Department of Computer Science
Series of Publications A
Report A-2012-6

Methods for Network Abstraction

Fang Zhou

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium XII, University Main Building, on August 16th, 2012, at 12 o'clock noon.

University of Helsinki
Finland

Supervisor
Hannu Toivonen, University of Helsinki, Finland

Pre-examiners
Nada Lavrač, Jožef Stefan Institute, Slovenia
Jiuyong Li, University of South Australia, Australia

Opponent
Christian Borgelt, European Centre for Soft Computing, Spain

Custos
Hannu Toivonen, University of Helsinki, Finland

Contact information

Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FI-00014 University of Helsinki
Finland

Email address: postmaster@cs.helsinki.fi
URL: http://www.cs.Helsinki.fi/
Telephone: +358 9 1911, telefax: +358 9 191 51120

ISSN 1238-8645
ISBN 978-952-10-8157-6 (paperback)
ISBN 978-952-10-8158-3 (PDF)
Computing Reviews (1998) Classification: G.2.2, G.2.3, H.2.8
Helsinki 2012
Unigrafia

Methods for Network Abstraction

Fang Zhou

Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
fang.zhou@cs.helsinki.fi
http://www.cs.helsinki.fi/u/fzhou/

PhD Thesis, Series of Publications A, Report A-2012-6
Helsinki, August 2012, 48 + 71 pages
ISSN 1238-8645
ISBN 978-952-10-8157-6 (paperback)
ISBN 978-952-10-8158-3 (PDF)

Abstract

We propose *network abstraction* as a research area. It is motivated by the growth of networks in many areas of life. Consider, for instance, networks of thousands of genes, millions of people, or billions of web pages. They are too large to be directly analyzed by users. The aim of network abstraction is to summarize a large network as a smaller one. An abstracted network can then help users to see the overall topology of a large network, or to understand the connections of distant nodes.

The general network abstraction task is: given a large network, transform it into a smaller one, which contains in some well-specified sense the most relevant information. In this thesis, we analyze this research area and propose methods to solve some instances of the problem. The methods also provide different trade-offs between the graph quality and simplicity, as well as between result quality and efficiency.

More specifically, we propose two approaches to abstracting a network. The first one is to simplify a weighted network by removing edges under the constraint that distances between all pairs of nodes are preserved. We first empirically show that a number of edges can be removed from real biological networks without losing any graph connectivity. We next relax the constraint of fully preserving original graph connectivity, extend lossless network simplification to lossy network simplification, and demonstrate that many more edges can be removed with little loss of quality.

The second approach we give for network abstraction is to compress a weighted network by grouping nodes and edges. We propose novel methods and experimentally show that real graphs can be compressed efficiently with relatively little error. We next consider graphs with weights also on the nodes, and utilize them as node importances to extend the definition of weighted graph compression. We present new compression operations and demonstrate that the compressed graph can preserve more information related to more important nodes.

Furthermore, we propose the idea of using node weights and compression to summarize the metabolisms in a set of organisms, and apply the methodology to better understand the metabolic biodiversity between *Archaea* and *Eubacteria*, the two most fundamental branches of life.

Computing Reviews (1998) Categories and Subject Descriptors:
G.2.2 Graph Theory
G.2.3 Applications
H.2.8 Data mining

General Terms:
Algorithms, Experimentation

Additional Key Words and Phrases:
Weighted Network, Abstraction, Graph Mining, Bioinformatics

Acknowledgements

I am most grateful to my supervisor, Prof. Hannu Toivonen, for teaching me every aspect of research and guiding me through the doctoral studies. His insightful questions led me to explore the paths of science. He always helped and supported me in every important step.

Thanks to Prof. Ross D. King for his advice and encouragement. I am grateful to him for the discussions that helped me to sort out the biological work. He has done an excellent job in mentally supporting me.

I appreciate my mentor Dr. Huizhen Yu and Dr. Sèbastien Mahler for their guidance during the first two years of my Ph.D studies. Thanks to Prof. Ming Li and Asst. Prof. Jonathan H. Badger for their help in the biological work.

I would like to thank my pre-examiners Prof. Jiuyong Li and Prof. Nada Lavrač for their useful comments that improved the quality of the thesis, and Marina Kurtén, MA, for her help in improving the language of the thesis.

Thanks to the previous and current members in the Discovery group for their help during this work. Particularly thanks to Laura Langohr, Aleksi Hartikainen, Dr. Petteri Hintsanen, Lauri Eronen, Esther Galbrun, Atte Hinkka and Kimmo Kulovesi.

I would like to thank Hongyu Su, Liang Wang, Jie Xiong, Chengyu Liu, Yi Ding, Lu Cheng and my other friends, I have enjoyed our discussions on random topics.

This work has been carried out at the Department of Computer Science at the University of Helsinki. I am thankful to the administration staff and IT support staff at our department for creating an efficient and pleasant working environment.

I appreciate the financial support from Helsinki Graduate School in Computer Science and Engineering (Hecse), Chinese scholarship, and the Bison project (European Commission under the Seventh Framework Programme FP7-ICT-2007-C FET-OPEN).

Lastly, but most importantly, I would like to thank Shaohua Fan and my parents, Huiying Zhou and Weihua Zhou. Thanks for the endless love and for always supporting me to do what I love. To them, I dedicate this thesis.

Fang Zhou @ Helsinki

June, 2012

Contents

List of Publications and The author's Contributions

This thesis consists of this introductory part and the following five original publications, reprinted at the end of the thesis.

Article I Review of Network Abstraction Techniques, Fang Zhou, Sébastien Mahler, and Hannu Toivonen. In *Workshop on Explorative Analytics of Information Networks,* Bled, Slovenia, 2009. (Extended version)

The author participated in discussing and defining the area of network abstraction. She reviewed articles and took part in writing the paper.

Article II A Framework for Path-Oriented Network Simplification, Hannu Toivonen, Sébastien Mahler and Fang Zhou. In *Advances in Intelligent Data Analysis IX, 9th International Symposium, IDA 2010,* pages 220-231. Springer, 2010.

The author implemented and analyzed the algorithms, performed the experiments, and analyzed the experimental results.

Article III Network Simplification with Minimal Loss of Connectivity, Fang Zhou, Sébastien Mahler, and Hannu Toivonen. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM'10, pages 659-668. Washington, DC, USA, 2010. IEEE Computer Society.

The author took part in conceiving the research idea. She had main responsibility for designing and implementing the algorithms, analyzing the computational complexity, performing experiments, and analyzing the experimental results. She also wrote a majority of the paper.

Article IV Compression of Weighted Graphs, Hannu Toivonen, Fang Zhou, Aleksi Hartikainen, and Atte Hinkka. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* KDD'11, pages 965-973, New York, NY, USA, 2011, ACM.

The author had main responsibility for developing the research idea. She designed the algorithms, performed the experiments, analyzed the results, and took part in writing the paper.

Article V Abstracting Weighted Graphs: Generalization based on Node Importance, Fang Zhou, Hannu Toivonen, Aleksi Hartikainen, Jonathan H. Badger, and Ross D. King. (submitted)

The author participated in developing the research idea. She designed the algorithms and implemented parts of them, performed the experiments, analyzed the results, and co-wrote the paper.

Chapter 1

Introduction

The topic of this thesis is *network abstraction*, and the goal is to propose computational tools to transform a large network into a smaller one which is more useful for visualization, analysis and understanding. This thesis consists of the present introductory part in six chapters and five original articles reprinted at the end of the thesis.

In this chapter, I introduce the topic, describe the motivations of the work, present research questions, and give an overview of the original articles.

1.1 Topic of the thesis

The goal of this research work is to propose methods to abstract a simpler and useful network from a large one, with respect to a possible application. The author of this thesis and other contributors review and analyze the filed, and propose two approaches to abstract a network. We next give an overview of the research area, and return to the contributions of the thesis later.

Networks are a common and powerful formalism for linked data: nodes represent objects and links (or edges) represent connections between objects. Prominent examples include World Wide Web, social networks (such as Facebook), biological networks (e.g. metabolic network), and communication networks. Networks play a crucial role in describing how objects interact with each other. The terms network and weighted graph are interchangeable in this thesis.

Numerous data mining techniques have been proposed to explore network properties and discover new knowledge. For example, frequent subgraph mining techniques find frequent patterns (or subgraphs) in a large graph (or a database of graphs); clustering techniques try to find communities in a large graph. Our work differs from such mainstream graph mining and analysis: we investigate methods to make networks simpler. Our methods may, however, have applications as a pre-processing step for some graph mining methods.

Networks are often large. Consider networks of thousands of genes, millions of people, billions of web pages. They are too large to be directly visualized by users. The big size increases the complexity of understanding. For instance, it is hard for a user to see the overall topology of the network. Furthermore, it increases the complexity of exploring the hidden knowledge. One solution is to transform a large network into a simpler and smaller one, which maintains the most relevant information related to one application, although some irrelevant information may be lost. We call this approach *network abstraction*.

As an example, consider a social network, such as Facebook. Nodes represent individuals and edges represent friendships. Suppose a user wants to know how two persons, A and B, know each other. A smaller graph, which is maximally relevant to the user-specified query nodes A and B, contains enough information to make the relations between A and B understandable. Other nodes, which are irrelevant to query nodes, do not necessarily have to be shown to the user, and their existence increases the complexity of understanding.

In more general, network abstraction is a process or result of generalization by reducing or factoring out details. Typically this is done in order to retain only information which is relevant to a particular purpose, so that one can focus on a few most important relations at one time. Several operations can be used to obtain an abstracted graph. The simplest operation is removing redundant or unimportant nodes and edges. A minimum spanning tree is an extreme example of this: a graph is abstracted into a tree. Another operation is dividing a graph into several components. Yet another operation is replacing some parts of a graph by general ones. We provide a taxonomy and review of network abstraction methods and operations in Article I.

Network abstraction techniques may bring several benefits. First, it simplifies the structure of a large network, so that a user can more easily study the topology of a large network and understand the connections between distant nodes. Second, it allows a user to concentrate on a small part of the graph. Third, it may help a user to discover hidden knowledge by eliminating the irrelevant information. Fourth, as a pre-processing for other data mining algorithms, it may reduce the computational complexity by extracting the most important information related to the application.

In this work, network abstraction techniques are investigated for weighted and undirected graphs. The tasks include exploring how to measure the quality of an abstracted network, how to balance loss of information over the gained simplicity, how the users can control these operations, and how to make methods feasible and efficient for large networks.

1.2 Research questions

In this thesis, we seek to answer the following questions. The first question helps to set the exact goal for a specific network abstraction problem.

I *How can we measure the quality of an abstracted network?*

Given that we are able to tell if an abstraction is good or not, we face the second research question:

II *What kind of operations and methods can we use to abstract a graph?*

The second question will lead us to explore feasible abstraction techniques, balancing between quality and efficiency. When we obtain an abstracted graph, we will face the third question:

III *How useful is an abstracted graph?*

This thesis will aim to shed light on these questions. Most of the answers will be measures and methods for specific network abstraction problems. The usefulness will be mainly evaluated in the context of a novel biological application.

1.3 Overview of articles

The scientific contributions of this thesis are in the original Articles I-V. Article I reviews the state-of-the-art of network abstraction techniques. Articles II-V address two technical subtopics: network simplification and network compression. Below I briefly describe the motivation of each work and summarize their main results. Discussions of the results are deferred to Chapter 5.

Article I Review of Network Abstraction Techniques

This paper surveys techniques that can abstract a large network into a smaller one (Research question II). We classify these techniques based on two orthogonal characteristics. One consists of three elementary operations: *prune* unimportant (or irrelevant) nodes and edges, *partition* a graph to several components, *replace* parts of graphs by more general ones. The other characteristic classifies techniques based on whether user focus is allowed or not: *objective vs. subjective*. Furthermore, we propose areas for further research based on the gaps observed in the review.

Article II A Framework for Path-Oriented Network Simplification

We propose a generic framework and methods for network simplification by removing redundant edges. It is based on the observation that a network can have a simpler structure but still maintain the original quality of the best paths between all pairs of nodes (Research question I). We propose four algorithms to remove redundant edges (Research question II). The framework is applicable to different kinds of graphs, and the quality of a path can be measured in different ways. The methods are evaluated empirically using real data.

Article III Network Simplification with Minimal Loss of Connectivity

We significantly extend and generalize our work in Article II. We propose methods to simplify weighted graphs by pruning more edges while minimizing the loss of connectivity. The intention is that a user can flexibly choose a suitable trade-off between simplicity and connectivity of the resulting graph (Research question I). Four algorithms are offered (Research question II), and experiments show that significant pruning can be achieved with little loss of connectivity.

Article IV Compression of Weighted Graphs

We propose models and methods for compression of weighted graphs. The idea is that nodes with similar neighbors can be grouped together, and space can be saved by presenting their shared connections only once. The task is to compress a weighted graph this way into a smaller graph that contains approximately the same information as the original graph (Research question I).

We formulate a simple and a generalized weighted graph compression problem. The simple task aims to maintain individual edge weights, while the generalized task tries to preserve longer-range connectivities between nodes. We give algorithms for the tasks (Research question II), and evaluate them on real data.

Article V Abstracting Weighted Graphs: Generalization based on Node Importances

Methodologically, this paper extends the definition of weighted graph compression of Article IV to also consider node importances. The goal is to compress a large graph into a smaller one with less error related to more important nodes (Research question I). We also propose new compression operations and improve algorithms that allow removal of low-weight edges and disconnected nodes (Research question II).

Furthermore, we propose the idea of assigning weights to nodes to summarize the metabolism information in a set of organisms. We apply the graph generalization method to the problem of understanding the metabolic biodiversity in bacteria (Research question III).

1.4 Structure of the introductory part

This introductory part of this thesis is not a summary of our research work, because the scientific materials are already described in Articles I-V. The introductory part gives the background and context of the research work, and it concludes by summing up our scientific contributions.

The following three chapters, Chapter 2, 3 and 4, give an overview of the area, and provide general answers to the three research questions. Thematically these three chapters involve three subtopics: network properties, potential abstraction approaches, and a biological application, corresponding to the three research questions.

Chapter 2 gives an overview of the first research question. It introduces the basic graph concepts, describes main properties of a graph, and then discusses two types of structural change with respect to different viewpoints. It also explores ways to measure how well information is preserved during abstraction.

The role of Chapter 3 is to give a general answer to the second research question, what kinds of operations and methods can we use to abstract a graph, by proposing potential operations and reviewing promising approaches.

Chapter 4 briefly introduces the biological problem, as a concrete application of graph abstraction as an analysis method.

Chapter 5 is a discussion of our contributions, and also explicitly answers the three research questions based on the contributions of this thesis.

Finally, Chapter 6 gives an overall conclusion.

Chapter 2

Network Properties and Evaluation Criteria

In this chapter, I first briefly introduce the concepts concerning a graph, and then discuss several important graph properties. Next, I discuss two kinds of structural change with respect to different viewpoints. Finally, I approach the first research question, *how can we measure the quality of an abstracted network,* by presenting multiple evaluation criteria that can be used to measure the quality of an abstracted graph.

2.1 Graph concepts

In this thesis, network abstraction techniques are investigated for undirected and weighted graphs. A *weighted graph* is a triple $G = (V, E, w)$, where V is a set of nodes (or vertices), $E \subset V \times V$ is a set of edges (or links), and $w : E \to \mathbb{R}^+$ assigns a non-negative weight $w(e)$ to each edge $e \in E$. Each edge links two nodes. In an *undirected* graph, an edge e connecting nodes u, v is an unordered node pair $\{u, v\}$. The vertices incident to an edge are called its *endpoints* (or *endvertices*). An edge where the two endpoints are the same node is called *self-edge.* Edges that are incident to the same two vertices are called *parallel edges.*

A *path* P is a set of consistent edges $P = \{\{u_1, u_2\}, \{u_2, u_3\}, \ldots, \{u_{k-1}, u_k\}\} \subset E$. P is a path between u_1 and u_k, that is, u_1 and u_k are the endvertices of P. A path P can be regarded as the concatenation of several sub-paths, i.e., $P = P_1 \cup \ldots \cup P_n$, where P_i is a path. Two nodes are connected if there exists a path between them. A graph is *connected* if any two nodes of the graph are connected.

A *subgraph* $G' = (V', E', w)$ of the graph $G = (V, E, w)$ is a graph where

V' is a subset of V and E' is a subset of E. E' only contains edges whose endvertices are in V'.

A graph $S = (V_S, E_S, W_S)$ is a *compressed representation (or compressed graph)* [Article IV] of the graph $G = (V, E, w)$ if V_S is a partition of (sub)set V. The nodes $v_s \in V_S$ are called *supernodes*, and edges $e_S \in E_S$ are called *superedges*. $T(v_S) \subset V$ represents the set of original nodes within the supernode v_S. A superedge $e_S = \{u_S, v_S\}$ represents all possible edges between an original node in u_s and an original node in v_S. Compression may lose information (and typically does for weighted graphs). The *decompressed graph* [Article IV] $dec(S)$ of S is a graph $dec(S) = (V', E', w')$ such that $V' = \cup_{v_S \in V_S} T(v_S)$ and $E' = \{\{u', v'\} \mid \{u_S, v_S\} \in E_S, u' \in T(u_S), v' \in T(v_S)\}$ and $w(\{u', v'\}) = w(\{u_S, v_S\})$. The decompressed graph contains exactly the same information as the compressed graph.

2.2 Graph properties

In this section, I introduce several important characteristics of network topology, network centrality, and network connectivity.

2.2.1 Network topology

Degree

In an undirected network, the *degree* of a node v is the number of edges connected to the node, and $p(k)$ is a fraction of nodes having degree k in the network. $p(k)$ thus is the probability that a randomly chosen node has degree k. A histogram plot of $p(k)$ for a given network shows the *degree distribution* of the network.

In a random graph, the degree distribution is homogeneous, binomial distribution. However, the degree distributions of many real-world networks are inhomogeneous, with a heavy right tail.

For weighted graphs, the concept of degree can also be extended to take edge weights into account, by defining the *volume* of a node as the sum of the absolute values of weights on adjacent edges [27].

The *degree variability* shows the variation in the degree distribution. Let $d = d_1, d_2, \ldots, d_n$ be the sequence of degrees , and $\langle d \rangle$ be the average degree. The degree variation $C_V(d)$ [18] is defined as

$$C_V(d) = \{\frac{\sum_{i=1}^{n} (d_i - \langle d \rangle)^2}{n-1}\} / \langle d \rangle.$$

The *degree correlation* [18] measures the correlation of a node's degree and the degree of its neighbors, that is, $\sum_{\{u,v\}\in G} d(u)d(v)$, where $d(u)$ is the degree of node u.

Path length

In a graph, the *distance* between any two nodes u, v is the length of the *shortest path* between them. If the graph is unweighted, then the shortest path is the minimum number of edges that need to be traversed from u to v. If the graph is weighted, then the shortest path refers to the length of the path that has the best quality between u and v.

The *diameter* of a network is defined as the longest distance of any pair of nodes. The *average path length* of a network is defined as the average path length between all pairs of nodes.

Clustering coefficient

In many real-world networks, if node u connects to node v and node v connects to node z, there is a high probability that nodes u and z are connected. This is known as the "small-world" phenomenon. The *clustering coefficient* of node v measures the probability that v's neighbors are connected. Let k be v's degree, and E_v represent the actual number of edges between the neighbors of v. The clustering coefficient [54] of node v is the proportion of the actual edge number between v's neighbors with respect to the maximal possible edge numbers between the neighbors,

$$C_v = \frac{2 * E_v}{k * (k-1)}.$$

The clustering coefficient of a network can then be defined as the average clustering coefficient over all nodes, that is,

$$C = \frac{1}{|V|} \sum_{v\in V} C_v.$$

An alternative definition [39] of the global clustering coefficient is the fraction of triples of nodes that have the third edge to form a triangle, that is,

$$C = \frac{3 * \text{number of triangles in the network}}{\text{number of connected triples}}.$$

Community structure

One topological property of networks, the *community structure*, shows that groups of nodes have denser connections within them, and have lower connections between them.

Radicchi et al. [42] gave two definitions of community: one in a strong sense, another in a weak sense. Let k_v be the degree of node v. Consider a subgraph $G' \subset G$, to which v belongs. Let k_v^{in} denote the number of edges connecting v to other nodes inside G', and k_v^{out} denote the number of edges connecting v to other nodes that do not belong to G'. So $k_v = k_v^{in} + k_v^{out}$.

The subgraph G' is a community in a strong sense if

$$k_v^{in} > k_v^{out}, \forall v \in G'.$$

It requires that each node has more connections within the community than with the rest of the graph.

The subgraph G' is a community in a weak sense if

$$\sum_{v \in G'} k_v^{in} > \sum_{v \in G'} k_v^{out}.$$

It only requires that the sum of connections within the community is larger than the sum of connections with the rest of the graph. If a subgraph is a community in the strong sense, then it is a community also in the weak sense.

Frequent subgraph

In a network, some subgraphs (or patterns) occur frequently. A subgraph is *frequent* if its number of occurrences is greater than or equal to a user-specified threshold. A frequent subgraph is *maximal* if none of its supergraphs are frequent. A frequent subgraph is *closed* if none of its supergraphs have the same frequency. Finding frequent subgraphs faces the problem of *subgraph isomorphism*, which is to find a mapping from a subgraph g_1 to another subgraph g_2.

Another problem is how to define the *support* (i.e., the number of occurrences) of a subgraph. If a set of graphs (or a graph database) is given, the support of a subgraph is simply the number of graphs in the given database that contains the subgraph. If a large graph is given, it is difficult to find an appropriate definition. In [61], there are two methods to count the occurrences. The first method defines that two occurrences are different if they have at least one different edge. So overlaps of occurrences of the same subgraph are allowed. But it does not have the *downward closure* *property*, which requires that the support of a graph is not smaller than the support of its super-graph. The property is important because it can reduce the search space. The second method defines that two occurrences are different if they do not share any edge, but are allowed to share nodes. This definition has the downward closure property.

2.2.2 Network centrality

Node centrality

The importance of a node often depends on its location in a graph. A simple measure of the importance of a node v is its *degree centrality*, which counts the number of edges connected to v. By this definition, nodes with more edges are more central.

The second well-known measure is *betweenness centrality* [8, 4, 10]. It measures how influential a node is in connecting pairs of nodes. A node's betweenness is the number of times the node appears on the paths between all other nodes. It can be computed for shortest paths or for all paths [11].

Closeness centrality [12] is defined as the sum of distances from a given node to all others in the network. The distance can be defined as mean geodesic distance, or as the reciprocal of the sum of geodesic distances. Other proposed measures are *eccentricity centrality* [16] and *eigenvector centrality* [3].

The above node indices evaluate the importance of a node relative to all other nodes in the graph. Other measures evaluate the importance of a node relative to a given set of nodes (see, e.g., [55], [Article V]).

Edge centrality

In a weighted graph, each edge has a weight to show its strongness. Furthermore, their positions within a graph also show how important they are.

The most applied index is *edge betweenness* [38], which measures how often the edge is present in the paths between node pairs. It can be calculated by using shortest-path betweenness, random-walk betweenness and current-flow betweenness.

Another index is *Birnbaum importance* [2]. It is directly defined on (Bernoulli) random graphs where edge weights are probabilities of the existence of the edge. The Birnbaum importance of an edge depends directly on the overall effect of the existence of the edge. An edge whose removal has a large effect on the probability of other nodes to be connected, has a high importance.

2.2.3 Network connectivity

In this subsection, we introduce a generalization of distance (to path quality), and average distance (to connectivity of a graph).

In a weighted graph, two nodes u and v may be connected by a direct edge, or one or several paths, or none in a disconnected graph. How strong their connection is is decided by the qualities of paths between them. A *path quality function* $q(P) \to \mathbb{R}^+$ measures the quality of a path P. The form of path quality function depends on the type of graph. For example, in a random graph, the quality of a path P is the probability that all of its edges co-exist: $q(P) = \pi_{\{u,v\} \in P} w(\{u,v\})$. In a flow graph, the quality of a path is the capacity of the worst edge along the path: $q(P) = \min_{\{u,v\} \in P} w(\{u,v\})$.

A simple way to quantify the connectivity between two nodes is to measure the quality of the best path between them [Articles II, III]. The *connectivity between two nodes* is defined as

$$C(u,v;E) = \begin{cases} \max_{P \subset E} q(P) & P \text{ is a path between } u \text{ and } v, \\ -\infty & \text{otherwise.} \end{cases}$$

For instance, in a flow graph, the connectivity between two nodes is the maximal flow that can pass from u to v (or from v to u). In a random graph, the connectivity between two nodes is the largest probability that a path exists between them.

A more general measurement takes path length into account. It measures the quality of the best path within a certain path length λ between two nodes [Article IV]. The $\lambda -$ *connection* between a pair of nodes is defined as

$$C_\lambda(u,v;E) = \begin{cases} \max_{P \subset E, |P| \leq \lambda} q(P) & P \text{ is a path between } u \text{ and } v, \\ 0 & \text{otherwise.} \end{cases}$$

Another more complicated measurement takes all paths' qualities into account. For example, in a probabilistic graph, it measures the probability that a path exists between two nodes:

$$C(u,v;E) = 1 - \Pi(1 - q(p)).$$

A natural measure for the *connectivity of a graph* [Article III] is then the average connectivity over all pairs of nodes, that is,

$$C(V,E) = \frac{2}{|V|(|V|-1)} \sum_{u,v \in V, u \neq v} C(u,v;E),$$

where $|V|$ is the number of nodes in the graph.

2.3 Two types of structural change with respect to different viewpoints

In order to abstract a graph, some information is inevitably removed or transformed. However, with regard to an application requirement, the removal or change of some information may only result in little loss. In this section, we discuss two kinds of structural change based on whether the user's preferences are involved or not. One is unimportant information, decided by the graph itself. The other is irrelevant information, depending on a user's preference.

2.3.1 Unimportant information

Depending on the application and its needs, some parts of a network can be less important than others. Below we list some examples of unimportant information.

- If an application focuses on the connectivities of the graph (or the connectivities of nodes), then an edge whose removal does not change the connectivity between the edge's endvertices is unimportant; and an edge, whose removal slightly changes the connectivity of the graph, has only some importance.

- Another example is if an application is interested in the communities of the graph, then a node that is marginally connected to communities is less important; and so is potentially an edge that links two communities.

- The third example is if an application is interested in the main topology of a graph. Then a node with low centrality is less important; and so is an edge.

2.3.2 Irrelevant information

Irrelevant information is chosen by the user. It can be categorized into two groups: information that the user is not interested in; and information that the user is already familiar with.

We first list some examples of the information that a user may not be interested in.

- A user may be interested in the connectivities between all nodes. Then how the node degrees distribute is irrelevant information.

- A user may be interested in the best paths of a graph. Then the paths that do not contribute to the best paths are irrelevant information.
- A user may be interested in how two (or more) nodes connect with each other. These specified nodes and their connections are the information that the user is interested in. The nodes and edges that are not central to the connections of the specified nodes are irrelevant information.
- A user may be interested in how communities connect to each other. It follows that the nodes that are located within the communities are not the user's focus.
- A user may be interested in the structure of a certain community. It follows that the nodes that are outside the community are irrelevant.
- A user may want to analyze some types of nodes. Nodes of other types are irrelevant information.

In general, discovering new knowledge is the goal of scientific research. The information that a user already has is not "new", and thus can be regarded as irrelevant.

We next list two examples of information that a user may already know.

- A user may want to extract some useful patterns in a graph, so that patterns he (or she) already knows are irrelevant information. For example, in a biological network, nodes labeled "gene" always connect with nodes labeled "protein" with an edge of type "codes for".
- A user may want to explore communities within a graph, so that the communities he (or she) already knows are irrelevant information. For example, in a social network, the user may be familiar with the community of a research group, or the community of a research institute, or the community of a research area.

2.4 Evaluation criteria

Each application has its own requirement. For example, one application may focus on the topological structure of a graph, another application may focus on the connectivity of a set of nodes, another application may be interested in the hidden communities, and so on. The evaluation criteria in these applications are different, and are derived from application requirements. In this section, we discuss the main abstraction evaluation criteria.

2.4.1 Topological change

Some applications want the abstracted graph to have similar (or scaled-down) properties of topology as compared to the original graph [30].

Degree

Suppose G is the original graph, and H is an abstracted graph. Let P_G be the distribution of number of nodes with respect to degree in the original graph, and P_H be the distribution in the abstract graph. If the abstraction is good, then its degree distribution is proportional to the degree distribution of the original graph [30], that is, $P_H \propto P_G$.

The second criterion is to measure how much each node degree is altered on average. Let $d_G = d_1^G, d_2^G, \ldots, d_n^G$ be the sequence of degrees in the original graph G, and d_H be the degree sequence in an abstracted graph H. The *change of node degree* [18] is measured by using Mallows distance which is the L_p distance between two sequences, that is,

$$Mallows_p(d_G, d_H) = \{\frac{1}{n}\sum_{i=1}^{n} |d_i^G - d_i^H|^p\}^{1/p}.$$

Others [18] also measure the change of maximum degree, the change of degree variation, and the change of degree correlation.

Path

One criterion is to measure the change of the average path length. Let $\langle P_G \rangle$ be the average path length of the original graph G, and $\langle P_H \rangle$ be the average path length of the abstracted graph H. The *change of average path length* is $\langle P_G \rangle - \langle P_H \rangle$. If the value is positive, it means the average path length is shortened.

Clustering coefficient

One measure is to evaluate the change of clustering coefficient. Let C_G be the clustering coefficient of the original graph G, and C_H be the clustering coefficient of the abstracted graph H. The *change of clustering coefficient* is $C_G - C_H$.

2.4.2 Connectivity change

Some applications need the information of the connectivities between nodes or the connectivity of a graph. In such applications, abstraction techniques need to maximally maintain the connectivity information.

One evaluation criterion is to measure how much connectivity of the graph is maintained in an abstracted graph. Suppose G is the original graph, and H is the abstraction result. The connectivity of the original graph is $C(V_G, E_G)$, and the connectivity of the resulting graph is $C(V_H, E_H)$. The *ratio of connectivity maintained* [Article III] in the abstraction is defined as

$$r(H, G) = \frac{C(V_H, E_H)}{C(V_G, E_G)}, \text{ where } V_H \subseteq V_G, E_H \subseteq E_G.$$

The range of r is from 0 to 1. When $r = 1$, it means that the connectivity of the original graph is maintained. When $r < 1$, it implies that some connectivities in the result are lost. The criterion is used in Article III in a context where edges are removed and connectivity can only decrease in abstraction.

Another evaluation criterion is to measure how much connectivities between nodes are changed in an abstracted graph. Let $C_G(u, v)$ be the connectivity between u and v in G, and $C_H(u, v)$ be their connectivity in H. The *connectivity change* is the square root of the sum of change of connectivities over all pairs of nodes, that is,

$$l(G, H) = \sqrt{\sum_{u \in V_G} \sum_{v \in V_G} (C_G(u, v) - C_H(u, v))^2}.$$

The range of $l(G, H)$ is $[0, +\infty)$. When $l = 0$, it means all pairs of nodes have maintained their original connectivities. When $l > 0$, it implies some connectivities are changed.

In some applications, a user can specify an area he wants to study. Nodes within the area have higher importance than the ones outside of that area. The connectivity change between a pair of nodes is weighted by the node importances. Thus, the total connectivity change is

$$l(G, H) = \sqrt{\sum_{u \in V_G} \sum_{v \in V_G} \mathcal{I}(u, v)(C_G(u, v) - C_H(u, v))^2}.$$

2.4.3 Adjacency matrix change

Some applications need to use the adjacency matrix of a graph, that is, the adjacent edges of nodes and edge weights. During abstraction, techniques require maintaining as much original information as possible.

The evaluation criterion measures how much the adjacency matrix is changed from the original graph to an abstracted graph. Suppose G is the original graph, and H is the abstracted graph. The *adjacency matrix change* [Article IV] between G and H is the square root of the sum of changes over all edges, that is,

$$sc(G,H) = \sqrt{\sum_{u \in V_G} \sum_{v \in V_G} (w_G(u,v) - w_H(u,v))^2}.$$

If nodes have importances, or a user assigns weights to nodes, then the *adjacency matrix change* [Article V] between G and H is weighted by node importances $\mathcal{I}$, that is,

$$sc(G,H) = \sqrt{\sum_{u \in V_G} \sum_{v \in V_G} \mathcal{I}(u,v)(w_G(u,v) - w_H(u,v))^2}.$$

Chapter 3

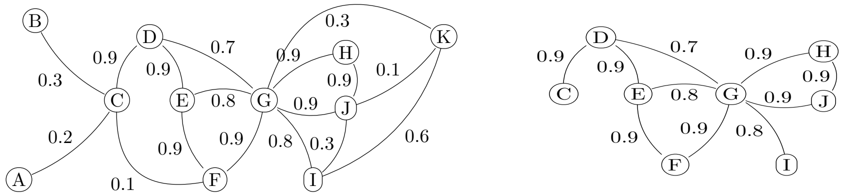
Network Abstraction Approaches

In this chapter, I approach the second research question: *what kind of operations and methods can we use to abstract a graph*, by discussing three types of abstraction operations: prune, partition and replace; and two different viewpoints: objective and subjective. It is a brief summary of Article I. In addition, I discuss the novelty of our proposed approaches with respect to the related work.

3.1 Operation types

We classify the operations that can produce abstractions of networks into three categories: *prune*, *partition*, and generalize by *replacing*.

1. *Prune nodes and edges.* This approach simplifies the structure of a network by removing peripheral or irrelevant nodes or edges, with the aim of keeping only the most interesting or relevant nodes and edges. Figure 3.1 removes edges with weights below 0.7, and it follows that the separated nodes are removed as well.

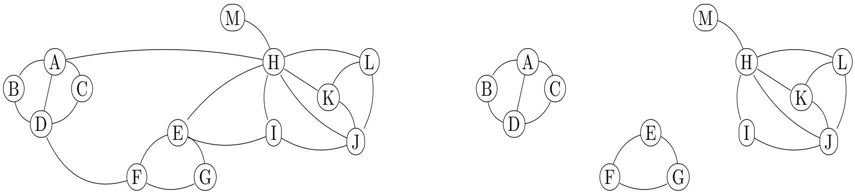


(a) Original graph

(b) Result graph

Figure 3.1: Prune operation: some nodes and edges are removed from the original graph.

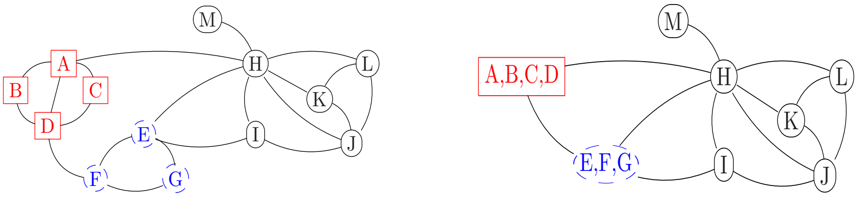
2. *Partition a network into smaller ones.* An example is shown in Figure 3.2. The long connections between nodes are cut, and compact subgraphs are subsequently obtained. The resulting graph is abstracted, as mutual dependencies are reduced. Each small subnetwork now is easier to explore.



(a) Original graph (b) Result graph

Figure 3.2: Partition operation: the original graph is divided into separate parts.

3. *Replace parts of a network by more general structures.* Generalization may, for instance, replace a path with a single edge, or parallel paths with a single one, or a subgraph by a node, in order to simplify the network. For example, in Figure 3.3, nodes A, B, C, D have strong connections, and nodes E, F, G have close relationships, so both are replaced by general nodes.



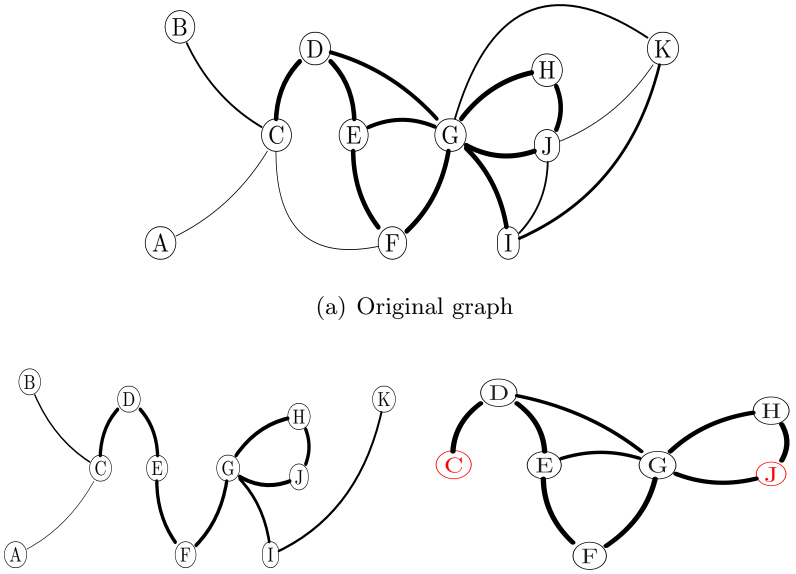
(a) Original graph (b) Result graph

Figure 3.3: Replace operation: parts of the graph are replaced by more general nodes.

3.2 Two viewpoints

Based on whether the user's preferences are involved or not, we classify techniques into two groups: *objective* and *subjective*. An objective technique abstracts a graph with regard to the information only available from the graph, such as graph topology, node importance, or edge weights. It disregards user-specific emphasis on any part of the network. In contrast, a subjective technique allows the user to indicate which parts of the network should retain more details, and abstracts a graph based on the user's preference.

An example is shown in Figure 3.4. Thicker lines mean stronger connections. In the result of objective pruning (Figure 3.4 (b)), all nodes are preserved and the strong connections between nodes are preserved as well. Only weak connections whose removal does not separate the result are removed. In subjective pruning, in turn (Figure 3.4 (c)), the strong connections between two interesting nodes C and J are maintained, but other peripheral nodes and edges are removed.



(a) Original graph

(b) Objective result

(c) Subjective result

Figure 3.4: Pruning results from different viewpoints. (a) Original graph. (b) A result of objective pruning. (c) A result of subjective pruning, when the user wants to focus on connections between C and J.

3.3 Existing approaches

Existing approaches can now be classified based on two characteristics: the type of operation and the point of view. Table 3.1 gives a summary of the related work reviewed in Article I.

		Points of view	
		Objective	Subjective
Operations	Pruning	Relative neighborhood graph [52, 25] T-spanner [40] Network simplification [***Articles II, III***] PathFinder network [44] Graph sampling [43, 30, 23, 32] Node centrality [13, 47]	Relevant subgraph extraction [15, 9, 51, 21, 22] Detecting interesting nodes or paths [55, 31] Personalized PageRank [26, 17]
	Partitioning	Graph partitioning - Spectral bisection method [41, 19] - Geometric methods [33, 1] - Multilevel graph partitioning [28, 20] - Kernighan-Lin algorithm [29] Hierarchical clustering [45] Edge betweenness [38] Attributed graph clustering [59, 60]	Constrained clustering [53] Supervised clustering [7]
	Replacing	Graph compression [35, 36], [***Article IV***] Frequent subgraphs [24, 5]	Graph summarization [48, 58], [***Article V***] Replacing user input subgraph - Exact search [46, 57] - Similar search [49, 50]

Table 3.1: Existing approaches

3.4 Novelty of our approaches

The pruning approach as presented in our work [Articles II, III] removes redundant and unimportant edges. The proposed methods have wide generality: they are applicable to different kinds of graphs, such as flow networks and random graphs. The relative neighborhood graph [52, 25] is a special case of the framework proposed in Article II, as it only connects relatively close pairs of nodes. Methods for finding PathFinder networks [44] are also similar to Article II, as they prune an edge if there is a shorter path between the endpoints of the edge. They use Minkowski distance as a metric to compute the distance of paths. Article III differs from all these works in a significant manner: it allows the loss of the quality of the best paths.

The proposed replacement-based approach in Article IV and Article V groups nodes that have similar neighborhoods into supernodes, and generalizes their edges to superedges. The work most closely related is by Navlakha et al. [36], who proposed to construct graph summaries of unweighted graphs by grouping nodes and edges to supernodes and superedges, and by Tian et al. [48], who proposed to summarize labeled graphs by grouping nodes according to node labels and edge attributes. We generalize these approaches in three important and related directions: to weighted graphs, to long-range, indirect (weighted) connections between nodes, and to taking node importances into account.

Chapter 4

Application to Metabolic Networks

Abstracted networks have various applications: interactive visualization, community discovery, biological module revealing [37], graph pattern mining [6], and so on. In this chapter, I focus on one bioinformatics application. The details of the work are described in Article V. The purpose here is to use this application as an example to evaluate *how useful an abstracted graph is*.

The target of this application is to better understand the metabolic biodiversity in bacteria. In this work, we apply a weighted graph compression method to metabolic networks.

4.1 Metabolic networks of Archaea and Eubacteria

Living organisms are self-sustaining autocatalytic chemical reactions. A subset of these reactions are termed *metabolic*, these reactions convert input chemicals into biochemicals. The input chemicals may be simple, such as the minerals, water, oxygen, and carbon dioxide that plants require, or the complex organic molecules required by animals. The output chemicals are similar in all forms of life: DNA, RNA, proteins, etc.

In a metabolic reaction, an enzyme (a protein that is encoded by a gene) catalyzes the chemical reaction of a set of biochemicals into another set of biochemicals. The number of known enzymes is now in the order of a few thousand. These can be combined to form a super-metabolic network containing all known reactions. Any particular genome will only have genes that encode a subset of these enzymes. Biochemists have for ease of understanding traditionally dissected metabolism into separate *pathways*: glycolysis, the TCA cycle, etc. In the order of one hundred pathways have been recognized.

The most versatile metabolisms are found in bacteria, where there exists a vast diversity of different biochemical reactions. Over the last fifteen years there has been a revolution in the ease and efficiency of gene sequencing. The complete genomes of over a thousand bacterial species are now known. This has resulted in an explosion in the amount that is known about bacterial metabolism. The biological challenge is now how to interpret this large amount of complex data to generate biological knowledge.

According to the three-domain system [56], the most fundamental split in the evolution of life was between the lineage leading to the *Eubacteria* and the *Archaea*. It is unclear when this happened, but it was probably 2-3 billion years ago. The bacteria, that is, the *Eubacteria* and *Archaea*, dominate the nutrient cycles on the planet. The third major group are the Eukaryotes (we are Eukaryotes), these evolved from a symbiosis between *Archaea* and *Eubacteria*, these dominate macroscopic life.

We are interested in the biodiversity within metabolism in *Archaea* and the *Eubacteria*, the two main branches (kingdoms) of life.

One of the most fundamental questions in evolutionary biology is to what extent the paths that evolution has taken are stochastic, and to what extent they are determined by constraints imposed by the environment and biochemistry. Eminent evolutionary biologists have taken different views of this question of stochasticity. Stephen Jay Gould in many essays, and most notably in his book *Wonderful Life* [14] argued for contingency in evolution. For him evolutionary biology, in seeking to explain the past, was a historical science; if the process would somehow be run again then one would expect a radically different result. In contrast, Simon Conway Morris [34] has argued that the constraints on living organisms are such that it is likely that evolution would take broadly the same path and intelligent organisms such as humans are likely to evolve.

The central problem with scientifically investigating this question is that it is not possible to repeat the experiment - evolution. However, it is possible to gain some understanding of the stochasticity of the problem by looking at cases where evolution started from similar starting points.

Our idea is to use the newly available data on the biodiversity of metabolic pathways to understand how pathways have evolved since the divergence of the *Archaea* and the *Eubacteria*.

4.2 Weighted metabolic networks and compression

In studying metabolism the most natural abstraction is to use graphs. A number of different mappings and types of graph have been used. The one we consider most natural is for enzymes to be nodes, and metabolites to be edges. The super-metabolic network can thus be represented as a graph with a node for every possible enzyme, and the metabolic complement of a genome is a specific instantiation of this graph. A set of N genomes would then be represented as a set of such instantiations of the super-metabolic network. Unfortunately, in our view, this representation is too complex to be easily analyzed. We therefore propose the use of weighted graphs as a simpler alternative.

In these taxonomy graphs the weights are based on in how many genomes a particular enzyme exists. Enzymes with high weights are ubiquitous and those with low weights rarely occur. Such a weighted graph summarizes the information in the set of these instantiations of the meta-metabolic network. We are unaware of the use of weighted graphs in these ways before.

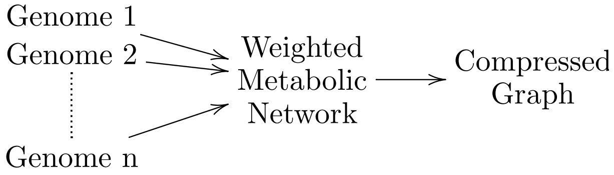


Figure 4.1: The bioinformatic flowchart.

Combining weighted graphs and compression is useful to refine the large amount of bacterial metabolic data to answer biological questions. The bioinformatic flowchart is shown in Figure 4.1. Specifically we propose to compress weighted metabolic networks for analysis of the two large weighted graphs: one for *Archaea* and one for *Eubacteria*. This will enable us to understand the metabolic biodiversity between two kingdoms.

Chapter 5

Summary of Contributions

This chapter first discusses our contributions in Articles I-V, and then lists future work at the end of this chapter.

5.1 Defining the field of network abstraction

In this thesis, we propose *network abstraction* as a research area. It includes various types of existing approaches in data mining and also raises new research problems.

According to the characteristics of different approaches to network abstraction, we propose three classes of abstraction operations: prune, partition and replace. In Article I, we review the potential approaches and classify them based on a clear structure: one axis is the type of operation, and another axis is the point of view.

Thematically, our method development includes two sub-topics: network simplification and network compression. Network simplification aims to simplify a network by removing redundant and unimportant edges, and network compression compresses a network by grouping nodes and edges. This type of compression can also be considered as a type of generalization. We will next look at these contributions in more detail.

5.2 Network simplification

A network structure can be simplified by removing edges. In a weighted network, nodes can be connected by several paths with different qualities. For example, in Figure 5.1, nodes A and B are connected by one edge and two other paths. We suppose the graph in Figure 5.1 is a probabilistic graph. The direct edge $\{A, B\}$ has such a weak quality, 0.1, that it can be

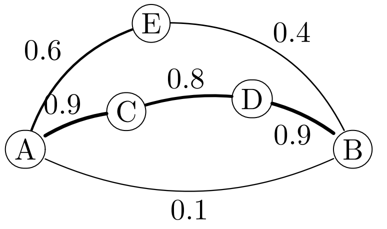


Figure 5.1: $\{A, B\}$ is redundant in the probabilistic graph.

removed without affecting the qualities of best paths between any pair of nodes.

In Article II, we propose methods to simplify a network by removing redundant edges. We define the *redundant edge* based on the definition of network connectivity in Subsection 2.2.3. An edge is *redundant* if and only if its removal does not affect the best path quality for any pair of nodes. The quality of path $ACDB$ in Figure 5.1 is 0.9*0.8*0.9 = 0.648, and the quality of path AEB is 0.6*0.4=0.24. The edge $\{A, B\}$ is thus redundant.

We propose two ways to identify redundant edges. One is *global search*. It evaluates the redundancy of an edge by finding and evaluating the best path between its endpoints. The other is *triangle search*. It evaluates the non-redundancy by checking the qualities of paths consisting of two edges. Then we propose two ways to search redundancy: *iterative* and *static*. Combining the ways of searching and the ways of identifying, we present four algorithmic variants.

The framework presented in Article II is applicable to a large variation of network types and path qualities, with the general assumption that path quality functions obey monotone property, that is, replacing a segment of a path by a better one never decreases the quality of the whole path. Experiments were carried out on flow graphs and probabilistic graphs. Results show that a certain number of edges can be removed without affecting the best path connectivity between any pair of nodes.

The number of edges that can be removed by the methods in Article II is limited, due to the requirement that the original connectivity of the graph should be maintained. In Article III, we propose to relax this constraint and allow removing more edges. The intention is that the user can flexibly choose a suitable trade-off between the simplicity and connectivity of the resulting graph. For example, in Figure 5.1, the original connectivity between E and B is 0.4, so $\{E, B\}$ is the best path between them. After removing edge $\{A, B\}$, the connectivity between E and B does not change. If we remove an extra edge $\{E, B\}$, then there is only one path $EACDB$

between E and B. The quality of path $EACDB$ is 0.6*0.9*0.8*0.9=0.389. The connectivity between E and B only decreases from 0.4 to 0.389 and other pairs of nodes still have original connectivities, but the network is much simpler. The problem then is to simplify a network structure by removing edges while minimizing the loss of connectivity.

The evaluation criterion that measures the ratio of connectivity maintained in an abstracted graph is described in Subsection 2.4.2. The way of computing the ratio of connectivity kept is multiplicative. That is, the ratio of connectivity kept after removing an edge set E_R can be represented as the product of ratios of connectivity kept for each edge in any permutation. Based on this property, we derive a lower bound on the ratio of connectivity kept, which is the ratio of connectivity kept for the endpoints of the removed edge.

Utilizing the two ways of computing the best (or approximately best) path for the endpoints of the edge in Article II, *global search* and *triangle search*, we derive two increasingly fast and approximate ways of bounding the ratio of connectivity kept. The first lower bound is to compute the exact ratio kept for the endpoints of the removed edge based on finding the best alternative path between them. The second further lower bound is based on the quality of the alternative best path of length two.

We then present four algorithms to complete the simplification task with different trade-offs between connectivity kept and time complexity. The first one is a naive approach, which simply prunes a fraction of weakest edges by sorting edges according to edge weights. The second one is the computationally demanding brute-force approach, which iteratively computes the exact ratio kept for each edge and prunes the one whose removal best keeps the connectivity. The third approach, path simplification, uses the global best search to compute the lower bound of ratio of connectivity kept and prunes the edge with the largest lower bound. The fourth one, the combinational approach, uses the triangle search to compute the lower bound, and then if needed, uses the global best search to compute the lower bound to remove additional edges.

We carried out experiments on real graphs to compare the performances of the four algorithms. Experiments showed that the naive approach in most cases is the fastest one, but it induces a large loss of connectivity. The brute force is very slow in selecting the best removed edge set. The path simplification and combinational approach were able to select a good edge set with a short time. Furthermore, compared with the methods in Article II, experiments showed that significantly more edges can be removed with little loss of connectivity.

5.3 Network compression

One option to simplify the large structure of a network into a smaller one is to group nodes and edges into supernodes and superedges. In what we call network compression, we group nodes that connect with a set of similar neighbors. An example is in Figure 5.2. Node A has neighbors C and D. Node B has neighbors C, D and E. Nodes A and B can be grouped into a supernode AB with connections to all previous neighbors. The connections are weighted to the mean value of original edge weights.

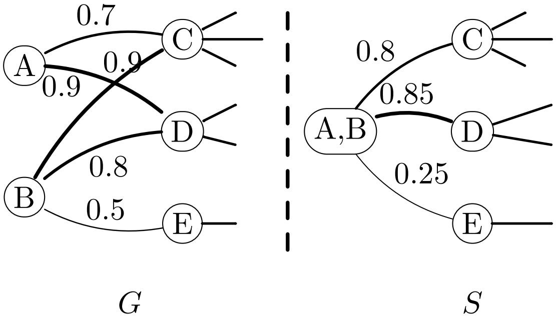


Figure 5.2: Grouping nodes A and B into supernode A, B. G is the original graph, and S is one version of the compressed graph.

In Article IV, we address the problem of compressing weighted graphs. The goal is to compress a weighted graph into a smaller one that contains approximately the same information as the original graph. The compression problem now consists of how to evaluate the quality of a compressed graph, how to choose supernodes, and how to assign weights to superedges.

The evaluation criterion in Article IV is to measure the dissimilarity between the original and the compressed graphs. We present two measurements. One measures the adjacency matrix change between the original and compressed graph, as described in Subsection 2.4.3. Based on this, we formulate the *simple weighted graph compression problem*, which aims to minimize the compression error with respect to the edge weights.

The other measure is the connectivity change between nodes, as described in Subsection 2.4.2. The connectivity between nodes can be measured in a more general form by taking path length λ into account, as described in Subsection 2.2.3. We then propose the *generalized weighted graph compression problem*, where the goal is to produce a compressed graph that maintains connectivities across the graph. The simple weighted graph compression problem is an instance of the generalized problem with paths of length one.

A compressed graph is obtained through a series of merge operations, merging two supernodes at a time to a new supernode. The weights of superedges are assigned to minimize the dissimilarity. In the simple weighted graph compression problem, when merging two supernodes, the structural changes include the change of edge weights between all nodes that are inside the supernodes and their neighbors. The optimal superedge weight is the mean of the original edge weights (including zero-weight edges for those pairs of nodes that are not connected by an edge). In the generalized problem, it is much more complex to compute the optimal superedge weight. Edge weights contribute to best paths, and it is difficult to optimize the distance by setting each superedge weight independently. We then propose an efficient and approximate solution, that is, the mean of the $\lambda-$connectivities.

The goal in Article IV is to compress a graph into a specified size while minimizing merger errors. The distance measurements satisfy the triangle inequality: the total merger error is upper bounded by the sum of individual merger errors. In the simple compression problem, the individual merger error is simply the change of edge weights between the new supernode and its neighbors. However, in the generalized problem, the individual merger error is complex to compute. We derive an upper bound for the connectivity change for any pair of nodes, and then we get an upper bound for the individual merger error.

We propose four algorithms to compress a weighted graph. They all work in an incremental fashion, merging a (super)node pair at a time. The *brute-force greedy algorithm* computes the effects of all possible mergers, and executes the best one. The *thresholded algorithm* merges two nodes whose merger error is within the current threshold. The *randomized semi-greedy algorithm* each time first randomly chooses a node, and then chooses another optimal node whose merger produces the smallest error. The *random pairwise compression* method simply merges two nodes at random. Experimental results show that graphs can be efficiently compressed a lot with little loss of information.

Article IV treats nodes equally. However, nodes can have different importances to the user. For example, if the user wants to understand the connections between the key nodes that s/he is interested in, then the nodes that are central to the key nodes will have higher weights than others. In Article V, we propose compression of graphs that have weights on both nodes and edges. An example is in Figure 5.3. Node importances are given

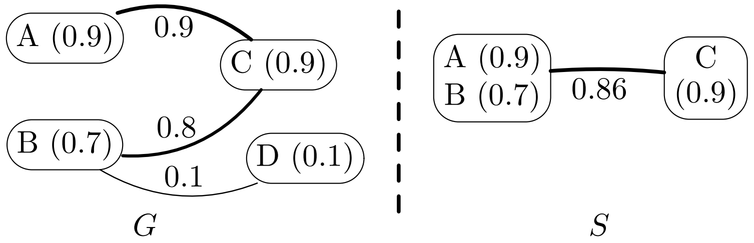


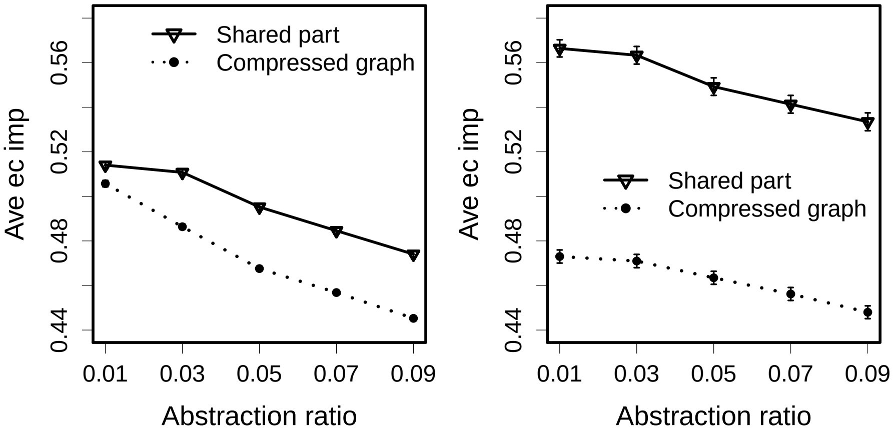
Figure 5.3: Grouping nodes A and B into supernode AB with respect to node importances. G is the original graph, and S is one version of the compressed graph.

in parenthesis. The goal is to compress a graph into a smaller one with less error related to more important nodes. The intention is that the abstracted graph will allow the user to focus on the information most relevant to a specific application and to ignore other details.

We introduce a definition of node importance in a weighted graph when a set of key nodes is given by the user, and also show how to balance the overall importances over key nodes. The compressed graph will be useful for personalized visualization, as it gives more details between the key nodes while less relevant information will be abstracted more.

In Article V, we present two new elementary compression operations: node-pair merger with possible omission of (super)edges, and individual edge deletion. The merger operation differs from the merger operation in Article IV where no (super)edge is removed. Operations are chosen according to normalized error, that is, error per size reduction. We consider two ways to compute normalized error. The first one is *standard* normalized error, which is the exact distance between the original and the current graph divided by the total size reduction. The second one is *local* normalized error, following the idea in Article IV: it is the distance between the previous and the new state divided by the additional size reduction.

Following Article IV, we have two main algorithms, the *brute-force* and *randomized* ones. Combining with the two normalized errors, we have four algorithmic variants in total. We carried out experiments on real-world graphs to evaluate the performances of the new operations and the four algorithms. Experiments show that node importance can guide the process to a better compression in both normalized error cases, and that node pair merger performs better than the merger operation in Article IV.



(a) Archaea (b) Eubacteria

Figure 5.4: Average importance of enzymes in the compressed graphs, and average importance of enzymes in the shared part of the compressed graphs.

5.4 Compression of metabolic networks

In Article V, we also apply weighted graphs and compression to metabolic networks, in order to better understand the biodiversity and evolution of bacterial metabolic pathways. The background of this application was introduced in Chapter 4.

We apply one of our compression methods to compress metabolic networks utilizing enzyme weights as node importances. After compressing the graphs for *Archaea* and *Eubacteria*, we compute the average enzyme importance in the compressed graphs, and in their shared part. The results are shown in Figure 5.4.

We notice that the average enzyme weight in the shared part is much higher than the average weight in either kingdom-specific compressed graph. This implies that the shared part is essential to both kingdoms, and probably illustrates the conservation of evolution. Hence, the shared part of what remains after compression is core metabolism, and more likely to have been present in the ancestral organism.

5.5 Answers to the research questions

I *How can we measure the quality of an abstracted network?*

In Section 2.4, we already discussed the properties that can be preserved in an abstracted graph. In our work, we propose methods to preserve two specific graph properties. The first one is the qualities of the best paths, proposed and used in Articles II and III. The second one is the adjacency matrix, used in Articles IV and V.

II *What kind of operations and methods can we use to abstract a graph?*

In Article I, we categorize network abstraction approaches by three elementary operations: *prune*, *partition* and *replace*, and by two viewpoints: *objective* and *subjective*. We also review existing techniques using this taxonomy.

In network simplification, we apply the *prune* operation and propose objective pruning methods. We delete unimportant edges (Subsection 2.3.1) in Articles II and III.

In network compression, we apply the *replace* (and *prune*) operation. In Article IV, we propose an objective replacement method to group nodes and edges into supernodes and superedges. In Article V, we propose a subjective replacement method, in which we also allow removal of nodes in compression.

III *How useful is an abstracted graph?*

In Article V, we apply the graph compression method to weighted metabolic networks. The average enzyme weight in the shared part of compressed graphs is higher than the average weight in either kingdom-specific compressed graph. The shared part of compressed graphs thus are likely to be informative about the metabolism of the common ancestor of all life. Furthermore, the compressed graphs provide a hierarchical structure for users to understand the metabolisms from different abstraction levels.

5.6 Open questions

We now discuss future work based on the results of this thesis.

First, as discussed above, our works investigate undirected graphs. One open question is whether these methods are still applicable to directed graphs. For example, the quality of the best path from a node u to another node v is the same as from v to u in an undirected graph, but not necessarily in a directed graph. It is also possible that there is no path from v to u.

In network simplification, we can easily modify the calculation of connectivity between nodes to take path direction into account. We can also use methods in Articles II and III to remove edges to simplify directed graphs.

In network compression, the modification is a bit more complicated. A superedge represents all edges between nodes in its endpoints. The directions need to be added into superedges. It brings up several questions: how to decide the direction of a superedge, how to assign an optimal weight to a superedge, and how to calculate the compression error. In the *simple weighted graph compression* problem, one solution is that the direction of a superedge is the majority direction of edges represented by it. Edges with opposite directions should incur a larger error, e.g. $w_o(u, v) - sgn((u, v)) \cdot w_s(u, v)$. Another solutions is to consider different directions separately. This would introduce less error. However, the modification in the *generalized weighted graph compression* is even harder, since it needs to take path direction into account.

Second, the results from our network simplification methods are connected graphs. An edge whose removal cuts the graph will not be considered, because we aim to preserve the connectivities of all node pairs. This raises the question if removing some weak edges can lead to a good abstraction even if it may cut the graph. Suppose a node connects with the rest of the graph only through a single edge. Removal of such an edge separates the node from the rest of the graph. However, this may be insignificant overall, and in a sense, such removal could be meaningful.

Third, to evaluate results from a subjective viewpoint is difficult. For example, one advantage of an abstract graph is that the structure is simpler and smaller, which makes visualization convenient. We have not carried out any user studies to investigate the usefulness of network abstraction for an end user.

Last, in our work, the main goal is to give an overview of a large graph which maintains some essential qualities of the original graph. A related problem is (social) network anonymization, where the goal is to preserve some important properties (the “utility” of the network) but hide identities of nodes or some other sensitive information [18]. The methods in privacy preserving graph mining try to transform the graph to avoid re-identification of individuals (nodes), but still maintain the utility of the transformed graph. Take graph compression as an example of the similarities of the applications. By requiring that each supernode consists of at least k nodes, one achieves k-anonymity: any individual (node) is identical with as least $k - 1$ other individuals. It would be interesting to study the connections between these two research areas further.

Chapter 6

Conclusions

Networks exist in various domains, such as biology, chemistry, and sociology. Large graphs are difficult to visualize and understand. In different areas of computer science, several approaches have been proposed to help understand graphs, like partitioning, sampling, and extracting subgraphs.

In this thesis, we defined the field of network abstraction, including all approaches that aim to produce a simpler and useful network from a large one to help a user understand networks more easily.

We first studied simplifying a graph structure by removing edges. We provided definitions of redundant edges and unimportant edges, and proposed methods that are applicable to different kinds of graphs. The experiments show that graphs can have a fewer number of edges while the connectivity of the original graph is well maintained.

We then investigated presenting a graph at a higher abstraction level by grouping nodes and edges. Graphs can be compressed based on the graph structure alone, or with respect to node importances which reflect the emphasis of the application. We also have shown the utility of the compressed graph through a biological application.

The contributions of this thesis include discussion of the network abstraction field and proposing approaches for two specific cases. The area of network abstraction is large, and the solutions presented in this thesis are only one part of it. By identifying this as a research area and proposing some novel methods, we believe that the work provides new insight into understanding and visualizing graphs.

References

[1] M. J. Berger and S. H. Bokhari. A Partitioning Strategy for Nonuniform Problems on Multiprocessors. *IEEE Transactions on Computers*, 36(5):570–580, 1987.

[2] Z. W. Birnbaum. On the importance of different components in a multicomponent system. In *Multivariate Analysis - II*, pages 581–592, New York, 1969. Academic Press.

[3] P. Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2(1):113–120, 1972.

[4] U. Brandes. A Faster Algorithm for Betweenness Centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[5] B. Bringmann and S. Nijssen. What is Frequent in a Single Graph? In *Proceedings of the 12th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD'08, pages 858–863, Berlin, Heidelberg, 2008. Springer-Verlag.

[6] C. Chen, C.X. Lin, M. Fredrikson, M. Christodorescu, X. Yan, and J. Han. Mining Graph Patterns Efficiently via Randomized Summaries. *Proceedings of the VLDB Endowment*, 2(1):742–753, 2009.

[7] C. F. Eick, N. Zeidat, and Z. Zhao. Supervised Clustering – Algorithms and Benefits. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, pages 774–776, Washington, DC, USA, 2004. IEEE Computer Society.

[8] M. Everett and S. P. Borgatti. Ego network betweenness. *Social Networks*, 27(1):31–38, January 2005.

[9] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast Discovery of Connection Subgraphs. In *Proceedings of the tenth ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, pages 118–127, New York, NY, USA, 2004. ACM.

[10] L. C. Freeman. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40:35–41, 1977.

[11] N. E. Friedkin. Theoretical Foundations for Centrality Measures. *American Journal of Sociology*, 96(6):1478–1504, 1991.

[12] S. Gert. The centrality index of a graph. *Psychometrika*, 31(4):581–603, December 1966.

[13] A.C. Gilbert and K. Levchenko. Compressing network graphs. In *Proceedings of the LinkKDD workshop at the 10th ACM Conference on KDD*, 2004.

[14] S.J. Gould. *Wonderful Life: The Burgess Shale and the Nature of History.* Norton, 1990.

[15] M Grötschel, C. L. Monma, and M. Stoer. Design of survivable networks. In *Handbooks in Operations Research and Management Science*, volume 7, pages 617–672, 1995.

[16] P. Hage and F. Harary. Eccentricity and centrality in networks. *Social networks*, 17(1):57–63, 1995.

[17] T. H. Haveliwala. Topic-Sensitive PageRank. In *WWW '02: Proceedings of the 11th International Conference on World Wide Web*, pages 517–526, New York, NY, USA, 2002. ACM.

[18] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis. Resisting Structural Re-identification in Anonymized Social Networks. *Proceedings of the VLDB Endowment*, 1(1):102–114, August 2008.

[19] B. Hendrickson and R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing*, 16(2):452–469, 1995.

[20] B. Hendrickson and R. Leland. A Multi-Level Algorithm For Partitioning Graphs. In *Proceedings of the 1995 ACM/IEEE Conference on Supercomputing (CDROM)*, New York, NY, USA, 1995. ACM.

[21] P. Hintsanen and H. Toivonen. Finding reliable subgraphs from large probabilistic graphs. *Data Mining and Knowledge Discovery*, 17:3–23, August 2008.

[22] P. Hintsanen, H. Toivonen, and P. Sevon. Fast Discovery of Reliable Subnetworks. In *Proceedings of the 2010 International Conference on Advances in Social Networks Analysis and Mining*, pages 104–111, Los Alamitos, CA, USA, 2010. IEEE Computer Society.

[23] C. Hubler, H. P. Kriegel, K. Borgwardt, and Z. Ghahramani. Metropolis Algorithms for Representative Subgraph Sampling. In *Proceedings of the eighth IEEE International Conference on Data Mining*, pages 283–292, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[24] A. Inokuchi, T. Washio, and H. Motoda. An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 13–23, London, UK, 2000. Springer-Verlag.

[25] J.W. Jaromczyk and G.T. Toussaint. Relative Neighborhood Graphs and Their Relatives. *Proceedings of the IEEE*, 80(9):1502–1517, 1992.

[26] G. Jeh and J. Widom. Scaling Personalized Web Search. In *WWW '03: Proceedings of the 12th International Conference on World Wide Web*, pages 271–279, New York, NY, USA, 2003. ACM.

[27] B. H. Junker and F. Schreiber. *Analysis of Biological Networks*. Wiley Series in Bioinformatics. John Wiley & Sons, 2011.

[28] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20:359–392, 1998.

[29] B. W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *Bell System Technical Journal*, 49(1):291–307, 1970.

[30] J. Leskovec and C. Faloutsos. Sampling from Large Graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 631–636, New York, NY, USA, 2006. ACM.

[31] S. Lin and H. Chalupsky. Unsupervised Link Discovery in Multi-relational Data via Rarity Analysis. In *ICDM '03: Proceedings of the third IEEE International Conference on Data Mining*, page 171, Washington, DC, USA, 2003. IEEE Computer Society.

[32] A.S. Maiya and T.Y. Berger-Wolf. Sampling Community Structure. In *Proceedings of the 19th International Conference on World Wide Web*, pages 701–710, New York, NY, USA, 2010. ACM.

[33] G. L. Miller, S. H. Teng, W. Thurston, and S. A. Vavasis. Geometric Separators for Finite-Element Meshes. *SIAM Journal on Scientific Computing*, 19(2):364–386, 1998.

[34] S.C. Morris. *The Crucible of Creation: The Burgess Shale and the Rise of Animals*. Oxford University Press paperback. Oxford University Press, 2000.

[35] S. Muggleton. Duce, an oracle based approach to constructive induction. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 287–292, San Francisco, CA, USA, 1987. Morgan Kaufmann Publishers Inc.

[36] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph Summarization with Bounded Error. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 419–432, New York, NY, USA, 2008. ACM.

[37] S. Navlakha, M.C. Schatz, and C. Kingsford. Revealing Biological Modules via Graph Summarization. *Journal of Computational Biology*, 16(2):253–264, 2009.

[38] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004.

[39] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64:026118, Jul 2001.

[40] D. Peleg and A. A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

[41] A. Pothen, H. D. Simon, and K.-P. Liou. Partitioning Sparse Matrices with Eigenvectors of Graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, July 1990.

[42] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101:2658–2663, 2004.

[43] D. Rafiei. Effectively visualizing large networks through sampling. In *16th IEEE Visualization (VIS 2005)*, pages 375–382, Washington, DC, USA, 2005. IEEE Computer Society.

[44] R. W. Schvaneveldt. *Pathfinder Associative Networks: Studies in Knowledge Organization*. Ablex Publishing Corp., Norwood, NJ, USA, 1990.

[45] J. Scott. *Social Network Analysis: A Handbook*. Sage Publications, UK, January 2000.

[46] D. Shasha, J. T. L. Wang, and R. Giugno. Algorithmics and Applications of Tree and Graph Searching. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 39–52, New York, NY, USA, 2002. ACM.

[47] X. Shi, M. Bonner, L. A. Adamic, and A. C. Gilbert. The Very Small World of the Well-connected. In *Proceedings of the nineteenth ACM Conference on Hypertext and Hypermedia*, pages 61–70, New York, NY, USA, 2008. ACM.

[48] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient Aggregation for Graph Summarization. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 567–580, New York, NY, USA, 2008. ACM.

[49] Y. Tian, R. C. Mceachin, C. Santos, D. J. States, and J. M. Patel. SAGA: a subgraph matching tool for biological graphs. *Bioinformatics*, 23(2):232–239, 2007.

[50] Y. Tian and J. M. Patel. TALE: A Tool for Approximate Large Graph Matching. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 963–972, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[51] H. Tong and C. Faloutsos. Center-Piece Subgraphs: Problem Definition and Fast Solutions. In *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 404–413, New York, NY, USA, 2006. ACM.

[52] G. T. Toussaint. The Relative Neighbourhood Graph of a Finite Planar Set. *Pattern Recognition*, 12(4):261–268, 1980.

[53] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrodl. Constrained K-means Clustering with Background Knowledge. In *Proceedings of the eighteenth International Conference on Machine Learning*, pages 577–584, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[54] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[55] S. White and P. Smyth. Algorithms for Estimating Relative Importance in Networks. In *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 266–275, New York, NY, USA, 2003. ACM.

[56] C.R. Woese, O. Kandler, and M.L. Wheelis. Towards a natural system of organisms: proposal for the domains Archaea, Bacteria, and Eucarya. *Proceedings of the National Academy of Sciences*, 87(12):4576–4579, 1990.

[57] X. Yan, P. S. Yu, and J. Han. Graph Indexing: A Frequent Structure-based Approach. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 335–346, New York, NY, USA, 2004. ACM.

[58] N. Zhang, Y. Tian, and J. M. Patel. Discovery-Driven Graph Summarization. In *Proceedings of the 26th International Conference on Data Engineering*, pages 880–891, Los Alamitos, CA, USA, 2010. IEEE Computer Society.

[59] Y. Zhou, H. Cheng, and J. X. Yu. Graph Clustering Based on Structural/Attribute Similarities. *Proceedings of the VLDB Endowment*, 2(1):718–729, 2009.

[60] Y. Zhou, H. Cheng, and J. X. Yu. Clustering Large Attributed Graphs: An Efficient Incremental Approach. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, pages 689–698, Washington, DC, USA, 2010. IEEE Computer Society.

[61] R. Zou and L. B. Holder. Frequent Subgraph Mining on a Single Large Graph Using Sampling Techniques. In *Proceedings of the eighth Workshop on Mining and Learning with Graphs*, MLG '10, pages 171–178, New York, NY, USA, 2010. ACM.