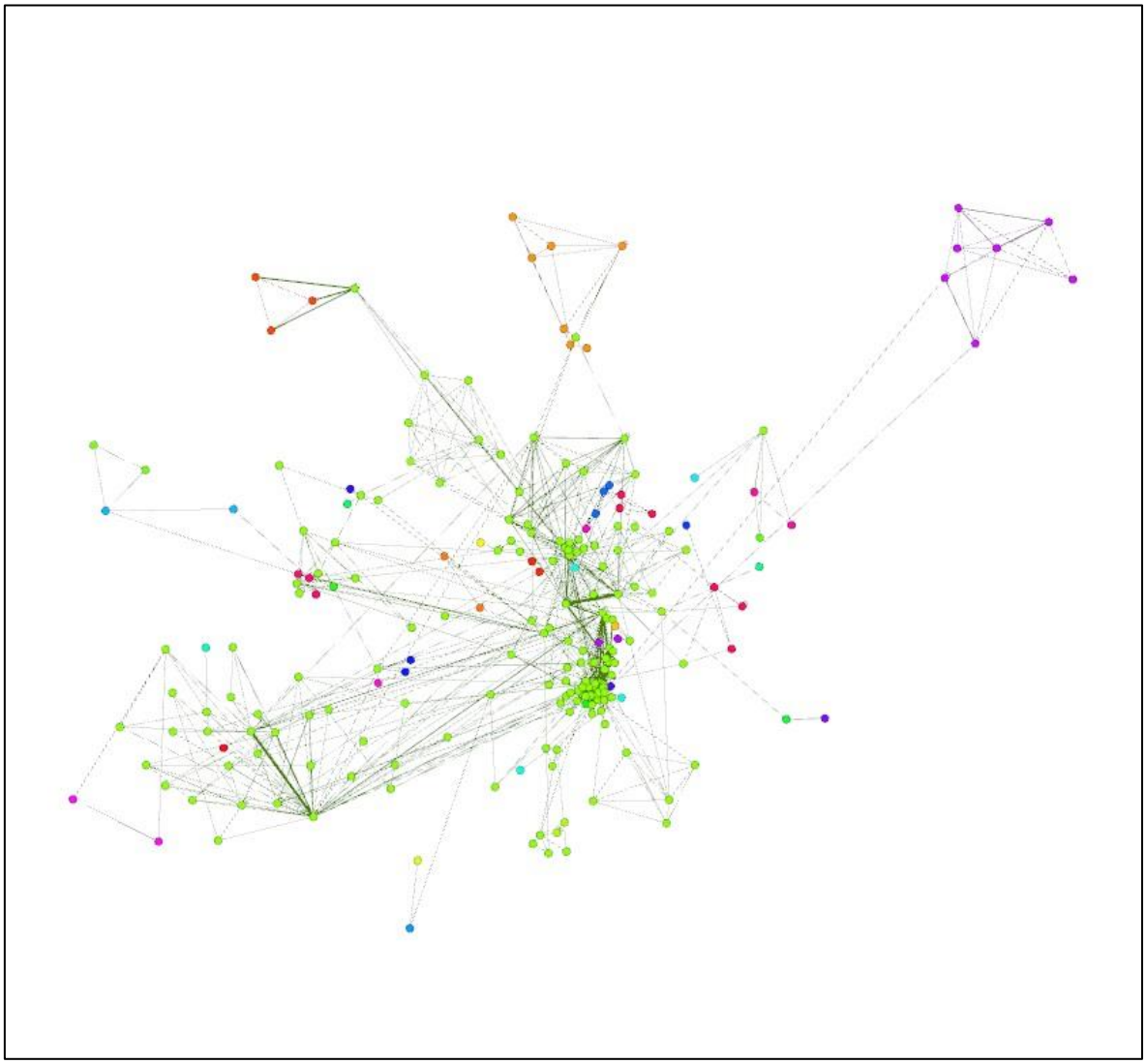
SAGA: OO+CW



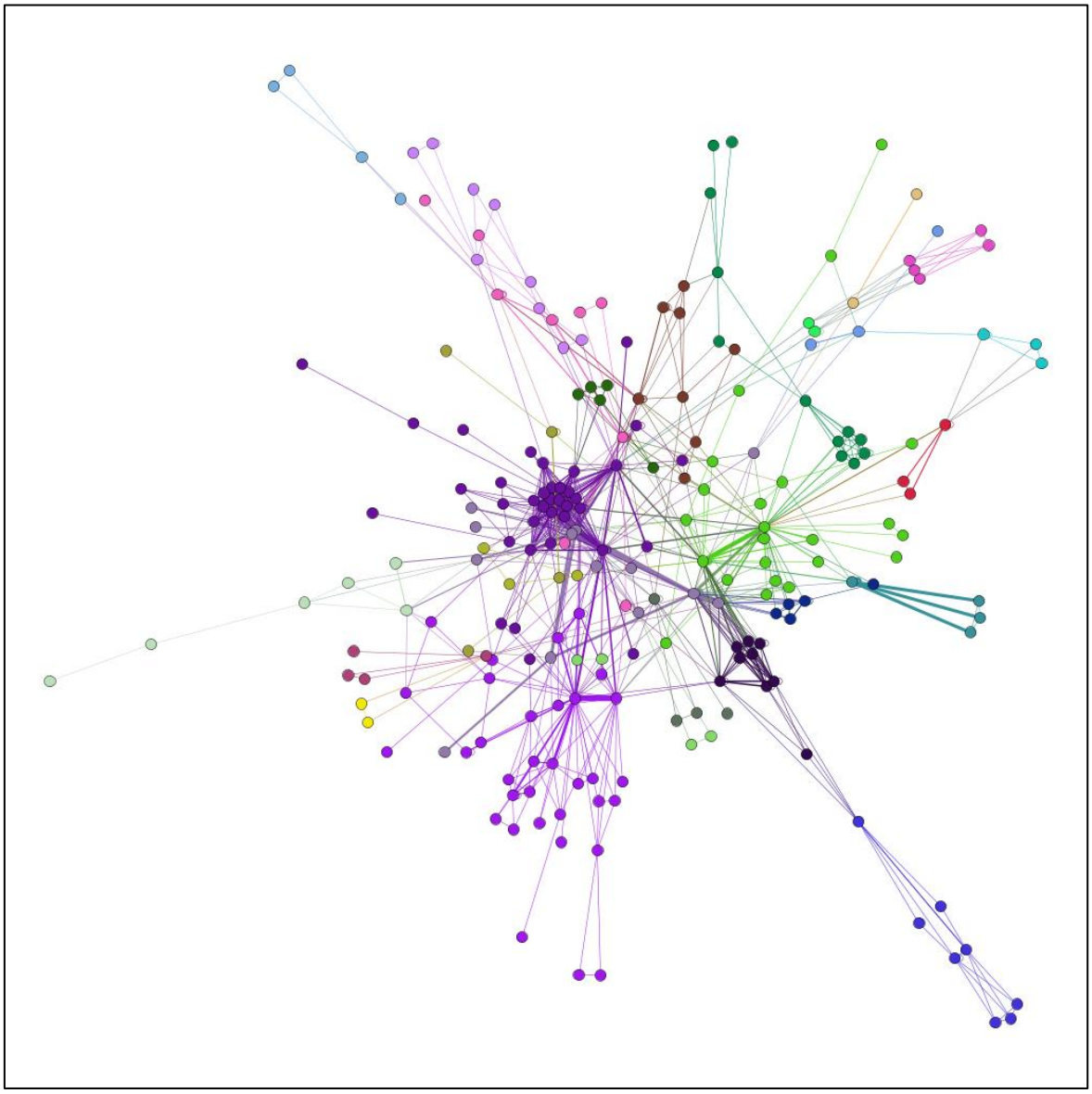
SAGA: OO+MC



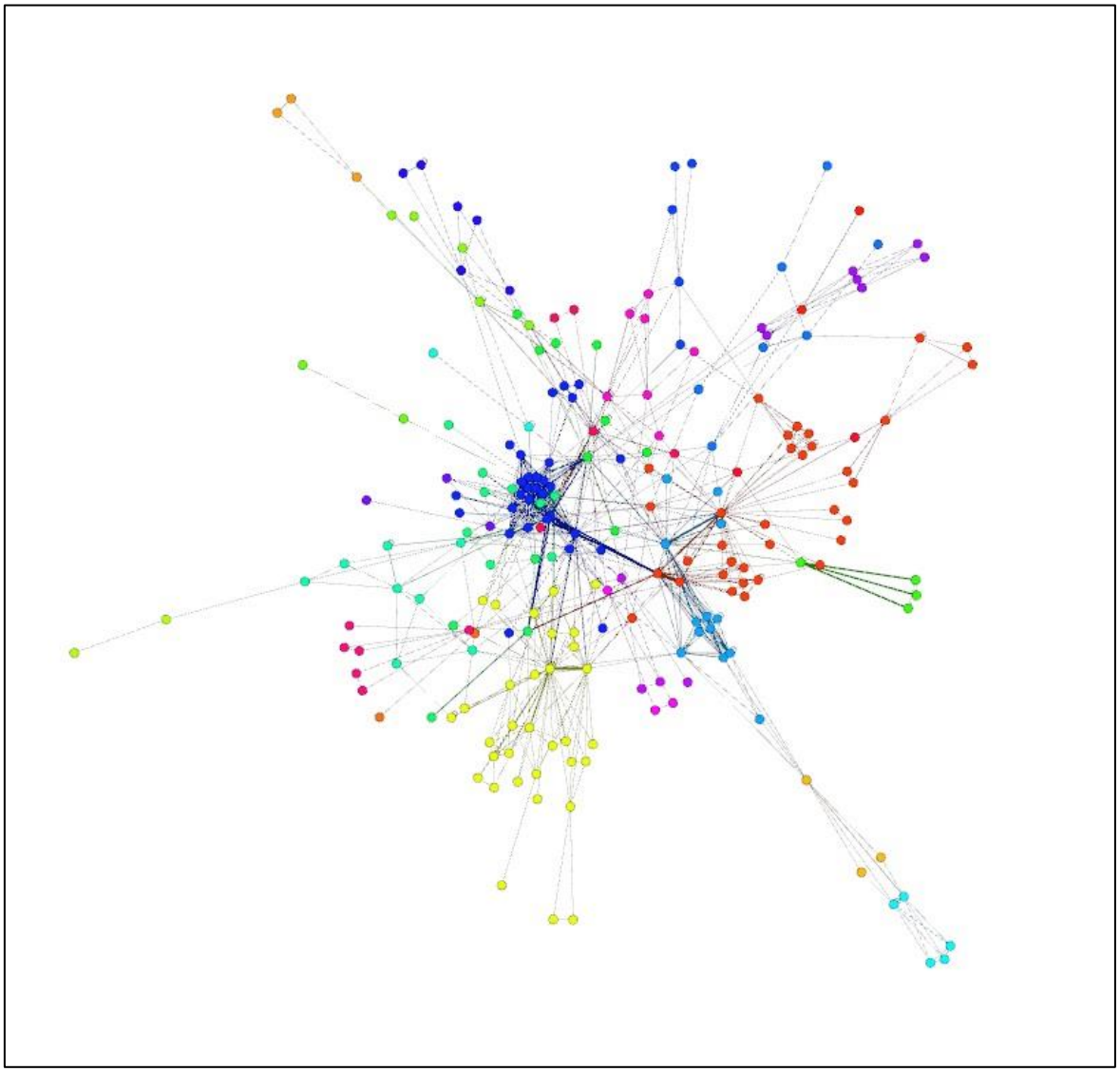
SAGA: OO+GN



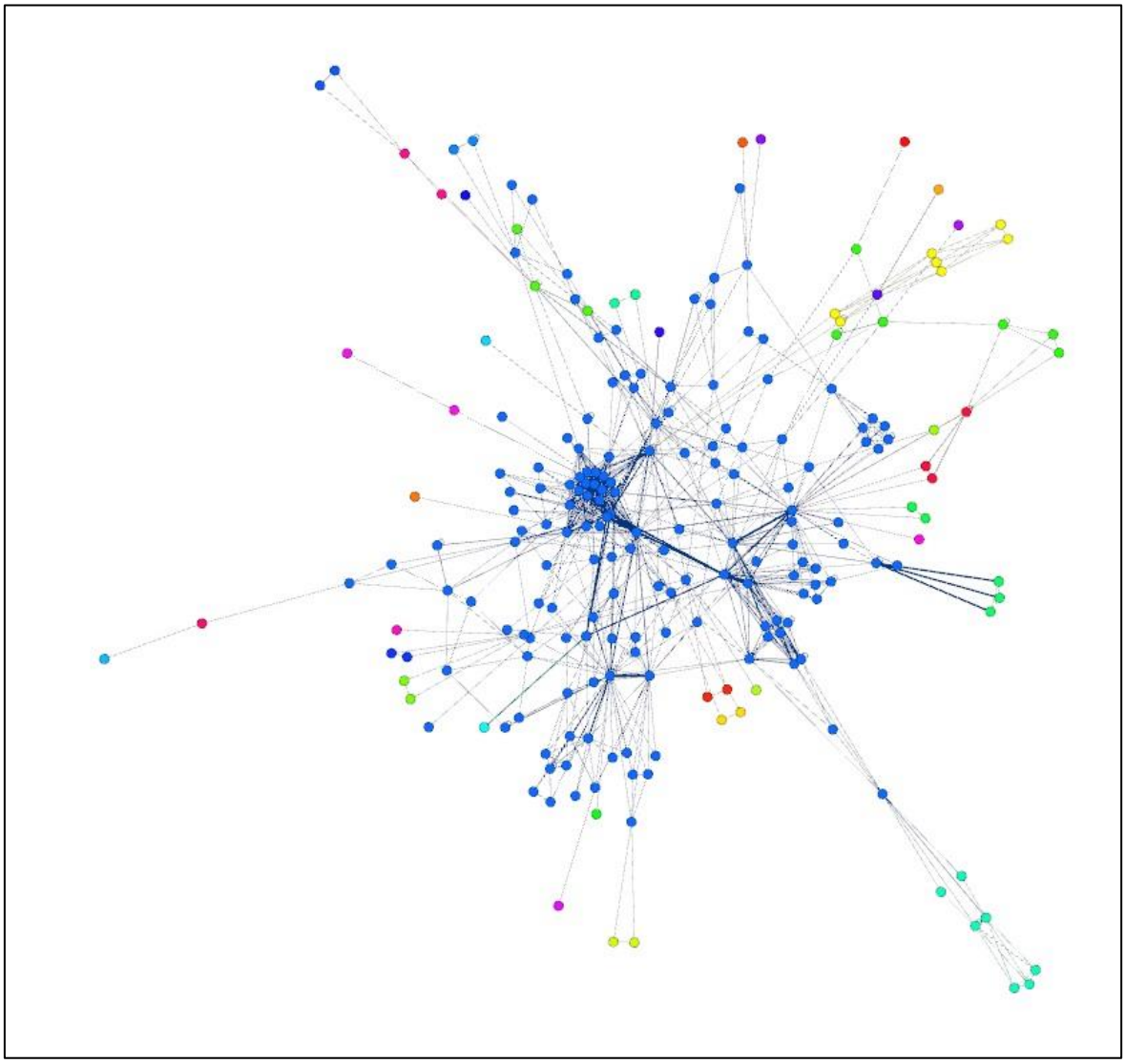
SAGA: HU+CW



SAGA: HU+MC

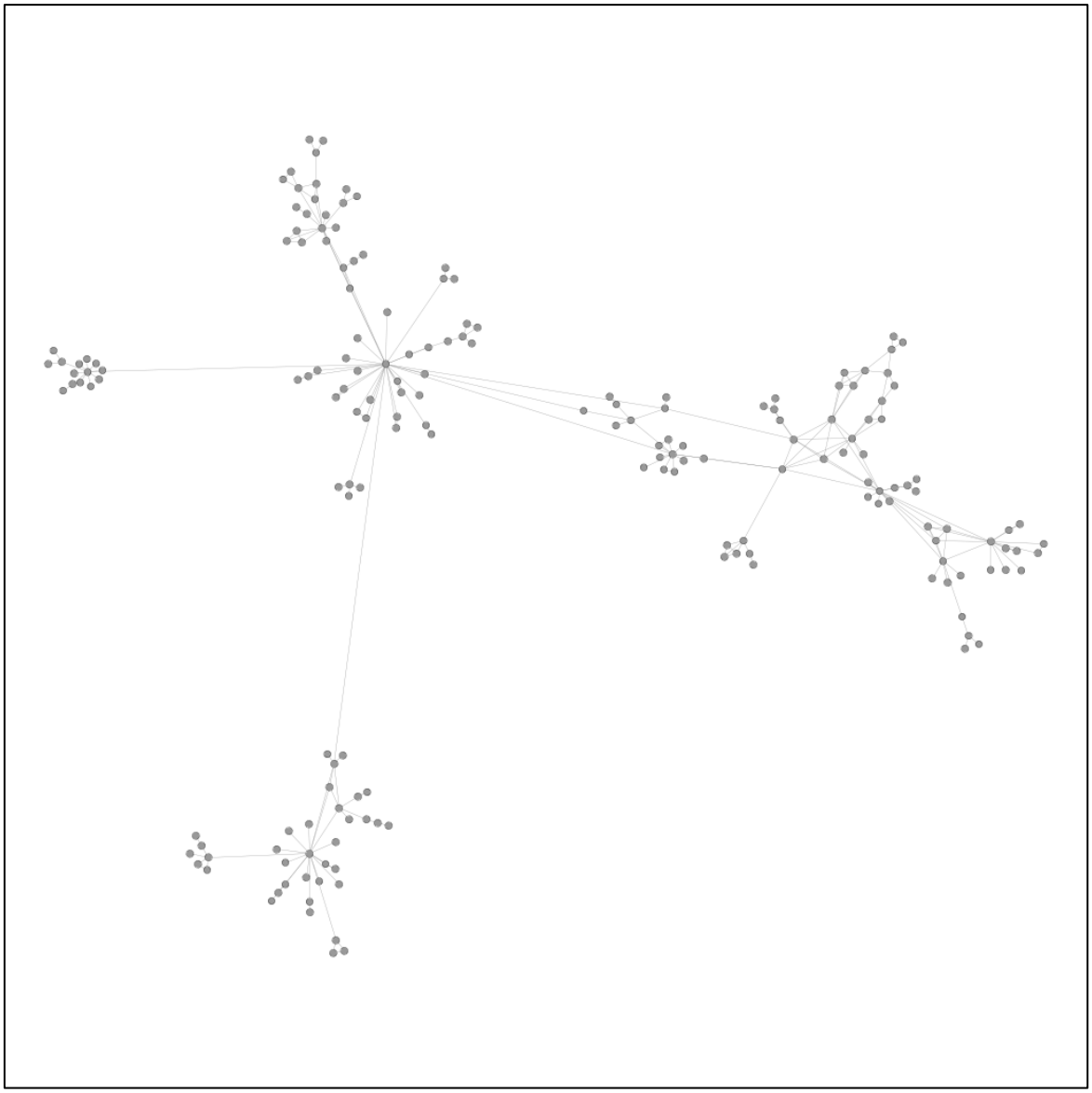


SAGA: HU+GN

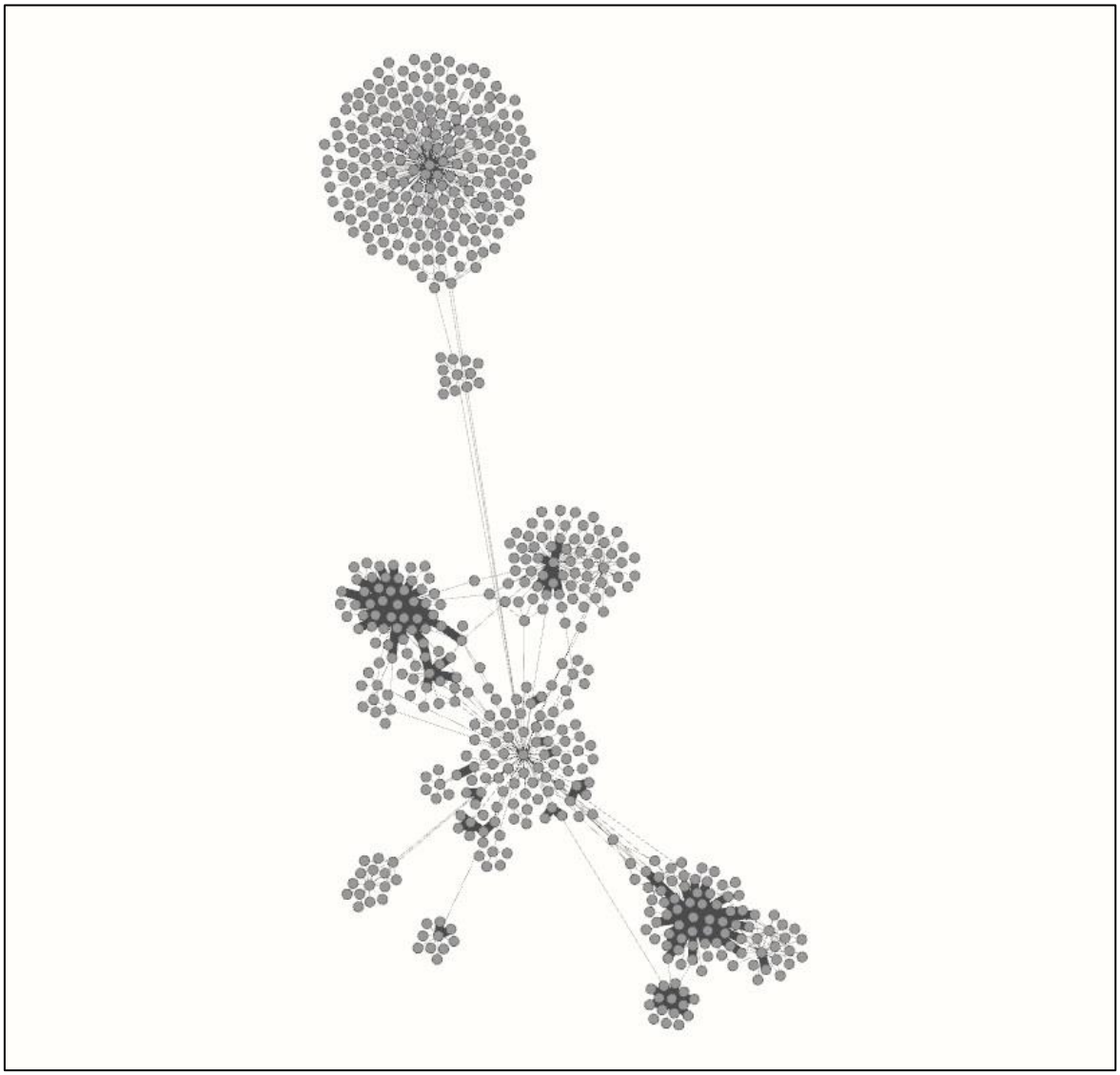


Appendix D: Output networks of graph layout algorithms on Biotext and dBPedia dataset

Biotext: FA2

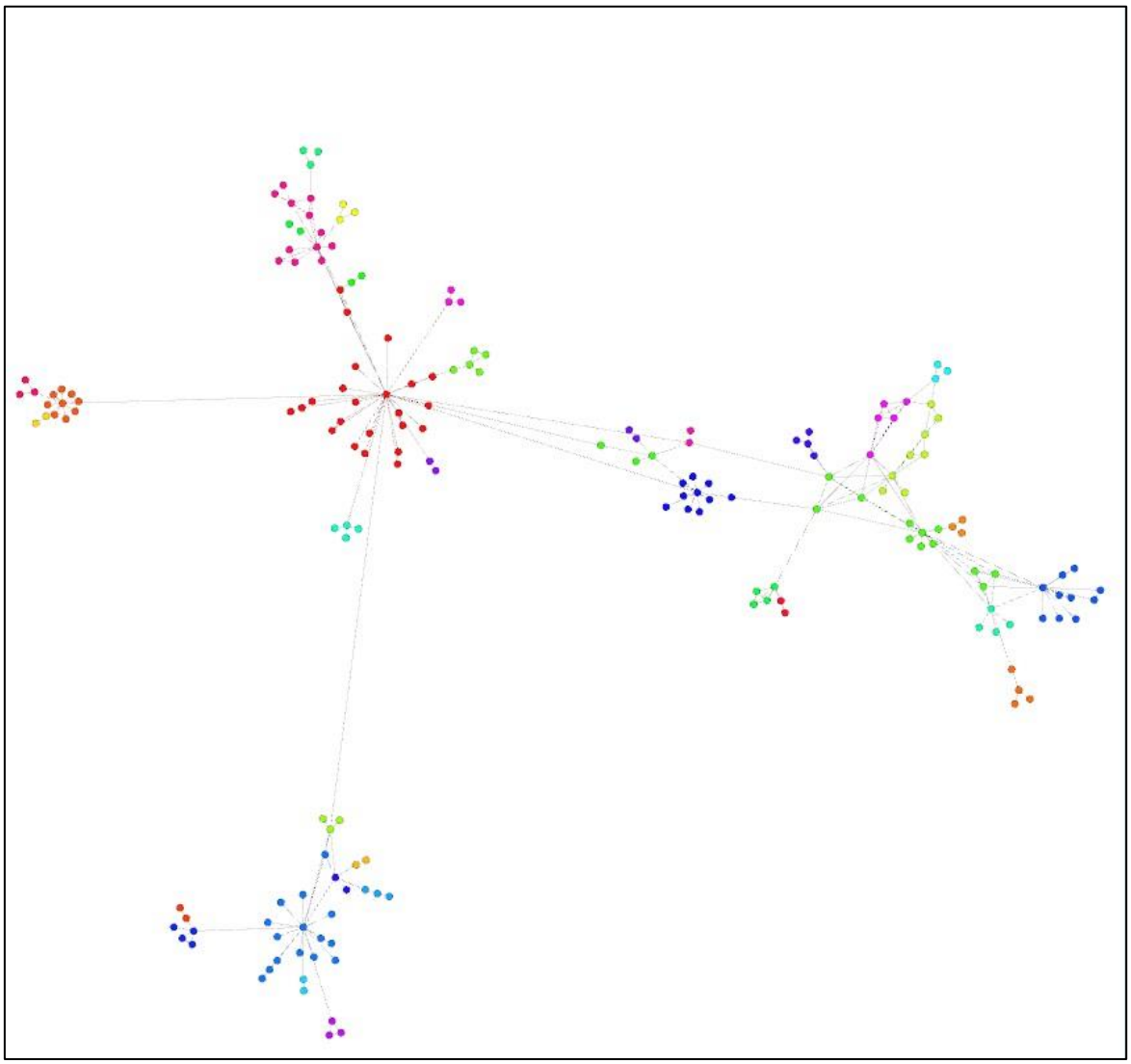


dBPedia: FA2

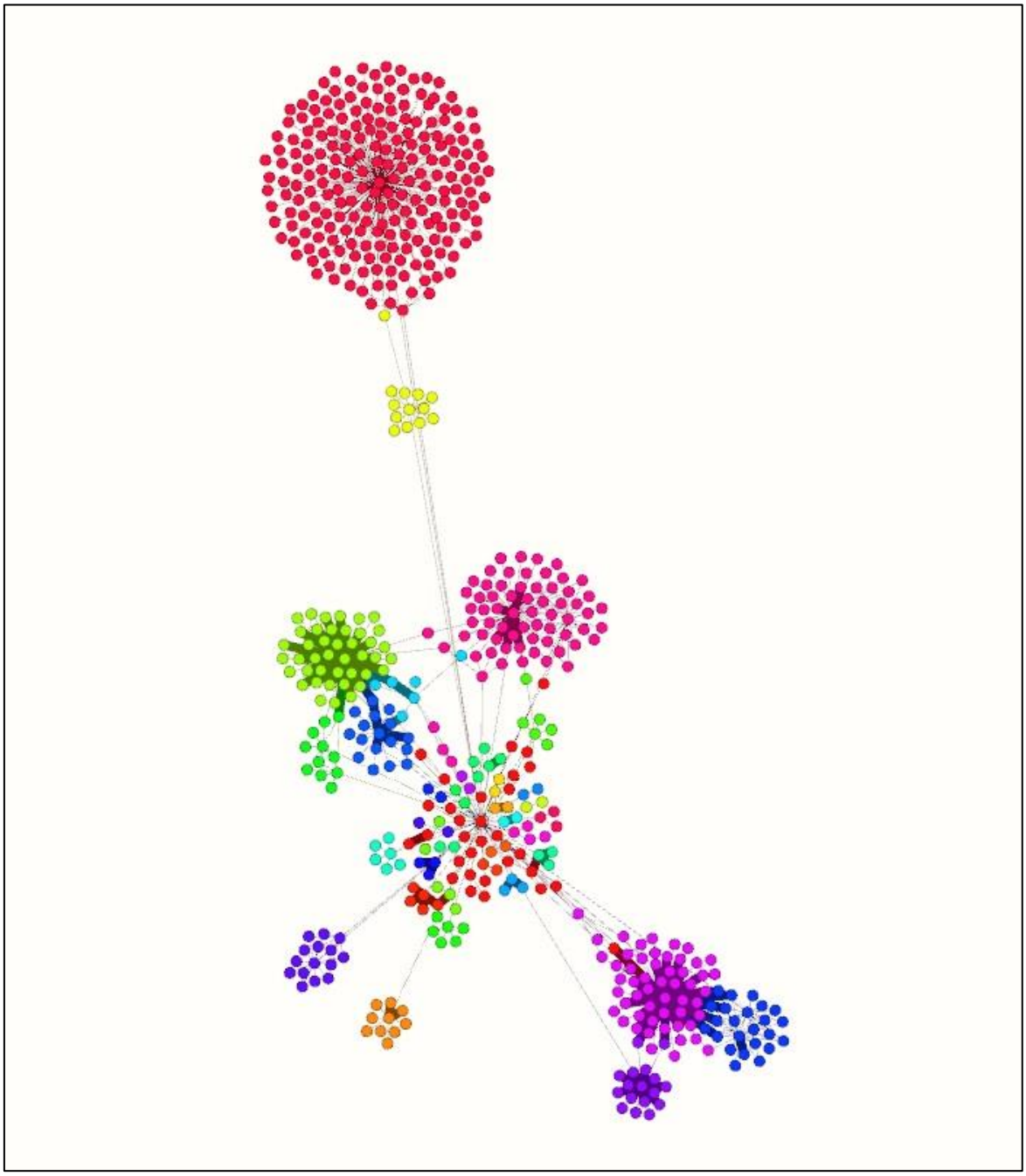


Appendix E: Output networks of graph clustering algorithms on Biotext and dBPedia dataset

Biotext: FA2+MC



dBPedia: FA2+MC



Appendix F: Further Evaluation Results on Biotext and dBPedia Datasets

Metric Scores of Graph Layout Algorithms

Data	Layout Algorithm	Running Time (min)	No of Edge Crossings	No of Cluttered Vertices
FA2	SAGA	Unlimited; stable position at 01:04.98	655	0
	Biotext	Unlimited; stable position at 00:52.71	21	0
	dBPedia	Unlimited; stable position at 01:21.91	230	0
HU	SAGA	00:13:13	732	353
	Biotext	00:10:95	31	0
	dBPedia	00:08:74	215	128
OO	SAGA	00:06:87	771	520
	Biotext	00:05:96	110	184
	dBPedia	00:03:09	648	741

Metric Scores of Graph Clustering Algorithms

Dataset	Layout	Clustering Algorithm	No of Clusters	Cluster Evaluation Metric			
				Modularity	CPL	Relative Density	Conductance
SAGA	MC	FA2	29	0.5366	0.7622	0.7418	0.5571
		HU	29	0.5366	0.7622	0.7418	0.5571
		OO	29	0.5366	0.7622	0.7418	0.5571
	CW	FA2	24	0.5535	0.7506	0.7857	0.6873
		HU	26	0.5476	0.7504	0.7921	0.6658
		OO	23	0.5639	0.7487	0.7697	0.6888
	GN	FA2	37	0.1466	0.4321	0.4794	0.2296
		HU	37	0.1466	0.4321	0.4794	0.2296
		OO	37	0.1466	0.4321	0.4794	0.2296
Biotext	MC	FA2	37	0.6898	0.7898	0.6871	0.7185
		HU	37	0.6898	0.7898	0.6871	0.7185
		OO	37	0.6898	0.7898	0.6871	0.7185
	CW	FA2	26	0.7210	0.6946	0.5699	0.7519
		HU	25	0.7198	0.6971	0.5708	0.7509
		OO	24	0.7232	0.6821	0.5528	0.7611
	GN	FA2	30	0.6514	0.7089	0.6552	0.6417
		HU	30	0.6514	0.7089	0.6552	0.6417
		OO	30	0.6514	0.7089	0.6552	0.6417
dBPedia	MC	FA2	39	0.7433	0.7480	0.5798	0.7590
		HU	39	0.7433	0.7480	0.5798	0.7590
		OO	39	0.7433	0.7480	0.5798	0.7590
	CW	FA2	34	0.7117	0.6860	0.4903	0.7596
		HU	31	0.7175	0.6790	0.4767	0.7649
		OO	35	0.7145	0.6967	0.5053	0.7557
	GN	FA2	26	0.5222	0.8028	0.7051	0.7287
		HU	26	0.5222	0.8028	0.7051	0.7287
		OO	26	0.5222	0.8028	0.7051	0.7287